



Cairo University
Faculty of Graduate Studies for Statistical Research
Department of Computer Science
ICare– Outpatient/Clinic Management System

**A Project Presented for Fulfillment
For Diploma Project in Computer Science**

Submitted By

- 1. Ali Mansour Mohamed (202300425)**
- 2. Thorya Mahfoz Ahmed (202300519)**
- 3. Mohamed Galal Basiouny (202300498)**

**Under supervision of
Dr. Shahira Azzazy**

**Cairo
May, 2025**

ABSTRACT

In the modern era, digitalization has become essential in all aspects, especially in the healthcare domain. Patients face difficulties in scheduling appointments, getting healthcare services, or managing consultations due to the lack of centralized and efficient systems. To address such issues, our project proposes a mobile-based application to render outpatient and clinic management easier.

Through a user-friendly Android platform, the system connects patients with clinics, doctors, pharmacies, and diagnostic centers while enabling healthcare staff to effectively manage appointments and consultations. The application has role-based access for patients, doctors, clinical staff, and administrators, enabling secure and controlled interactions.

With the integration of Firebase Authentication, RESTful APIs, and SQLite caching, the system offers stable, fast, and secure access to crucial medical services and records. The objective of this project is to enhance the healthcare experience for all by making available open and modern tools to ease the delivery and management of healthcare services.

ACKNOWLEDGMENT

At the start and end, we thank Allah for guiding us and providing us with patience and strength to complete this work and ask Allah to benefit us with what he taught us and teach us what will benefit us.

We are expressively thankful for the continuous guidance and supervision of **Dr. Shahira Azzazy**, who provided invaluable advice and support in all phases of this project.

We would like to express our thanks to all the departmental faculty members of the Faculty of Graduate Studies for Statistical Research, who have helped in shaping our academic background.

Last but not least, our heartfelt thanks to our families for their constant support, sacrifices, and motivation, which provided us with power to overcome hurdles and reach this milestone.

TABLE OF CONTENTS

1.	Chapter 1	11
1.1.	Introduction.....	11
1.2.	System Overview	12
1.3.	Project Scope.....	13
1.3.1.	In-Scope Features	13
1.3.2.	Out-of-Scope Features	14
1.4.	Deliverables	14
1.5.	Technologies Utilized in the ICare Android Application	14
1.5.1.	<i>Mobile Development</i>	14
1.5.2.	<i>Backend Development</i>	15
1.5.3.	<i>Authentication & Security</i>	15
1.5.4.	<i>Design & Prototyping</i>	15
1.5.5.	<i>Other Tools</i>	15
1.6.	Problem Analysis.....	16
1.6.1.	<i>Current Challenges</i>	16
1.6.2.	<i>Need for a Mobile-Based ICare System</i>	17
2.	Chapter 2	19
2.1.	Architecture Style.....	19
2.2.	Clean Architecture Layers.....	19
2.2.1.	UI Layer.....	19
2.2.2.	Domain Layer	19
2.2.3.	Data Layer.....	20
2.3.	Feature-Based Modularization.....	20
2.4.	Dependency Dynamics	21
	21
2.5.	Functional Requirements	22
2.6.	User Stories.....	23
2.7.	Work Backlog.....	24
2.8.	Project Duration	24
2.9.	Use Case Diagrams	25
2.9.1.	Authentication use case diagram.....	25
2.9.2.	Home Use Case Diagram.....	26
2.9.3.	Consultation Use Case Diagram.....	27
2.9.4.	Administration use case diagram	28
2.10.	Actor Description	29
2.11.	Use Case Description	30
2.12.	Traceability Matrix	56

2.13	Activity Diagrams	59
2.14.	Sequence Diagrams.....	70
2.15.	Wire Frames prototyping	81
2.15.1.	Wireframes screens sample	81
2.16.	Non-functional Requirements	83
2.15.1	Performance Requirements:	83
2.15.2	Security Requirements:.....	83
2.15.3	Availability & Reliability Requirements:.....	83
2.15.4	Usability Requirements:	83
2.15.5	Maintainability & Scalability Requirements:.....	83
2.15.6	Compatibility Requirements:	83
2.15.7	Backup & Data Integrity Requirements:	83
3.	Chapter 3	85
3.1.	Component Diagram	85
3.2.	Class Diagram.....	86
3.3.	Enhanced Entity Relationship Diagram (EERD).....	87
3.4.	Mapping.....	88
4.	Chapter 4	90
4.1.	Testing (User Acceptance Test)	90
4.1.1.	Test Cases.....	90
5.	Future Work	101
6.	References.....	102

TABLE OF FIGURES

Figure 2.4.1 Icare clean system Architecture MVVM overview	21
Figure 2.9.1.1 Authentication Use case Diagram.....	25
Figure 2.9.2.1 Home use Case diagram.....	26
Figure 2.9.3.1 consultation use case diagram.....	27
Figure 2.9.4.1 Administration Use case diagram.....	28
Figure 2.13.1 Patient Register an Account Activity Diagram	59
Figure 2.13.2 User Login Using Email & Password Activity Diagram.....	60
Figure 2.13.3 User Link Google Account Activity Diagram	60
Figure 2.13.4 User Login Using Google Account Activity Diagram	61
Figure 2.13.5 Patient Book Appointment Activity Diagram.....	61
Figure 2.13.6 Clinic Staff Confirm Appointment Activity Diagram	62
Figure 2.13.7 Doctor Add Consultation Activity Diagram	62
Figure 2.13.8 Admin Add a New ENTITY (Clinic, Doctor, Clinic Staff, Pharmacy, Pharmacist, Center, or Center Staff) Activity Diagram	63
Figure 2.13.9 Admin View Cached Entity (Clinic, Doctor, Pharmacy, Center) List Activity Diagram	64
Figure 2.13.10 Admin Update Existing Entity (Clinic, Doctor, Clinic Staff, Pharmacy, Pharmacist, Center, or Center Staff) Activity Diagram	64
Figure 2.13.11 Admin View Non-Cashable Entity (Clinic Staff, Pharmacist, or Center Staff) List Activity Diagram	65
Figure 2.13.12 User Fetch Data (Clinics, Pharmacies, Centers, and Doctors) From The BACKEND ACTIVITY Diagram.....	65
Figure 2.13.13 Doctor View His Schedule and Update His Service Price Activity Diagram	65
Figure 2.13.14 Patient Re-schedule Appointment Activity Diagram	67
Figure 2.13.15 Patient View Top Doctors Activity Diagram	67
Figure 2.13.16 Patient Cancel Appointment Activity Diagram.....	68
Figure 2.13.17 Patient View Cached Entity (Doctor, Pharmacy, Center) List Activity Diagram	68
Figure 2.13.18 Doctor View Medical Record and Update Consultation Activity Diagram.....	69
Figure 2.13.19 Clinic Staff View Pending and Confirmed Appointments Activity Diagram.....	69
Figure 2.14.1 Admin Add a New Entity(Clinic, Doctor, Clinic Staff, Pharmacy, Pharmacist, Center, or Center Staff) Sequence Diagram	70
Figure 2.14.2 Admin View Cached Entity (Clinic, Doctor, Pharmacy, Center) List Sequence Diagram	70
Figure 2.14.3 Admin Update Existing Entity(Clinic, Doctor, Clinic Staff, Pharmacy, Pharmacist, Center, or Center Staff) Sequence Diagram	71
Figure 2.14.4 Admin View Non-Cashable Entity (Clinic Staff, Pharmacist, or Center Staff) List Sequence Diagram	72
Figure 2.14.5 Clinic Staff Confirm Appointment Sequence Diagram.....	72
Figure 2.14.6 Clinic Staff View Pending and Confirmed Appointments Sequence Diagram.....	72
Figure 2.14.7 Doctor Add Consultation Sequence Diagram	73
Figure 2.14.8 Doctor View Medical Record and Update Consultation Sequence Diagram.....	73
Figure 2.14.9 Patient View Cached Entity (Doctor, Pharmacy, Center) List Sequence Diagram	74
Figure 2.14.10 Patient View Top Doctors Sequence Diagram.....	74
Figure 2.14.11 User Fetch Data (Clinics, Pharmacies, Centers, and Doctors) From The Backend Sequence Diagram.....	75
Figure 2.14.12 User Link Google Account Sequence Diagram	75
Figure 2.14.13 User Login Using Google Account Sequence Diagram	76
Figure 2.14.14 Doctor View His Schedule and Update His Service Price Sequence Diagram	77
Figure 2.14.15 User Login Using Email & Password Activity Diagram	78
Figure 2.14.16 Patient Booking Appointment Sequence Diagram	79
Figure 2.14.17 Patient Canceling An Appointment Sequence Diagram	79
Figure 2.14.18 Patient Register an Account Sequence Diagram	80
Figure 2.14.19 Patient Rescheduling An Appointment Sequence Diagram	80
Figure 2.15.1.1 Clinic Admin Home.....	81
Figure 2.15.1.2 Scheduled Appointments.....	81

Figure 2.15.1.4 Home service listing.....	82
Figure 2.15.1.3 service booking - choose doctor	82
Figure 3.1.1 Component Diagram	85
Figure 3.2.1 class diagram	86
Figure 3.3.1 ENHANCED ENTITY RELATIONSHIP DIAGRAM (EERD).....	87
Figure 3.4.1 Mapping Scheme	88

TABLE OF TABLES

Table 2.5.1 Functional Requirements.....	22
Table 2.6.1 User Stories.....	23
Table 2.7.1 Work backlog	24
Table 2.10.1 Actor Description.....	29
Table 2.11.1 (uc-1) Register Patient Account	30
Table 2.11.2 (uc-2) User Login Using Email/Password	31
Table 2.11.3 (uc-3) Link Google Account.....	32
Table 2.11.4 (uc-4) Login Using Linked Google Account.....	32
Table 2.11.5 (uc-5) Cache Clinics List	33
Table 2.11.6 (uc-6) Cache Doctors List.....	33
Table 2.11.7 (uc-7) Cache Imaging Centers List.....	34
Table 2.11.8 (uc-8) Book Appointment.....	34
Table 2.11.9 (uc-9) Reschedule Appointment.....	35
Table 2.11.10 (uc-10) Cancel Appointment.....	35
Table 2.11.11 (uc-11) View List of Top Doctors.....	36
Table 2.11.12 (uc-12) View Contracted Pharmacies.....	36
Table 2.11.13 (uc-13) View contracted lab centers	37
Table 2.11.14 (uc-14) View Contracted Imaging Centers	37
Table 2.11.15 (uc-15) Clinical Staff View Appointments.....	38
Table 2.11.16 (uc-16) Confirm Pending Appointments	38
Table 2.11.17 (uc-17) View Doctor Appointment Schedule	39
Table 2.11.18 (uc-18) Doctor Update Service Price.....	39
Table 2.11.19 (uc-19) Add Consultation	40
Table 2.11.20 (uc-20) View Clinics List.....	40
Table 2.11.21 (uc-21) Add New Clinic	41
Table 2.11.22 (uc-22) Update Existing Clinic	42
Table 2.11.23 (uc-23) View Doctors List	43
Table 2.11.24 (uc-24) Add New Doctor.....	43
Table 2.11.25 (uc-25) Update Existing Doctor.....	44
Table 2.11.26 (uc-26) View Clinic Staff List.....	45
Table 2.11.27 (uc-27) Add New Clinic Staff Member	45
Table 2.11.28 (uc-28) Update Existing Clinic Staff Member	46
Table 2.11.29 (uc-29) View Pharmacy List.....	47
Table 2.11.30 (uc-30) Add New Pharmacy	47
Table 2.11.31 (uc-31) Update Existing Pharmacy	48
Table 2.11.32 (uc-32) View Pharmacists List.....	49
Table 2.11.33 (uc-33) Add New Pharmacist	50
Table 2.11.34 (uc-34) Update Existing Pharmacist	51
Table 2.11.35 (uc-35) View Lab/Imaging Centers List	52
Table 2.11.36 (uc-36) Add New Lab/Imaging Center.....	52
Table 2.11.37 (uc-37) Update Existing Lab/Imaging Center	53
Table 2.11.38 (uc-38) View Center Staff List	54
Table 2.11.39 (uc-39) Add New Center Staff Member	55
Table 2.11.40 (uc-40) Update Existing Center Staff Member	56
Table 2.12.1 TRACEABILITY MATRIX.....	57
Table 4.1.1.1 TC-1.....	90
Table 4.1.1.2 TC-2.....	90
Table 4.1.1.3 TC-3.....	90
Table 4.1.1.4 TC-4.....	91
Table 4.1.1.5 TC-5.....	91

Table 4.1.1.6 TC-6.....	91
Table 4.1.1.7 TC-7.....	92
Table 4.1.1.8 TC-8.....	92
Table 4.1.1.9 TC-9.....	92
Table 4.1.1.10 TC-10	93
Table 4.1.1.11 TC-11	93
Table 4.1.1.12 TC-12	93
Table 4.1.1.13 TC-13	93
Table 4.1.1.14 TC-14	94
Table 4.1.1.15 TC-15	94
Table 4.1.1.16 TC-16	94
Table 4.1.1.17 TC-17	94
Table 4.1.1.18 TC-18	95
Table 4.1.1.19 TC-19	95
Table 4.1.1.20 TC-20	95
Table 4.1.1.21 TC-21	95
Table 4.1.1.22 TC-22	96
Table 4.1.1.23 TC-23	96
Table 4.1.1.24 TC-24	96
Table 4.1.1.25 TC-25	96
Table 4.1.1.26 TC-26	97
Table 4.1.1.27 TC-27	97
Table 4.1.1.28 TC-28	97
Table 4.1.1.29 TC-29	97
Table 4.1.1.30 TC-30	98
Table 4.1.1.31 TC-31	98
Table 4.1.1.32 TC-32	98
Table 4.1.1.33 TC-33	98
Table 4.1.1.34 TC-34	99
Table 4.1.1.35 TC-35	99
Table 4.1.1.36 TC-36	99
Table 4.1.1.37 TC-37	99
Table 4.1.1.38 TC-38	100
Table 4.1.1.39 TC-39	100
Table 4.1.1.40 TC-40	100

CHAPTER 1

INTRODUCTION

1. CHAPTER 1

1.1. INTRODUCTION

The last few years have seen a significant influence on the fast development of technology in all sectors of human living, including health care services.

Mobile applications have become essential tools for improving healthcare provision, patient interaction, and simplifying clinical procedures management.

The following graduation project discusses the development of a comprehensive medical care Android application with the objective of simplifying and automating the interaction between patients and health care practitioners in a secure, accessible, and convenient platform.

The targeted use is an interface uniting patient with various healthcare services like clinics, pharmacies, diagnostic labs, and medical professionals.

With the application of modern mobile platform development and cloud infrastructures, the system aims at enhancing healthcare accessibility, improving the scheduling of appointments, and ensuring easier overall management of medical consultations and administrative services.

Embedded at the core of the application is a system that has a formal four-tiered user role base of Admin, Clinical Staff, Doctor, and Patient. All these roles have permissions and functionality relevant to their operational needs.

Patients can simply register, book, reschedule, and cancel appointments and even search through a list of available doctors, clinics, pharmacies, lab centers, and imaging centers.

The Clinical Staff role focuses on scheduling and confirming appointments for efficient scheduling and resource management.

Doctors can see their calendars by day and add consultations for completed appointments to ensure seamless patient care documentation.

On the other hand, the admin has the power to manage all the objects within the system, e.g., creating and editing clinics, doctors, staff members, pharmacies, diagnostic centers, etc.

For secure patient authentication and data integrity, Firebase Authentication is utilized in the application to allow patients to register and log in based on a safe cloud-based identity management service. Admin, Clinical Staff, and Doctor accounts are automatically created by the system administrator via the admin dashboard.

RESTful APIs are employed to establish a communication channel between the mobile app and the backend database server for efficient, scalable, and organized data exchange.

One of the most differentiating features of the system is that it can cache vital information locally on the mobile device using SQLite in order to have an even smoother user experience through the capability of browsing frequently required information such as clinic, doctor, and pharmacy listings without constant internet dependence. This not just improves application responsiveness but also eliminates server load and data transfer cost.

1.2. System Overview

The planned ICare Android app is a combined mobile-based healthcare management system designed to maximize the interaction among medical professionals, patients, and administrative staff. The system leverages modern mobile development frameworks, cloud-based authentication services, and RESTful web services to deliver an efficient, secure, and reliable healthcare service platform. The system is designed to simplify appointment management, enhance easy access to healthcare resources, and improve the overall user experience for both medical professionals and patients.

The system is based on four independent user roles with clear functionalities and permissions:

- Admin
- Clinic Staff
- Doctor
- Patient

All the roles interact with the system using a different mobile interface suitable to their operational needs for ease and clarity. The basic functionalities and access rights of each role are clearly established for the maintenance of system integrity and easy operation workflows.

Mobile Application

The primary user interface is an Android smartphone application intended to support all user roles. The application enables easy navigation and role-based access to system functions, enabling users to do a number of things such as scheduling appointments, calendaring management, and viewing listings of healthcare services. The application sends requests to the centralized backend system through RESTful APIs, which handles data requests, processing of operations, and secure data exchange. In addition, the application includes a local SQLite database for caching of frequently accessed information such as clinic, doctor, and pharmacy lists. This improves performance and allows users to access critical information even in case of poor connectivity.

Authentication and Security

User authentication and security are managed through Firebase Authentication, a cloud-hosted identity service that supports email/password and third-party authentication. While administrators can sign up their accounts via the application, Admin, Clinical Staff, and Doctor accounts are created and managed directly by the admin using the system's admin controls.

Backend Server and Database

There is one, concentrated backend server that addresses all business logic, manages user requests, and maintains data consistency. The backend communicates with a properly structured relational database in which details of users, appointments, consultations, clinics, pharmacies, diagnostic centers, etc., are retained. CRUD operations such as creating, updating, fetching, and deleting records are performed using well-documented RESTful API endpoints.

System Entities and Interactions

The system manages a wide range of entities related to healthcare like:

- Clinics
- Doctors
- Clinical Staff
- Pharmacies and Pharmacists
- Lab Centers and Imaging Centers

- Appointments
- Consultations

All the entities are listable, addable, and updatable by utilizing some system functionality based on the user role. The interface among these elements follows a client-server architecture, ensuring modularity, scalability, and ease of maintenance.

Key Features

- Role-based access control for Admin, Clinical Staff, Doctor, and Patient
- Patient self-registration and secure login through Firebase Authentication
- Appointment booking, rescheduling, cancelation, and confirmation
- Display of doctors, clinics, pharmacies, and diagnostic centers listings
- Physician management of consultations
- Centralized RESTful API for efficient communication with backend services.
- Offline access to data with local SQLite cache
- System administrator tools with timely management

1.3. PROJECT SCOPE

1.3.1. IN-SCOPE FEATURES

User Authentication and Authorization

- Patient registration and login using Firebase Authentication.
- Role-based login for Admin, Doctor, and Clinical Staff (Admin creates accounts).

Appointment Management

- Patients can book, reschedule, and cancel doctor appointments.
- Clinical Staff can view and approve patient appointments.
- Doctors can view their schedule of appointments and complete appointments by adding consultations.

Medical Service Listings

- Patients can search for cached, locally stored clinics, doctors, pharmacies, lab centers, and imaging centers using SQLite for better offline access and application performance.

Administrative Management

- Admin can add, update, and manage system entities, such as:
 - Clinics
 - Doctors
 - Clinic Staff
 - Pharmacies and Pharmacists
 - Lab and Imaging Centers
 - Center Staff

Consultation Management

- Doctors can add consultations for confirmed appointments.
- System updates appointment status on consultation completion.

Backend and Data Management

- Communication with a backend server through RESTful APIs for data exchange.
- Relational database server to house system data like user profiles, appointments, consultations, and service listings.

Security and Data Integrity

- Secure authentication and transfer of data through Firebase Authentication and HTTPS-enabled APIs.
- Local caching to enhance performance and ensure access to critical data during low connectivity times.

1.3.2. OUT-OF-SCOPE FEATURES

- Versions of the application for iOS or web-based platforms.
- Web-based or online payment systems or medical services billing management.
- Real-time video consultation or chat feature.
- Electronic health record (EHR) integration or management of medical history.
- Automated appointment reminders via SMS or email and tracking.
- Advanced reporting or analytics dashboards for doctors.
- Inventory management of diagnostics centers or pharmacies.
- Insurance company integration.
- Virtual Consultations: Integrate video consultations for patients who cannot physically come for consultation due to distance or other constraints.
- Integration of External Referral System
- Pharmacy, Laboratory, and Imaging Center Integration: Pharmacies will be able to suggest substitute medications considering stock status.

1.4. DELIVERABLES

Upon project completion, the following deliverables will be produced:

- Final Android application for all user roles.
- RESTful API backend service.
- Relational database schema and deployment.
- System analysis, design, implementation, and testing documentation.
- System diagrams like: use case, sequence, class, and database diagrams

1.5. TECHNOLOGIES UTILIZED IN THE ICARE ANDROID APPLICATION

ICare Android application utilizes a carefully selected set of modern technologies, platforms, and tools for efficient, scalable, and secure operation. The technologies below were utilized during the project:

1.5.1. MOBILE DEVELOPMENT

- **Android SDK (Kotlin)**
Utilized for developing the native Android application, which provides the most important tools and APIs for creating user interfaces, managing device resources, and communicating with backend services.
- **Android Studio**
The Android Studio official integrated development environment (IDE) for Android app development, offering powerful code editing, debugging, and performance optimization features.
- **Room DB (SQLite)**
Light-weight, in-process relational database to cache critical healthcare service data such as clinic, doctor, and pharmacy listings on the user's device for improved offline availability and application responsiveness.

1.5.2. BACKEND DEVELOPMENT

- **RESTful APIs (Spring Boot)**
RESTful web services are used for client-server communication to read, submit, and update data securely. Business logic is obtained via these APIs, and data transfer from the mobile app to the backend database.
- **Relational Database (Microsoft SQL Server 2017)**
Relational database with structure stores system data like users, appointments, consultations, and service listings. It gives good data integrity and fast querying.

1.5.3. AUTHENTICATION & SECURITY

- **Firebase Authentication**
A cloud identity management service utilized to securely handle patient registration and login. It supports email/password and social accounts authentication, and integrates well with Android apps, providing user management and session control.
- **HTTPS / SSL Certificates**
All data exchanges between the mobile application and backend services are encrypted using HTTPS protocols to protect sensitive data transmissions.

1.5.4. DESIGN & PROTOTYPING

- **Draw.io / Diagrams.net**
An online diagramming tool used to design system diagrams, e.g., use case diagrams, sequence diagrams, and database schemas.
- **Visily**
An online wireframing tool used to design all screens wireframes.

1.5.5. OTHER TOOLS

- **Postman**
An API development and testing tool used to validate the functionality, security, and performance of RESTful APIs during the development process.
- **Git & GitHub**
Version control systems that are used to keep source code, track changes, and facilitate collaborative development during the project.

These technologies were selected based on their stability, compatibility, and ease of integration with the Android app environment. Together, they enable the creation of a secure, responsive, and user-friendly healthcare management app that meets modern development standards.

1.6. PROBLEM ANALYSIS

Appointment, consultation, and management of healthcare resources in most healthcare environments, particularly where there are not integrated digital solutions, remain time-consuming, disjointed, and manual exercise. Patients and clinicians experience a variety of operational issues with no single, streamlined, and user-friendly system in place. They typically result in inadequate accessibility of services, unnecessary delays in patient care, and wasteful administrative processes.

1.6.1. CURRENT CHALLENGES

- **Inefficient Appointment Management**
 - In conventional healthcare settings, clients are required to visit or call clinics and hospitals to make appointments. This can be time-consuming, inconvenient, and prone to scheduling errors or miscommunication.
 - Appointment rescheduling or cancellation entails direct human contact with the clinical team, adding to the administrative workload and lengthening the time for patients and healthcare providers.
- **Restricted Access to Healthcare Service Information**
 - Patients tend to lack timely information about available clinics, doctors, pharmacies, and diagnostic centers. This discourages them from making informed decisions among healthcare providers based on choice or distance.
 - Without an internet directory, patients are left to rely on out-of-date details, word of mouth, or visits to sites.
- **Splintered Consultation and Record Management**
 - Lacking a centralized system, doctors can rely on paper appointment books or stand-alone electronic records to manage their schedules and consultations. This can lead to lost or repeat appointments, as well as incomplete documentation of consultations.
 - Inadequate or incomplete patient appointment histories reduce the quality and continuity of care.
- **Healthcare Staff Administrative Burden**
 - Clinical and administrative staff spend considerable amounts of time working with appointment requests, confirming bookings, and manually updating service provider records.
 - Lack of streamlined computer management tools raises the risk of human error and lowers overall operational efficiency.
- **Security and Authentication Issues**
 - No system, if any, has secure user authentication systems in place, making patient and appointment sensitive information vulnerable to unauthorized access.
 - Normal login and account management systems are typically prone to manual intervention during account setup and management, increasing security risks.

1.6.2. NEED FOR A MOBILE-BASED ICARE SYSTEM

With the existing growth of mobile technology and increasing reliance on digital services for everyday things, an obvious need is for a mobile-based medical care application. This would enable patients to:

- Schedule, cancel, and reschedule appointments at their convenience through their smartphones.
- Retrieve up-to-date information about healthcare service providers.
- Reduce visits or calls required to schedule appointments.

At the same time, it would be enabling administrative staff and care providers to:

- Simplify scheduling and appointment management.
- Provide a single platform for consultation and scheduling management.
- Enable operational efficiency with secure, cloud-based communication and authentication.

The objective of this project is to address the above challenges through a secure, accessible, and user-friendly Android application that enables access to healthcare services, enhances patient-provider communication, and modernizes operational procedures within healthcare facilities.

CHAPTER 2

REQUIREMENT ANALYSIS

2. CHAPTER 2

2.1. ARCHITECTURE STYLE

The ICare Android application utilizes the Model-View-View Model (MVVM) pattern for a proper separation of concerns, maintainable codebase, and scalable development.

ICare Android application is built following the Clean Architecture principles, a layered architectural pattern that enforces a rigid separation of concerns and dependency management. This enhances the maintainability, testability, scalability, and modularity of the application.

The system is organized into separate, self-deployable Android Library Modules, each with its specific function.

The architecture follows the dependency principle of Clean Architecture, with dependencies always pointing inwards to the innermost circle of core business logic, thereby ensuring the outer layers are dependent on the inner layers, and not vice versa.

2.2. CLEAN ARCHITECTURE LAYERS

The application is organized into three primary layers:

2.2.1. UI LAYER

- Purpose: Handles all user interface and presentation logic.
- Technology: Written with Jetpack Compose, a new declarative UI framework for Android.
- Responsibilities:
 - Writing layouts, screens, and UI elements.
 - Handling screen states with View Models.
 - Calling Use Cases to execute business logic operations.
 - Gathering and displaying data received from View Models.

2.2.2. DOMAIN LAYER

- Purpose: Represents the core business logic and rules of the application.
- Structure: Includes models, use cases, and repository interfaces.
- Responsibilities:
 - Declaring system objects models as data classes
 - Identifying use cases for processes such as patient registration, scheduling an appointment, confirming an appointment, and inserting consultations.
 - The repository interfaces, such as AuthRepository, AppointmentRepository, and ConsultationRepository, hold the data manipulation methods that the use cases need.

2.2.3. DATA LAYER

- Purpose: Manages data operations, networking, and local data caching.
- Structure: Contains data sources, repository implementations, and model mapping utilities.
- Responsibilities:
 - Implementing Repository Interfaces of the Domain layer.
 - Exposing Remote Data Sources via Retrofit to interact with RESTful APIs.
 - Managing Local Persistence and Offline Caching using Room Database for local persistence and offline caching.
 - Mapping data models among Entities, Domain Models, and DTOs (Data Transfer Objects).

2.3. FEATURE-BASED MODULARIZATION

To enhance code organization and scalability even more, each major feature is developed in its own Android Library Module. This feature modularization provides:

- Improved delineation of responsibilities.
- Greater construction efficiency.
- Separate development, testing, and deployment for each feature.
- Examples of feature modules:
 - feature-auth
 - feature-home
 - feature-appointments
 - feature-consultations
 - feature-admin
 - feature-settings

Each of its feature modules obeys Clean Architecture's internal layering, maintaining its user interface, domain, and data distinct.

2.4. DEPENDENCY DYNAMICS

The dependency chain goes as follows:

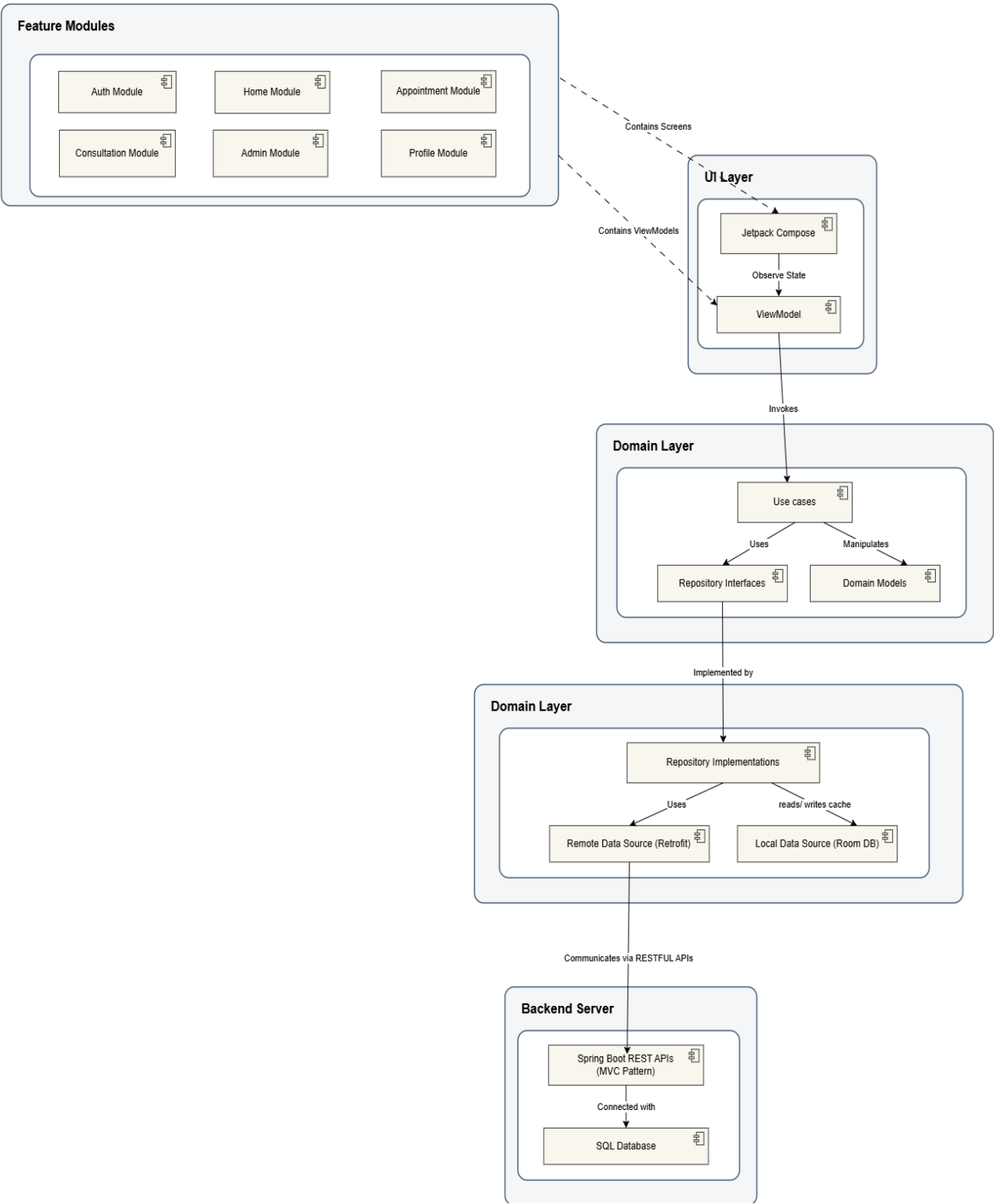


FIGURE 2.4.1 ICARE CLEAN SYSTEM ARCHITECTURE MVVM OVERVIEW

2.5. FUNCTIONAL REQUIREMENTS

TABLE 2.5.1 FUNCTIONAL REQUIREMENTS

ID	Priority	Requirement Description
REQ1	5	The system shall allow patients to register a new account using a registration form.
REQ2	5	The system shall allow all users (Admin, Clinical Staff, Doctor, Patient) to log in through a login screen using their registered email and password.
REQ3	3	The system should allow all users (Admin, Clinical Staff, Doctor, Patient) to link their Google account for login.
REQ4	3	The system should allow all users (Admin, Clinical Staff, Doctor, Patient) to log in through a login screen using their linked Google account
REQ5	4	The system shall fetch the lists of clinics from the backend server and cache them onto the user's device after login.
REQ6	4	The system shall fetch the lists of doctors and cache them onto the user's device after login.
REQ7	4	The system shall fetch the lists/imaging centers from the backend server and cache them onto the user's device after login.
REQ8	5	The system shall allow the patient to book an appointment by selecting a doctor and one of the available time slots.
REQ9	4	The system shall allow the patient to reschedule his existing appointment.
REQ10	4	The system shall allow the patient to cancel his existing appointment.
REQ11	3	The system should allow the patient to view a list of top doctors.
REQ12	4	The system shall allow the patient to view a list of contracted pharmacies.
REQ13	4	The system shall allow the patient to view a list of contracted lab centers.
REQ14	4	The system shall allow the patient to view a list of contracted imaging (scanning) centers.
REQ15	4	The system shall allow the Clinical Staff to view a list of pending and confirmed appointments.
REQ16	4	The system shall allow the Clinical Staff to confirm the pending appointments.
REQ17	4	The system shall allow the Doctor to view their appointment schedule, including pending and confirmed appointments.
REQ18	4	The system shall allow the Doctor to update his service price.
REQ19	4	The system shall allow the Doctor to add a consultation for a selected appointment and update its status to completed.
REQ20	5	The system shall allow the admin to view the clinics list.
REQ21	5	The system shall allow the admin to add a new clinic.
REQ22	5	The system shall allow the admin to update an existing clinic.
REQ23	5	The system shall allow the admin to view the doctors list.
REQ24	5	The system shall allow the admin to add a new doctor.
REQ25	5	The system shall allow the admin to update an existing doctor.
REQ26	5	The system shall allow the admin to view the clinic staff list.
REQ27	5	The system shall allow the admin to add a new clinic staff member.
REQ28	5	The system shall allow the admin to update an existing clinic staff member.
REQ29	5	The system shall allow the admin to view the pharmacy list.
REQ30	5	The system shall allow the admin to add a new pharmacy.
REQ31	5	The system shall allow the admin to update an existing pharmacy.
REQ32	5	The system shall allow the admin to view the pharmacists list.
REQ33	5	The system shall allow the admin to add a new pharmacist.
REQ34	5	The system shall allow the admin to update an existing pharmacist.
REQ35	5	The system shall allow the admin to view the lab/imaging centers list.
REQ36	5	The system shall allow the admin to add a new lab/imaging center.
REQ37	5	The system shall allow the admin to update an existing lab/imaging center.
REQ38	5	The system shall allow the admin to view the center staff list.
REQ39	5	The system shall allow the admin to add a new center staff member.
REQ40	5	The system shall allow the admin to update an existing center staff member.
REQ41	4	The system shall manage access control for each user type, ensuring they only access their permitted functionalities.
REQ42	2	The system shall send automated push notifications to remind patients of upcoming appointments or to confirm appointment status changes.
REQ43	1	The system should allow the admin to send notifications to the system users.
REQ44	1	The system should allow patients to rate and review doctors after completing a consultation.

ID	Priority	Requirement Description
REQ45	1	The system should allow patients to mark preferred doctors, pharmacies, or centers as favorite for quick access.

2.6. USER STORIES

TABLE 2.6.1 USER STORIES

Identifier	USER STORY	SIZE
ST-1	As a patient, I can register a new account using a registration form.	8pt
ST-2	As a user, I can log in using my registered email and password.	8pt
ST-3	As a user, I can link my Google account to my ICare account.	4pt
ST-4	As a user, I can log in using my linked Google account.	4pt
ST-5	As a patient, I can book an appointment with a doctor by choosing a date and time slot.	14pt
ST-6	As a patient, I can reschedule my existing appointment.	4pt
ST-7	As a patient, I can cancel an appointment if needed.	2pt
ST-8	As an admin, I can view, add, and update the clinics list.	8pt
ST-8	As an admin, I can view, add, and update the doctors list.	8pt
ST-9	As an admin, I can view, add, and update the clinic staff list.	8pt
ST-10	As an admin, I can view, add, and update the pharmacies list.	8pt
ST-11	As an admin, I can view, add, and update the pharmacists list.	8pt
ST-12	As an admin, I can view, add, and update the lab and imaging centers list.	8pt
ST-13	As an admin, I can view, add, and update pharmacists and lab/imaging center staff lists.	8pt
ST-14	As a doctor, I can view my appointment schedule and mark completed consultations.	20pt
ST-15	As a clinical staff member, I can confirm pending appointments.	4pt
ST-16	As a patient, I can view a list of top-rated doctors.	2pt
ST-17	As a patient, I can mark doctors, pharmacies, or imaging centers as favorites for quick access.	4pt
ST-18	As a registered user, I can only access the functionalities permitted for my role so that unauthorized access is prevented.	8pt
ST-19	As a clinical staff member, I can view a list of pending and confirmed appointments.	2pt
ST-20	As a doctor, I can update my service price.	1pt
ST-21	As a patient, I can view lists of contracted pharmacies, lab centers, and imaging centers.	12pt

2.7. WORK BACKLOG

TABLE 2.7.1 WORK BACKLOG

Work items	User story	Iteration no.	Estimated work duration
1)	ST-1 – Register a new patient account	Iteration 1	8pt (1 day)
2)	ST-2 – Log in using email and password	Iteration 1	8pt (1 day)
3)	ST-8 – Admin manages clinics	Iteration 1	8pt (1 day)
4)	ST-8 – Admin manages doctors	Iteration 1	8pt (1 day)
5)	ST-9 – Admin manages clinic staff	Iteration 2	8pt (1 day)
6)	ST-10 – Admin manages pharmacies	Iteration 2	8pt (1 day)
7)	ST-11 – Admin manages pharmacists	Iteration 2	8pt (1 day)
8)	ST-12 – Admin manages lab/imaging centers	Iteration 3	8pt (1 day)
9)	ST-13 – Admin manages center staff	Iteration 3	8pt (1 day)
10)	ST-5 – Book an appointment	Iteration 3	16pt (2 days)
11)	ST-6 – Reschedule appointment	Iteration 4	4pt (0.5 day)
12)	ST-7 – Cancel appointment	Iteration 4	2pt (0.25 day)
13)	ST-15 – Confirm pending appointments	Iteration 4	4pt (0.5 day)
14)	ST-20 – View appointments	Iteration 4	2pt (0.25 day)
15)	ST-14 – Doctor schedule & consultations	Iteration 5	20pt (2.5 days)
16)	ST-21 – Doctor updates service price	Iteration 5	2pt (0.25 day)
17)	ST-22 – View pharmacies/labs/imaging centers	Iteration 5	12pt (1.5 days)
18)	ST-3 – Link Google account	Iteration 6	4pt (0.5 day)
19)	ST-4 – Log in with linked Google	Iteration 6	4pt (0.5 day)
20)	ST-16 – View top-rated doctors	Iteration 6	2pt (0.25 day)
21)	ST-18 – Role-based access control	Iteration 6	8pt (1 day)

2.8. PROJECT DURATION

Total work size = 8 + 8 + 8 + 8 + 8 + 8 + 8 + 8 + 8 + 8 + 16 + 4 + 2 + 4 + 2 + 20 + 2 + 12 + 4 + 4 + 2 + 8 = 152 points
 = 152 / 8 = 19 days

2.9. USE CASE DIAGRAMS

2.9.1. AUTHENTICATION USE CASE DIAGRAM

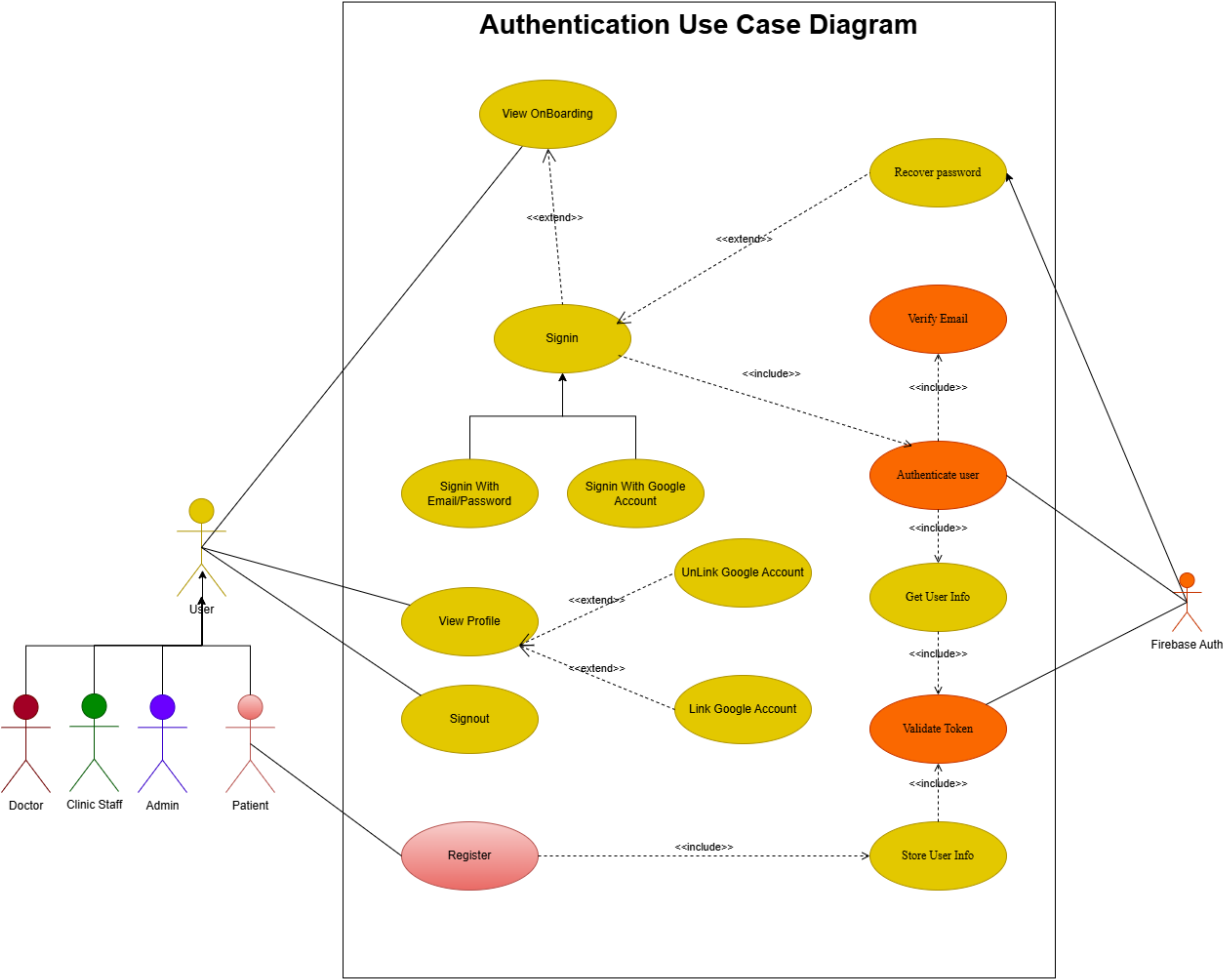


FIGURE 2.9.1.1 AUTHENTICATION USE CASE DIAGRAM

2.9.2. HOME USE CASE DIAGRAM

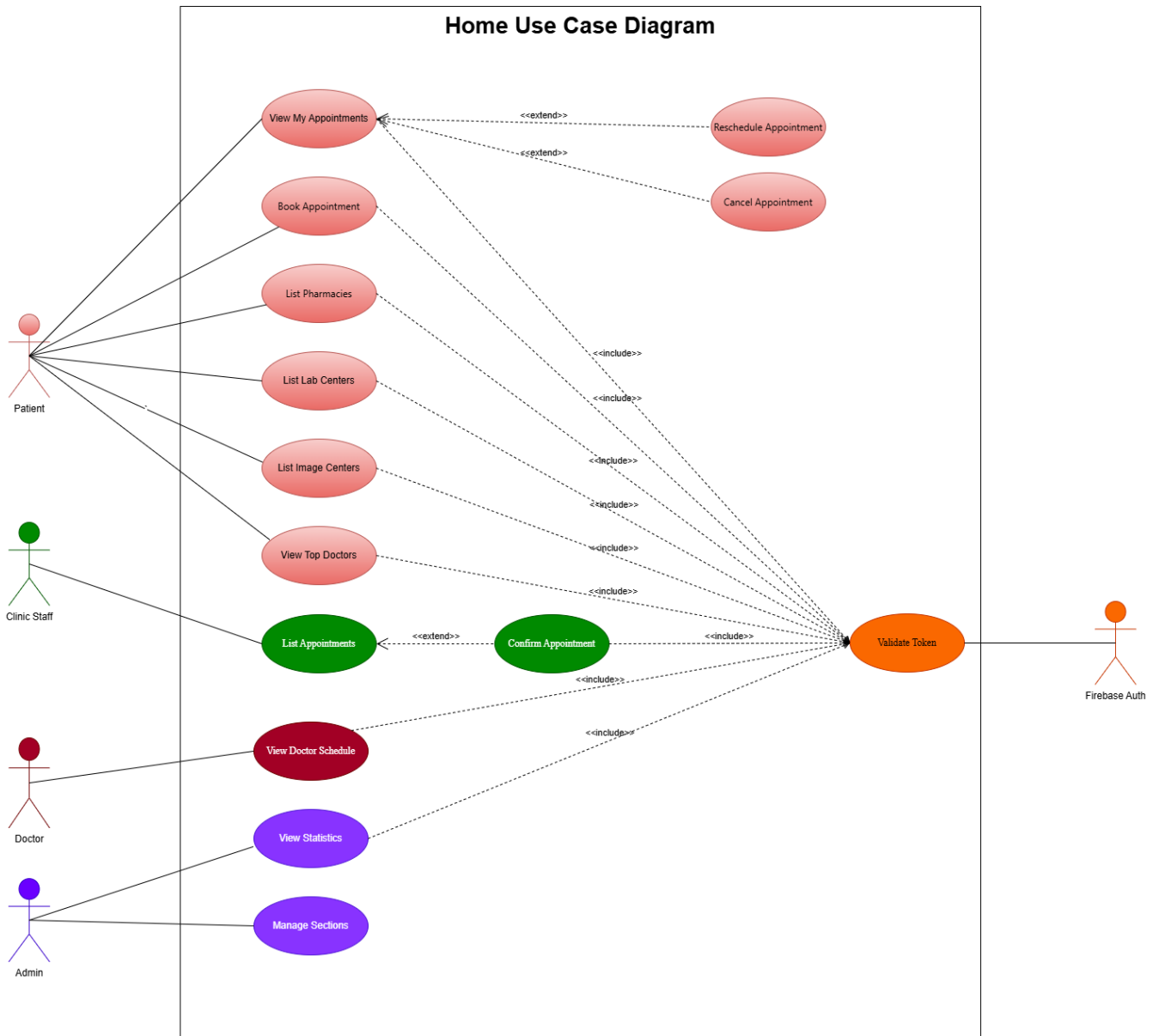


FIGURE 2.9.2.1 HOME USE CASE DIAGRAM

2.9.3. CONSULTATION USE CASE DIAGRAM

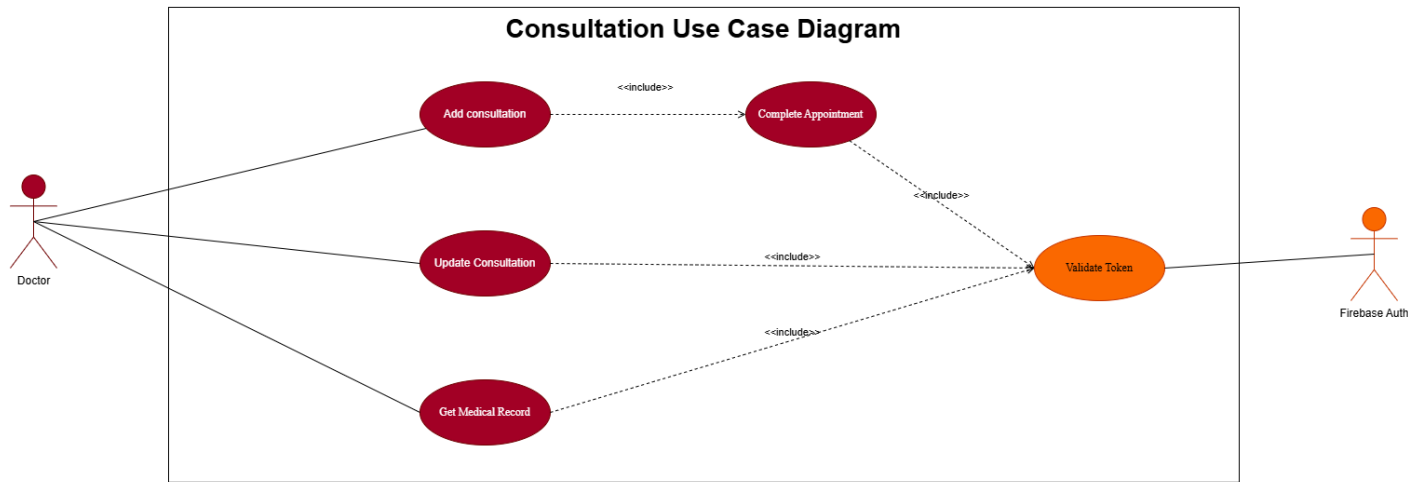


FIGURE 2.9.3.1 CONSULTATION USE CASE DIAGRAM

2.9.4. ADMINISTRATION USE CASE DIAGRAM

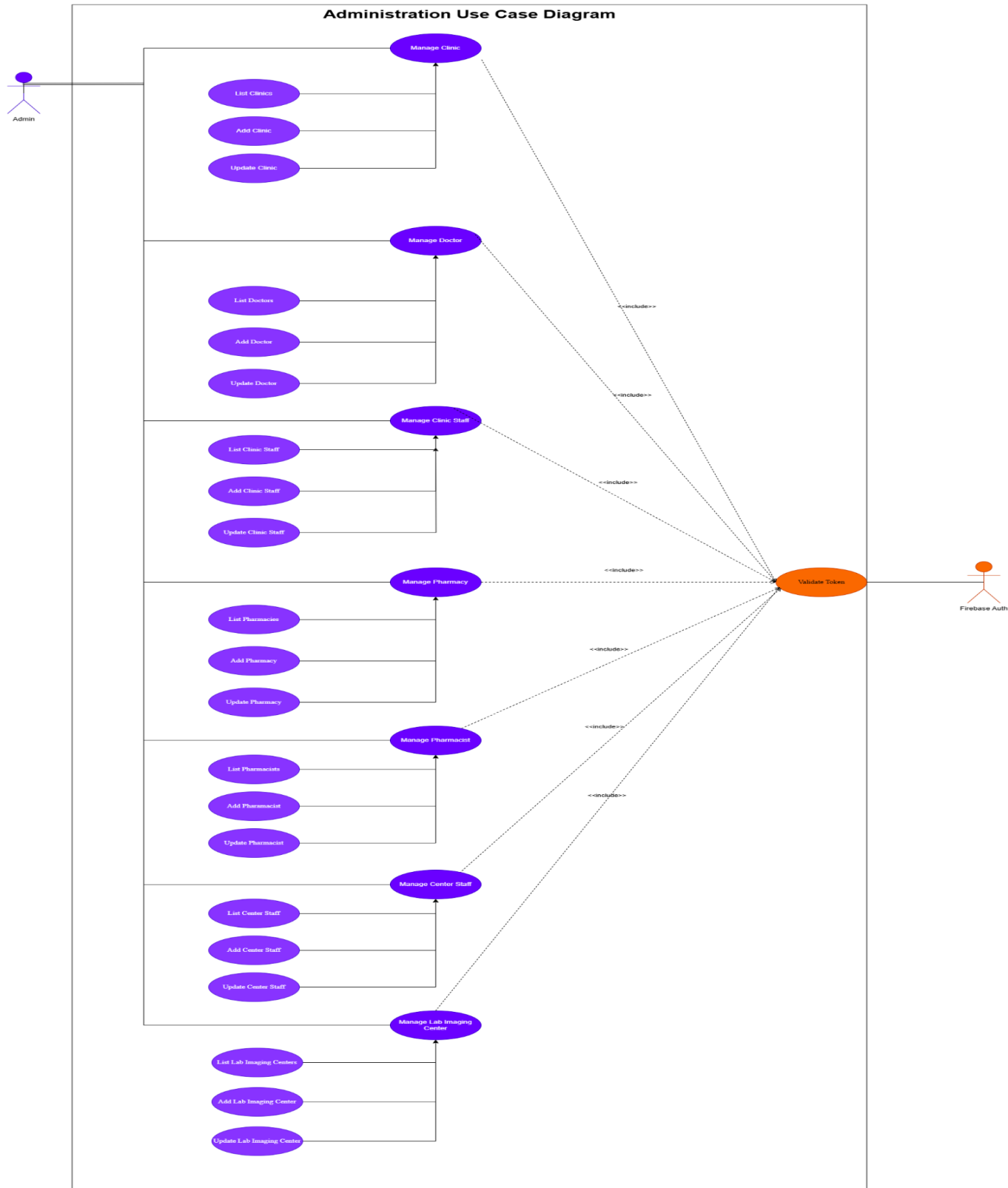


FIGURE 2.9.4.1 ADMINISTRATION USE CASE DIAGRAM

2.10. ACTOR DESCRIPTION

TABLE 2.10.1 ACTOR DESCRIPTION

Actor	Actor's Goal	Use case name
Patient	Register a new account	Register Patient Account – (UC–1)
All Users	Log in using email and password	User Login Using Email/Password – (UC–2)
All Users	Link Google account	Link Google Account – (UC–3)
All Users	Log in using Google account	Login Using Linked Google Account – (UC–4)
All Users	Cache clinic list after login	Cache Clinics List – (UC–5)
All Users	Cache doctor list after login	Cache Doctors List – (UC–6)
All Users	Cache imaging center list after login	Cache Imaging Centers List – (UC–7)
Patient	Book an appointment with a doctor	Book Appointment – (UC–8)
Patient	Reschedule an existing appointment	Reschedule Appointment – (UC–9)
Patient	Cancel a scheduled appointment	Cancel Appointment – (UC–10)
Patient	View top-rated doctors	View List of Top Doctors – (UC–11)
Patient	View contracted pharmacies	View Contracted Pharmacies – (UC–12)
Patient	View contracted lab centers	View Contracted Lab Centers – (UC–13)
Patient	View contracted imaging centers	View Contracted Imaging Centers – (UC–14)
Clinical Staff	View pending and confirmed appointments	Clinical Staff View Appointments – (UC–15)
Clinical Staff	Confirm pending appointments	Confirm Pending Appointments – (UC–16)
Doctor	View their appointment schedule	View Doctor Appointment Schedule – (UC–17)
Doctor	Update their service price	Doctor Update Service Price – (UC–18)
Doctor	Add consultation and complete appointment	Add Consultation – (UC–19)
Admin	View list of clinics	View Clinics List – (UC–20)
Admin	Add new clinic	Add New Clinic – (UC–21)
Admin	Update clinic details	Update Existing Clinic – (UC–22)
Admin	View list of doctors	View Doctors List – (UC–23)
Admin	Add new doctor	Add New Doctor – (UC–24)
Admin	Update doctor details	Update Existing Doctor – (UC–25)
Admin	View clinic staff list	View Clinic Staff List – (UC–26)
Admin	Add new clinic staff member	Add New Clinic Staff Member – (UC–27)
Admin	Update clinic staff details	Update Existing Clinic Staff Member – (UC–28)
Admin	View list of pharmacies	View Pharmacy List – (UC–29)
Admin	Add new pharmacy	Add New Pharmacy – (UC–30)
Admin	Update pharmacy details	Update Existing Pharmacy – (UC–31)
Admin	View pharmacists list	View Pharmacists List – (UC–32)
Admin	Add new pharmacist	Add New Pharmacist – (UC–33)
Admin	Update pharmacist details	Update Existing Pharmacist – (UC–34)
Admin	View lab/imaging centers list	View Lab/Imaging Centers List – (UC–35)
Admin	Add new lab/imaging center	Add New Lab/Imaging Center – (UC–36)
Admin	Update lab/imaging center details	Update Existing Lab/Imaging Center – (UC–37)
Admin	View lab/imaging center staff list	View Center Staff List – (UC–38)
Admin	Add new lab/imaging center staff member	Add New Center Staff Member – (UC–39)
Admin	Update lab/imaging center staff details	Update Existing Center Staff Member – (UC–40)

2.11. USE CASE DESCRIPTION

TABLE 2.11.1 (UC-1) REGISTER PATIENT ACCOUNT

Use Case 1 – (UC – 1) Register Patient Account	
Related Requirements	REQ1
Initiating Actor:	Patient
Actor's Goal:	To create a new patient account in the ICare system.
Participating Actors:	Firebase Authentication
Preconditions:	<ul style="list-style-type: none"> • The system displays the registration form to the patient. • The patient is not already registered in the system.
Postconditions:	The system creates a new patient account and redirects the user to the login screen.
Flow of Events for Main Success Scenario:	
→	1. The patient opens the registration form.
→	2. The patient provides the necessary personal information (i.e., name, email address, password, telephone number).
→	3. Patient submits the registration form.
←	4. System checks the entered details (e.g., email format validation, password strength, email uniqueness).
←	5. The system generates a new patient account and saves input.
←	6. The system confirms the successful registration to the Patient.
Flow of Events for Extensions (Alternate Scenarios):	
2a. Invalid Input:	
→	1. The patient provides inaccurate information in one or more areas.
←	2. The system identifies the invalid field(s) and displays appropriate error messages to the Patient.
→	3. The patient amends the incorrect information and re-submits the form (back to step 3).
4a. Email Already Exists:	
→	1. The patient gives an email address which is already being used.
←	2. The system checks for duplicate email and displays an error message.
→	3. The patient can select a different email or attempt to log in.
4b. Password Complexity Requirements Not Met:	
→	1. The patient logs in with a password that is not requirements compliant.
←	2. The system shows a message containing the password requirements.
→	3. The patient types a new password that satisfies the requirements (back to step 3).
4c. Network Error (if registration entails external service):	
←	1. There is an error in communication.
←	2. An error message is shown by the system.

TABLE 2.11.2 (UC-2) USER LOGIN USING EMAIL/PASSWORD



Use Case 2– (UC – 2) User Login Using Email/Password	
Related Requirements	REQ2
Initiating Actor:	All Users (Admin, Clinical Staff, Doctor, Patient)
Actor's Goal:	To log in using registered email and password.
Participating Actors:	Firebase Authentication
Preconditions:	<ul style="list-style-type: none"> • Users must have registered accounts. • Internet connectivity must be available.
Postconditions:	<ul style="list-style-type: none"> • Users gain access to their respective dashboard.
Flow of Events for Main Success Scenario:	
	<ol style="list-style-type: none"> 1. The user starts the login page. 2. The user supplies email and password credentials. 3. The system authenticates credentials using Firebase Authentication. 4. The user logs in and goes to his/her dashboard.
Flow of Events for Extensions (Alternate Scenarios):	
	<p>2a. Credentials are invalid: notifies the user and invites re-entry.</p> <p>3a. Login unsuccessful because of network problems: The system notifies the user to try later.</p>

TABLE 2.11.3 (UC-3) LINK GOOGLE ACCOUNT

Use Case 3– (UC – 3) Link Google Account	
Related Requirements	REQ3
Initiating Actor:	All Users (Admin, Clinical Staff, Doctor, Patient)
Actor's Goal:	To link a Google account to their existing ICare account.
Participating Actors:	Firebase Authentication
Preconditions:	<ul style="list-style-type: none"> The user must already have an existing ICare account. Internet connectivity must be available.
Postconditions:	<ul style="list-style-type: none"> The user's Google account is linked to their ICare account for future authentication.
Flow of Events for Main Success Scenario:	
	<ol style="list-style-type: none"> The user logs into the ICare application using their email address and password. The user navigates to the profile screen and chooses the "Link Google Account" option. A pop-up with a list of Google accounts appears, allowing you to select one. The system confirms that the Google account was successfully linked to the ICare profile.
Flow of Events for Extensions (Alternate Scenarios):	
	<p>3a. Google Authentication fails: System displays error message.</p> <p>3b. User cancels the Google linking process: The system retains the original authentication method and displays an error message.</p> <p>4a. The linking process encounters a network issue. The system suggests retrying once connectivity is restored.</p>

TABLE 2.11.4 (UC-4) LOGIN USING LINKED GOOGLE ACCOUNT

Use Case 4– (UC – 4) Login Using Linked Google Account	
Related Requirements	REQ4
Initiating Actor:	All Users (Admin, Clinic Staff, Doctor, Patient)
Actor's Goal:	To log into the ICare application using a pre-linked Google account
Participating Actors:	Firebase Authentication.
Preconditions:	<ul style="list-style-type: none"> The user's Google account must already be linked to their ICare account. Internet connectivity must be available.
Postconditions:	<ul style="list-style-type: none"> The user gains access to their respective dashboards upon successful login.
Flow of Events for Main Success Scenario:	
	<ol style="list-style-type: none"> The user opens the login page in the ICare app. The user selects Google Login. The system shows pop-up with google accounts list to choose from. The system verifies Google account connection using Firebase Authentication. The system grants the user access to dashboard.
Flow of Events for Extensions (Alternate Scenarios):	
	<p>4a. Google's account is not connected in the ICare system: <ul style="list-style-type: none"> The system displays a notification that the Google account is not linked with any of the existing ICare user accounts. </p> <p>5a. Network issues while authenticating: <ul style="list-style-type: none"> System notifies to attempt again once network becomes available. </p>

TABLE 2.11.5 (UC-5) CACHE CLINICS LIST



Use Case 5– (UC – 5) Cache Clinics List	
Related Requirements	REQ5
Initiating Actor:	All Users (Admin, Clinic Staff, Doctor, Patient)
Actor's Goal:	To fetch and cache the clinics list for offline access.
Participating Actors:	Firebase Authentication.
Preconditions:	<ul style="list-style-type: none"> The user must have successfully logged into the application. Internet connectivity is available to fetch data from the server.
Postconditions:	<ul style="list-style-type: none"> Clinics data is cached locally on the user's device.
Flow of Events for Main Success Scenario:	
	<ol style="list-style-type: none"> 1. User logs into the application successfully. 2. System requests clinics data from backend server. 3. Backend server responds with clinics list. 4. System saves the clinic's data into the local SQLite database. 5. User can view the clinics list, even offline.
Flow of Events for Extensions (Alternate Scenarios):	
	<p>2a. Server connection fails: System displays an error message suggesting pending clinics data sync.</p> <p>3a. Clinics data not available: System suggests checking connection or contacting later.</p>

TABLE 2.11.6 (UC-6) CACHE DOCTORS LIST



Use Case 6– (UC – 6) Cache Doctors List	
Related Requirements	REQ6
Initiating Actor:	All Users (Admin, Clinic Staff, Doctor, Patient)
Actor's Goal:	To fetch and cache the doctors list for offline access.
Participating Actors:	Firebase Authentication.
Preconditions:	<ul style="list-style-type: none"> The user must have successfully logged into the application. Internet connectivity is available to fetch data from the server.
Postconditions:	<ul style="list-style-type: none"> Doctors data is cached locally on the user's device.
Flow of Events for Main Success Scenario:	
	<ol style="list-style-type: none"> 1. User logs into the application. 2. System retrieves doctors' data from the backend server. 3. Backend server responds with doctors list. 4. The system caches the doctors list in the local SQLite database. 5. User has access to doctors list offline.
Flow of Events for Extensions (Alternate Scenarios):	
	<p>2a. Failures in request: • The system notifies user about doctors' data unavailability.</p> <p>3a. Error retrieving doctors list: • The system suggests retrying once connectivity stabilizes.</p>

TABLE 2.11.7 (UC-7) CACHE IMAGING CENTERS LIST








Use Case 7– (UC – 7) Cache Imaging Centers List	
Related Requirements	REQ7
Initiating Actor:	All Users (Admin, Clinic Staff, Doctor, Patient)
Actor's Goal:	To fetch and cache the imaging centers list for offline access.
Participating Actors:	Firebase Authentication.
Preconditions:	<ul style="list-style-type: none"> The user must have successfully logged into the application. Internet connectivity is available to fetch data from the server.
Postconditions:	<ul style="list-style-type: none"> Imaging centers data is cached locally on the user's device.
Flow of Events for Main Success Scenario:	
    	<ol style="list-style-type: none"> 1. User logs into the application. 2. System fetches list of imaging centers from backend server. 3. Backend server responds with required data. 4. System caches the imaging centers data locally. 5. User can view data in offline mode.
Flow of Events for Extensions (Alternate Scenarios):	
 	<p>2a. Unable to establish server connection: System notifies user of temporarily unavailable imaging data.</p> <p>3a. Error during retrieval: System suggests trying later after checking the network status.</p>

TABLE 2.11.8 (UC-8) BOOK APPOINTMENT








Use Case 8– (UC – 8) Book Appointment	
Related Requirements	REQ8
Initiating Actor:	Patient
Actor's Goal:	To book an appointment with a doctor.
Participating Actors:	Firebase Authentication.
Preconditions:	<ul style="list-style-type: none"> The patient is logged into the application. The patient must select a doctor and available time slot.
Postconditions:	<ul style="list-style-type: none"> An appointment is successfully booked.
Flow of Events for Main Success Scenario:	
    	<ol style="list-style-type: none"> 1. The patient gets logged into the ICare application. 2. The patient goes into the appointment booking module. 3. The patient selects a doctor and a slot of his choice. 4. The system verifies the availability of the selected slot with the backend server. 5. The system books the appointment and sends confirmation to the patient.
Flow of Events for Extensions (Alternate Scenarios):	
 	<p>3a. Time slot not available: • The system offers alternative slots for scheduling.</p> <p>4a. Booking fails due to server error: • The system suggests retrying or check connection.</p>

TABLE 2.11.9 (UC-9) RESCHEDULE APPOINTMENT

Use Case 9– (UC – 9) Reschedule Appointment	
Related Requirements	REQ9
Initiating Actor:	Patient
Actor's Goal:	To reschedule an existing appointment.
Participating Actors:	Firebase Authentication.
Preconditions:	<ul style="list-style-type: none"> The patient must have an existing appointment to reschedule. Patient must be logged in.
Postconditions:	<ul style="list-style-type: none"> Appointment is successfully rescheduled.
Flow of Events for Main Success Scenario:	
→	1. The patient logs in to the app.
→	2. The patient navigates to the appointment management module.
→	3. The patient selects an appointment to reschedule.
←	4. The system displays available slots.
→	5. The patient selects a new slot.
→	6. The system verifies the new slot availability and updates the appointment.
→	7. The patient receives rescheduling confirmation.
Flow of Events for Extensions (Alternate Scenarios):	
←	4a. No slots available: The system prompts for a retry option for rescheduling.
←	6a. Network problem during confirmation: The system displays error message recommending retry or calling support.

TABLE 2.11.10 (UC-10) CANCEL APPOINTMENT

Use Case 10– (UC – 10) Cancel Appointment	
Related Requirements	REQ10
Initiating Actor:	Patient
Actor's Goal:	To cancel an existing appointment.
Participating Actors:	Firebase Authentication.
Preconditions:	<ul style="list-style-type: none"> The patient is logged into the ICare application. The patient has an existing appointment to cancel.
Postconditions:	<ul style="list-style-type: none"> The appointment is successfully canceled.
Flow of Events for Main Success Scenario:	
→	1. The patient logs in on the app.
→	2. The patient proceeds to the page of scheduled appointments.
→	3. The patient selects the appointment he wishes to cancel.
←	4. The system signals cancellation to the backend server.
←	5. The system informs that the appointment is canceled.
Flow of Events for Extensions (Alternate Scenarios):	
←	4a. Network failure during cancellation: The system asks to retry once the connection is back.

TABLE 2.11.11 (UC-11) VIEW LIST OF TOP DOCTORS

Use Case 11– (UC – 11) View List of Top Doctors	
Related Requirements	REQ11
Initiating Actor:	Patient
Actor's Goal:	To view a list of top-rated doctors.
Participating Actors:	Firebase Authentication.
Preconditions:	<ul style="list-style-type: none"> The patient is logged into the application. Access to data is available.
Postconditions:	<ul style="list-style-type: none"> Patient views list of top-rated doctors
Flow of Events for Main Success Scenario:	
→	1. The patient logs in to the ICare application.
→	2. The patient opens the "Top Doctors" page.
←	3. The system retrieves top doctors list from the server.
←	4. The system displays the top doctors list for patient viewing.
Flow of Events for Extensions (Alternate Scenarios):	
←	3a. Top doctors' data not available: <ul style="list-style-type: none"> The system requests retry or check connectivity.

TABLE 2.11.12 (UC-12) VIEW CONTRACTED PHARMACIES

Use Case 12– (UC – 12) View Contracted Pharmacies	
Related Requirements	REQ12
Initiating Actor:	Patient
Actor's Goal:	To view a list of contracted pharmacies.
Participating Actors:	Firebase Authentication.
Preconditions:	<ul style="list-style-type: none"> Patient must be logged into the application. Connectivity is needed to access data from the server.
Postconditions:	<ul style="list-style-type: none"> Patient views list of contracted pharmacies.
Flow of Events for Main Success Scenario:	
→	1. The patient opens the application.
→	2. The patient navigates to the pharmacies.
←	3. The system retrieves contracted pharmacy data from the server.
←	4. The system presents the list to the patient.
Flow of Events for Extensions (Alternate Scenarios):	
←	3a. Server response fails: <ul style="list-style-type: none"> The system requests retry or checks network connection.

TABLE 2.11.13 (UC-13) VIEW CONTRACTED LAB CENTERS



Use Case 13– (UC – 13) View Contracted Lab Centers	
Related Requirements	REQ13
Initiating Actor:	Patient
Actor's Goal:	To view a list of contracted lab centers.
Participating Actors:	Firebase Authentication.
Preconditions:	<ul style="list-style-type: none"> • Patient must be logged into the application. • Connectivity to the backend server is essential.
Postconditions:	<ul style="list-style-type: none"> • Patient views contracted lab centers data.
Flow of Events for Main Success Scenario:	
 <ol style="list-style-type: none"> 1. The patient accesses the application. 2. The patient selects the option for lab centers. 3. The system fetches the contracted list of lab centers from the server. 4. The system displays the list to the patient for viewing. 	
Flow of Events for Extensions (Alternate Scenarios):	
 <ol style="list-style-type: none"> 3a. Failed server retrieval: <ul style="list-style-type: none"> • The system suggests retrying or checks network connection. 	

TABLE 2.11.14 (UC-14) VIEW CONTRACTED IMAGING CENTERS



Use Case 14– (UC – 14) View Contracted Imaging Centers	
Related Requirements	REQ14
Initiating Actor:	Patient
Actor's Goal:	To view a list of contracted lab centers.
Participating Actors:	Firebase Authentication.
Preconditions:	<ul style="list-style-type: none"> • Patient must be logged into the application. • Data access via server is required.
Postconditions:	<ul style="list-style-type: none"> • Patient views contracted imaging centers information.
Flow of Events for Main Success Scenario:	
 <ol style="list-style-type: none"> 1. The patient logs into the app. 2. The patient navigates to the imaging centers page. 3. The system retrieves imaging center records from the server. 4. The system displays the list to the patient. 	
Flow of Events for Extensions (Alternate Scenarios):	
 <ol style="list-style-type: none"> 3a. Failed retrieve from server: The system prompts retry or check network connectivity 	

TABLE 2.11.15 (UC-15) CLINICAL STAFF VIEW APPOINTMENTS

Use Case 15– (UC – 15) Clinical Staff View Appointments	
Related Requirements	REQ15
Initiating Actor:	Clinical Staff
Actor's Goal:	To view a list of contracted lab centers.
Participating Actors:	Firebase Authentication.
Preconditions:	<ul style="list-style-type: none"> Clinical staff must be logged into the application.
Postconditions:	<ul style="list-style-type: none"> Clinical staff views all relevant appointment lists.
Flow of Events for Main Success Scenario:	
→	1. Clinical staff logs into the application.
→	2. The staff visit the appointments section.
←	3. The system fetches the list of pending and confirmed appointments from server.
←	4. System displays an appointments list for staff to see.
Flow of Events for Extensions (Alternate Scenarios):	
←	3a. Failed retrieval from server: The system attempts to retrieve or verify network connection.

TABLE 2.11.16 (UC-16) CONFIRM PENDING APPOINTMENTS

Use Case 16– (UC – 16) Confirm Pending Appointments	
Related Requirements	REQ16
Initiating Actor:	Clinical Staff
Actor's Goal:	To confirm pending patient appointments.
Participating Actors:	Firebase Authentication.
Preconditions:	<ul style="list-style-type: none"> Clinical staff is logged in to the application. Pending appointments list is displayed..
Postconditions:	<ul style="list-style-type: none"> Appointment status is updated to confirmed.
Flow of Events for Main Success Scenario:	
→	1. The clinical staff member logs in the application.
→	2. The clinical staff member views pending appointments list.
→	3. The clinical staff member selects an appointment to confirm.
←	4. System updates appointment status in the server.
←	5. System presents confirmation message to the clinical staff member.
Flow of Events for Extensions (Alternate Scenarios):	
←	4a. Network error during update: <ul style="list-style-type: none"> System asks for retry confirmation.

TABLE 2.11.17 (UC-17) VIEW DOCTOR APPOINTMENT SCHEDULE



Use Case 17– (UC – 17) View Doctor Appointment Schedule	
Related Requirements	REQ17
Initiating Actor:	Doctor
Actor's Goal:	To view their appointment schedule, including pending and confirmed appointments.
Participating Actors:	Firebase Authentication.
Preconditions:	<ul style="list-style-type: none"> • Doctor must be logged into the application.
Postconditions:	<ul style="list-style-type: none"> • Appointment status is updated to confirmed.
Flow of Events for Main Success Scenario:	
	<ol style="list-style-type: none"> 1. The doctor navigates to the appointment schedule view. 2. The system asks the backend server for the Doctor's appointment schedule. 3. The backend Server retrieves the appointment details of the current doctor, pending as well as confirmed appointments. 4. The backend Server sends the appointment schedule details back to the system. 5. The system receives the appointment schedule details. 6. The system displays the appointment schedule for the Doctor, indicating the status of each appointment.
Flow of Events for Extensions (Alternate Scenarios):	
	4a. Network error during update: <ul style="list-style-type: none"> • System suggests retrying confirmation.

TABLE 2.11.18 (UC-18) DOCTOR UPDATE SERVICE PRICE



Use Case 18– (UC – 18) Doctor Update Service Price	
Related Requirements	REQ18
Initiating Actor:	Doctor
Actor's Goal:	To update their service price.
Participating Actors:	Firebase Authentication.
Preconditions:	<ul style="list-style-type: none"> • Doctor must be logged into the application.
Postconditions:	<ul style="list-style-type: none"> • Service price is updated on the server.
Flow of Events for Main Success Scenario:	
	<ol style="list-style-type: none"> 1. The physician logs in to the application. 2. The physician clicks on edit service price icon. 3. The system proceeds to edit physician screen. 3. The physician inputs a new service price. 4. The system updates server-side price data. 5. The system checks for price update validity.
Flow of Events for Extensions (Alternate Scenarios):	
	4a. Server update error: <ul style="list-style-type: none"> • The system prompts retry or support contact.

TABLE 2.11.19 (UC-19) ADD CONSULTATION

Use Case 19– (UC – 19) Add Consultation	
Related Requirements	REQ19
Initiating Actor:	Doctor
Actor's Goal:	To add a consultation to a selected appointment.
Participating Actors:	Firebase Authentication.
Preconditions:	<ul style="list-style-type: none"> • Doctor must be logged into the application. • Access to scheduled appointments
Postconditions:	<ul style="list-style-type: none"> • Consultation is recorded, and appointment status is updated.
Flow of Events for Main Success Scenario:	
	<ol style="list-style-type: none"> 1. The doctor logs in to the program. 2. The doctor selects an appointment to consult. 3. The doctor inputs consultation details. 4. The system updates consultation details and sets appointment as done on the server. 5. The doctor is assured of update.
Flow of Events for Extensions (Alternate Scenarios):	
	<p>4a. Unable to update on the server:</p> <ul style="list-style-type: none"> • The system suggests retrying or contact support.

TABLE 2.11.20 (UC-20) VIEW CLINICS LIST

Use Case 20 – (UC – 20) View Clinics List	
Related Requirements	REQ20
Initiating Actor:	Admin
Actor's Goal:	To see a list of all clinics registered in the system.
Participating Actors:	Firebase Authentication
Preconditions:	<ul style="list-style-type: none"> • The admin is logged into the system with administrative privileges. • The admin navigates to the clinic management section.
Postconditions:	<ul style="list-style-type: none"> • The system displays a list of all clinics, including relevant details.
Flow of Events for Main Success Scenario:	
	<ol style="list-style-type: none"> 1. The admin loads the clinic management section. 2. The system asks the backend server for the list of clinics. 3. The backend Server retrieves the clinic data from the database. 4. The backend Server sends the list of clinics to the system. 5. The system retrieves the list of clinics. 6. The system displays the list of clinics to the admin.
Flow of Events for Extensions (Alternate Scenarios):	
	<p>2a. Backend Server Not Available:</p> <ol style="list-style-type: none"> 1. Backend server is not reachable. 2. System displays an error message to Admin. <p>3a. No Clinics Found:</p> <ol style="list-style-type: none"> 1. Backend Server retrieves no data for the clinics. 2. System displays a message that there are no clinics registered yet. <p>4a. Network Error During Fetch:</p> <ol style="list-style-type: none"> 1. Network error occurs while retrieving the list of clinics. 2. System displays an error message to Admin. <p>6a. Display Error:</p> <ol style="list-style-type: none"> 1. A fault occurs when the system is attempting to display the list. 2. The system displays an error message to the Admin.

TABLE 2.11.21 (UC-21) ADD NEW CLINIC

Use Case 21 – (UC – 21) Add New Clinic	
Related Requirements	REQ21
Initiating Actor:	Admin
Actor's Goal:	To add a new clinic to the system.
Participating Actors:	Firebase Authentication
Preconditions:	<ul style="list-style-type: none"> The admin is logged into the system with administrative privileges. The admin navigates to the clinic management section.
Postconditions:	<ul style="list-style-type: none"> The new clinic's information is stored in the database. The system confirms the successful addition to the Admin. The new clinic appears in the clinic list.
Flow of Events for Main Success Scenario:	
→	1. The admin navigates to the clinic management section.
→	2. The admin initiates the process of adding a new clinic.
←	3. The system displays a blank form for entering the clinic's details.
→	4. The admin enters the required information.
→	5. The admin submits the form.
←	6. The system validates the entered information (e.g., checks for valid formats, mandatory fields).
←	7. The system updates the new clinic data to the backend server to create a new record.
←	8. The backend server adds the new clinic information to the database.
←	9. The backend server sends a confirmation response to the system.
←	10. The system sends a successful message to the admin.
Flow of Events for Extensions (Alternate Scenarios):	
	4a. Invalid Input:
→	1. The admin enters invalid data in one or more fields.
←	2. The system identifies the erroneous field(s) and displays appropriate error messages.
→	3. The admin enters the erroneous data and submits the form once more (to step 5 of main success scenario).
	6a. Server Validation Error:
←	1. The system sends the data to the server, while server-side validation will fail.
←	2. The server returns a server error message with details of the validation failures.
←	3. The system displays these error messages to the admin.
→	4. Admin edits the data from error messages and resubmits (back to step 5).
	7a. Backend Server Unavailable:
←	1. The backend server is unavailable to the system.
←	2. The system displays an error message to the admin.
	7b. Database Error:
←	1. Error occurs while inserting the clinic information into the database.
←	2. Backend server returns an error response.
←	3. The system displays an error message to the admin.

TABLE 2.11.22 (UC-22) UPDATE EXISTING CLINIC

Use Case 22 – (UC – 22) Update Existing Clinic	
Related Requirements	REQ22
Initiating Actor:	Admin
Actor's Goal:	To modify the details of an existing clinic in the system.
Participating Actors:	Firebase Authentication
Preconditions:	<ul style="list-style-type: none"> • The admin is logged into the system with administrative privileges. • The system displays a list of existing clinics. • The admin has selected a clinic to update. • The system displays the current details of the selected clinic in an editable form.
Postconditions:	<ul style="list-style-type: none"> • The details of the selected clinic are updated in the system's database. • The system confirms the successful update to the admin. • The updated clinic information is reflected in the clinics list (potentially requiring a refresh or automatic update).
Flow of Events for Main Success Scenario:	
→	1. The admin navigates to the clinic management page.
→	2. The admin selects a clinic from the list to edit.
←	3. The system displays the editable form with the current clinic data.
→	4. The admin fills in the edit fields.
→	5. The admin submits the edited form.
←	6. The system validates the updated data (e.g., valid formats, required fields).
→	7. The system sends the updated data to the backend server to update the clinic record.
←	8. The backend server updates the clinic information in the database.
←	9. The backend server sends a confirmation response to the system.
←	10. The system displays a success message to the admin.
Flow of Events for Extensions (Alternate Scenarios):	
→	4a. Invalid Input:
→	1. The admin enters invalid data in one or more fields.
←	2. The system indicates the invalid field(s) and displays appropriate error messages.
→	3. The admin confirms the invalid data and resubmits the form.
←	6a. Validation Error on Server:
←	1. The system submits the data to the server, but the server-side validation fails.
←	2. The server sends an error message containing details of the validation failures.
←	3. The system displays these error messages to the admin.
→	4. Admin corrects the data from the error messages and submits them again.
←	7a. Backend Server Unavailable:
←	1. The backend server is not accessible by the system.
←	2. The system shows an error message to the admin.
←	7b. Database Error:
←	1. There is a failure in updating the clinic details in the database.
←	2. The backend server returns an error.
←	3. The system shows an error message to the admin.

TABLE 2.11.23 (UC-23) VIEW DOCTORS LIST

Use Case 23 – (UC – 23) View Doctors List	
Related Requirements	REQ23
Initiating Actor:	Admin
Actor's Goal:	To see a list of all doctors registered in the system.
Participating Actors:	Firebase Authentication
Preconditions:	<ul style="list-style-type: none"> The admin is logged into the system with administrative privileges. The admin navigates to the doctor management section.
Postconditions:	<ul style="list-style-type: none"> The system displays a list of all doctors, including relevant details.
Flow of Events for Main Success Scenario:	
→	1. The admin is on the doctor management page.
→	2. The system asks for the list of doctors from the backend server.
→	3. The backend server retrieves the doctor's data from the database.
←	4. The backend server sends the list of doctors back to the system.
←	5. The system receives the list of doctors.
←	6. The system displays the list of doctors to the admin.
Flow of Events for Extensions (Alternate Scenarios):	
←	2a. Backend Server Unavailable:
←	1. The backend server is unreachable.
←	2. An error message to the admin is displayed.
←	3a. No Staff Found:
←	1. Doctors' information is requested by the backend server, but nothing is received.
←	2. A message is displayed indicating that no doctors are found that are linked to clinics.
←	4a. Network Error While Fetching:
←	1. A network error while fetching the list of doctors.
←	2. An error message is displayed to the admin.
←	6a. Display Error:
←	1. There is a fault when the system attempts to display the list.
←	2. The system displays a default error message to the admin.

TABLE 2.11.24 (UC-24) ADD NEW DOCTOR

Use Case 24 – (UC – 24) Add New Doctor	
Related Requirements	REQ24
Initiating Actor:	Admin
Actor's Goal:	To register a new doctor associated with a clinic.
Participating Actors:	Firebase Authentication
Preconditions:	<ul style="list-style-type: none"> The admin is logged into the system with administrative privileges. The admin navigates to the doctor management section. The system displays a form for adding a new doctor.
Postconditions:	<ul style="list-style-type: none"> The new doctor's information is stored in the system's database. The system confirms the successful addition to the admin. The new doctor appears in the doctors list (potentially requiring a refresh or automatic update).
Flow of Events for Main Success Scenario:	
→	1. The admin navigates the doctor's management page.
→	2. The admin initiates the process of adding a new doctor.
←	3. The system displays a blank form where the doctor's details need to be input.
→	4. The admin inputs the required details.
→	5. The admin submits the form.
←	6. The system validates the input provided (e.g., checks for valid format, mandatory fields).
←	7. The system sends the new doctor's information to the backend server to create a new record.
←	8. The backend server adds the new doctor's information to the database.

←	9. The backend server sends a success message back to the system.
←	10. The system displays a success message to the admin.
Flow of Events for Extensions (Alternate Scenarios):	
→	4a. Invalid Input:
→	1. The admin enters invalid information in one or more fields.
←	2. The system traps the invalid field(s) and provides correct error messages.
→	3. The admin corrects the invalid data and re-submits the form.
←	6a. Server-side Validation Error:
←	1. The system sends the data to the server, but server-side validation fails.
←	2. The server responds with an error message containing information about the validation failures.
←	3. The system displays these error messages to the admin.
→	4. The admin updates the data based on the error messages and resubmits.
←	7a. Backend Server Unavailable:
←	1. The backend server is inaccessible to the system.
←	2. The system displays an error message to the admin.
←	7b. Database Error:
←	1. There is a failure in adding the doctor's information into the database.
←	2. The backend server sends an error response.
←	3. The system displays an error message to the admin.

TABLE 2.11.25 (UC-25) UPDATE EXISTING DOCTOR

Use Case 25 – (UC – 25) Update Existing Doctor	
Related Requirements	REQ25
Initiating Actor:	Admin
Actor's Goal:	To modify the details of an existing doctor associated with a clinic.
Participating Actors:	Firebase Authentication
Preconditions:	<ul style="list-style-type: none"> The admin is logged into the system with administrative privileges. The system displays a list of existing doctors. The admin has selected a doctor to update. The system displays the current details of the selected doctor in an editable form.
Postconditions:	<ul style="list-style-type: none"> The details of the selected doctor are updated in the system's database. The system confirms the successful update to the admin. The updated doctor information is reflected in the doctors list (potentially requiring a refresh or automatic update).
Flow of Events for Main Success Scenario:	
→	1. Admin visits the doctor's management section.
→	2. Admin selects a doctor from the list to be edited.
←	3. System displays an editable form with existing doctor details.
→	4. Admin updates the necessary fields.
→	5. Admin saves the updated form.
→	6. System verifies the updated details (e.g., checks for correct formats, required fields).
←	7. The system invokes the backend server with new data to update the doctor's record.
←	8. The backend server updates the doctor's database.
←	9. The backend server sends a successful message to the system.
←	10. The system displays a success message to the admin.
Flow of Events for Extensions (Alternate Scenarios):	
→	4a. Invalid Input:
→	1. The admin provides invalid data in one or more fields.
←	2. The system identifies the invalid field(s) and shows appropriate error messages.
→	3. The admin corrects the invalid data and resubmits the form.
←	6a. Server-side Validation Error:
←	1. The system submits the data to the server, but the server-side validation has not succeeded.
←	2. The server gives an error message containing details of the failure of validations.
←	3. The system displays these error messages to the admin.
→	4. The admin then updates the details according to the error messages and resubmits.
←	7a. Backend Server Unavailable:
←	1. The backend server is unable to be accessed by the system.
←	2. The system displays an error message to the admin.
←	7b. Database Error:

←	1. There is a problem in updating the clinic staff member details in the database.
←	2. An error response is returned by the backend server.
←	3. An error message is displayed by the system to the admin.

TABLE 2.11.26 (UC-26) VIEW CLINIC STAFF LIST

Use Case 26 – (UC – 26) View Clinic Staff List	
Related Requirements	REQ26
Initiating Actor:	Admin
Actor's Goal:	To see a list of all staff members associated with clinics.
Participating Actors:	Firebase Authentication
Preconditions:	<ul style="list-style-type: none"> The admin is logged into the system with administrative privileges. The admin navigates to the clinic staff management section.
Postconditions:	<ul style="list-style-type: none"> The system displays a list of all clinic staff members, including relevant details.
Flow of Events for Main Success Scenario:	
→	1. The admin navigates to the management of clinic staff section.
→	2. The system prompts the list of clinic staff from the backend server.
→	3. The backend server retrieves the clinic staff details from the database.
←	4. The backend server provides the list of clinic staff to the system.
←	5. The system receives the list of clinic staff.
←	6. The system displays the list of clinic staff to the admin.
Flow of Events for Extensions (Alternate Scenarios):	
2a. Backend Server Unavailable:	
←	1. The backend server is inaccessible to the system.
←	2. The system displays an error message to the admin.
3a. No Staff Found:	
←	1. The backend server is not able to find any clinic staff information.
←	2. The system displays a message indicating that no staff are currently assigned to clinics.
4a. Network Error During Fetch:	
←	1. A network error is encountered while fetching the list of clinic staff.
←	2. The system displays an error message to the admin.
6a. Display Error:	
←	1. The system has an error while attempting to display the list.
←	2. The system displays a generic error message to the admin.

TABLE 2.11.27 (UC-27) ADD NEW CLINIC STAFF MEMBER

Use Case 27 – (UC – 27) Add New Clinic Staff Member	
Related Requirements	REQ27
Initiating Actor:	Admin
Actor's Goal:	To register a new staff member associated with a clinic.
Participating Actors:	Firebase Authentication
Preconditions:	<ul style="list-style-type: none"> The admin is logged into the system with administrative privileges. The admin navigates to the clinic staff management section. The system displays a form for adding a new staff member.
Postconditions:	<ul style="list-style-type: none"> The new clinic staff member's information is stored in the system's database. The system confirms the successful addition to the admin. The new staff member appears on the clinic staff list (potentially requiring a refresh or automatic update).
Flow of Events for Main Success Scenario:	
→	1. The admin accesses the clinic staff management area.
→	2. The admin begins the process of adding new staff.
←	3. The system displays a blank form to enter the details of the staff member.
→	4. The admin fills in required details.
→	5. The admin submits the form.

←	6. The system validates the entered details (e.g., valid formats, mandatory fields).
←	7. The system sends the new staff member data to the backend server to add a new record.
←	8. The backend server adds the new staff member details into the database.
←	9. Backend server provides a confirmatory response to the system.
←	10. The system displays a success message to the admin.
Flow of Events for Extensions (Alternate Scenarios):	
→	4a. Invalid Input:
←	1. The admin enters invalid data in one or more fields.
→	2. The system identifies the invalid field(s) and displays proper error messages.
→	3. The admin corrects the invalid information and submits the form again (back to step 5 of the primary success scenario).
←	6a. Server Validation Error:
←	1. The system passes the data onto the server, but server-side validation has failed.
←	2. The server error message includes details about the validation failures.
←	3. The system displays these error messages to the admin.
→	4. The admin corrects the data based on error messages and resubmits (return to step 5).
←	7a. Backend Server Down:
←	1. The system cannot reach the backend server.
←	2. The system displays an error message to the admin.
←	7b. Database Error:
←	1. An error occurs while adding the clinic staff member details to the database.
←	2. The backend server sends an error response.
←	3. The system displays an error message to the admin.

TABLE 2.11.28 (UC-28) UPDATE EXISTING CLINIC STAFF MEMBER

Use Case 28 – (UC – 28) Update Existing Clinic Staff Member	
Related Requirements	REQ28
Initiating Actor:	Admin
Actor's Goal:	To modify the details of an existing staff member associated with a clinic.
Participating Actors:	Firestore Authentication
Preconditions:	<ul style="list-style-type: none"> The admin is logged into the system with administrative privileges. The system displays a list of existing clinic staff members. The admin has selected a staff member to update. The system displays the current details of the selected staff member in an editable form.
Postconditions:	<ul style="list-style-type: none"> The details of the selected clinic staff member are updated in the system's database. The system confirms the successful update to The admin. The updated staff member information is reflected in the clinic staff list (potentially requiring a refresh or automatic update).
Flow of Events for Main Success Scenario:	
→	1. The admin navigates to the clinic staff management section.
→	2. The admin selects a staff member from the list to edit.
←	3. The system displays the editable form with the current staff member details.
→	4. The admin fills in the mandatory fields.
→	5. The admin submits the changed form.
←	6. The system verifies the updated information (e.g., correct formats, mandatory fields).
→	7. The system forwards the updated information to the backend server to update the staff member record.
←	8. The backend server changes the database with staff information.
←	9. The backend server returns a successful response to the system.
←	10. The system displays a success message to the admin.
Flow of Events for Extensions (Alternate Scenarios):	
→	4a. Invalid Input:
←	1. The admin enters incorrect data in one or more fields.
→	2. The system identifies the invalid field(s) and displays proper error messages.
→	3. Admin corrects the wrong data and resubmits the form.
←	6a. Server-side Validation Error:
←	1. System submits the data to the server, but server-side validation is unsuccessful.
←	2. The server sends an error response with details of the failed validations.

←	3. The system shows these error messages to the admin.
→	4. Admin corrects the data from error messages and resubmits.
	7a. Backend Server Not Available:
←	1. The backend server cannot be accessed by the system.
←	2. The system displays an error message to the admin.
	7b. Database Error:
←	1. There is a failure to update clinic staff member details in the database.
←	2. The backend server sends back an error response.
←	3. The system displays an error message to the admin.

TABLE 2.11.29 (UC-29) VIEW PHARMACY LIST

Use Case 29 – (UC – 29) View Pharmacy List	
Related Requirements	REQ29
Initiating Actor:	Admin
Actor's Goal:	To see a list of all pharmacies registered in the system.
Participating Actors:	Firebase Authentication
Preconditions:	<ul style="list-style-type: none"> The admin is logged into the system with administrative privileges . The admin navigates to the pharmacy management section.
Postconditions:	<ul style="list-style-type: none"> The system displays a list of all pharmacies, including relevant details (e.g., name, address, contact information).
Flow of Events for Main Success Scenario:	
→	1. The admin signs in the pharmacy management section.
→	2. The system asks the backend server for the latest list of pharmacies.
→	3. The backend server retrieves the data of pharmacies from the database.
←	4. The backend server forwards the list of pharmacies to the system.
←	5. The system gets the list of pharmacies.
←	6. The system shows the list of pharmacies to the admin.
Flow of Events for Extensions (Alternate Scenarios):	
	2a. Backend Server Not Available:
←	1. The system is unable to access the backend server.
←	2. The admin displays an error message by the system.
	3a. No Pharmacies Found:
←	1. The backend server retrieves no data regarding pharmacies.
←	2. The system displays that no pharmacies are registered currently.
	4a. Network Error During Fetch:
←	1. A network error occurs when fetching the list of pharmacists.
←	2. The system displays an error message to the admin.
	6a. Display Error:
←	1. The system error occurs while attempting to show the list.
←	2. The system displays a standard error message to the admin.

TABLE 2.11.30 (UC-30) ADD NEW PHARMACY

Use Case 30 – (UC – 30) Add New Pharmacy	
Related Requirements	REQ30
Initiating Actor:	Admin
Actor's Goal:	To register a new pharmacy in the system.
Participating Actors:	Firebase Authentication
Preconditions:	<ul style="list-style-type: none"> The admin is logged into the system with administrative privileges. The admin navigates to the pharmacy management section.




Postconditions:	<ul style="list-style-type: none"> • The system displays a form for adding a new pharmacy. • The new pharmacy's information is stored in the system's database. • The system confirms the successful addition to the admin. • The new pharmacy appears in the pharmacy list (potentially requiring a refresh or automatic update).
Flow of Events for Main Success Scenario:	
	<ol style="list-style-type: none"> 1. The admin navigates to the pharmacies management section. 2. The admin initiates adding a new one. 3. The system displays an empty form to fill in the pharmacy details (e.g., name, address, contact). 4. The admin fills in the required details. 5. The admin submits the form. 6. The system validates the entered data (e.g., correct formats, mandatory fields, unique name/identifier if applicable). 7. If the validation is successful, the system forwards the new pharmacy information to the backend server to save a new record. 8. The backend server saves the new pharmacy information into the database. 9. The backend server sends a confirmatory response to the system. 10. The system displays a success message to the admin.
Flow of Events for Extensions (Alternate Scenarios):	
	<p>4a. Invalid Input:</p> <ol style="list-style-type: none"> 1. The admin enters wrong data in one or more fields. 2. The system identifies the wrong field(s) and displays proper error messages. 3. The admin corrects the wrong data and re-submits the form (back to step 5 of the primary success scenario). <p>6a. Validation Error on Server:</p> <ol style="list-style-type: none"> 1. The system posts the data to the server, and server-side validation fails. 2. The server sends an error response with details of validation failures. 3. The system displays such error messages to the admin. 4. The admin corrects the data from the error messages and resubmits (to step 5). <p>7a. Backend Server Unavailable:</p> <ol style="list-style-type: none"> 1. The backend server is not accessible to the system. 2. The system displays an error message to the admin. <p>7b. Database Error:</p> <ol style="list-style-type: none"> 1. There is a failure when updating the pharmacy information in the database. 2. The backend server gives back an error response. 3. The system displays an error message to the admin.

TABLE 2.11.31 (UC-31) UPDATE EXISTING PHARMACY

Use Case 31 – (UC – 31) Update Existing Pharmacy	
Related Requirements	REQ31
Initiating Actor:	Admin
Actor's Goal:	To modify the details of an existing pharmacy in the system.
Participating Actors:	Firebase Authentication
Preconditions:	<ul style="list-style-type: none"> • The admin is logged into the system with administrative privileges . • The system displays a list of existing pharmacies (potentially through UC29). • The admin has selected a pharmacy to update. • The system displays the current details of the selected pharmacy in an editable form.
Postconditions:	<ul style="list-style-type: none"> • The details of the selected pharmacy are updated in the system's database. • The system confirms the successful update to The admin. • The updated pharmacy information is reflected in the pharmacy list (potentially requiring a refresh or automatic update).
Flow of Events for Main Success Scenario:	
	<ol style="list-style-type: none"> 1. The admin navigates to the pharmacy management section. 2. The admin selects a pharmacy from the list to update. 3. The system displays editable form with the current pharmacy details. 4. The admin updates the necessary fields (e.g., name, address, contact information). 5. The admin submits the updated form. 6. The system validates the updated data (e.g., correct formats, mandatory fields).







	7. If valid, the system passes the updated data to the backend server for updating the pharmacy record. 8. The backend server processes the data of the pharmacy in the database. 9. The backend server sends a successful response to the system. 10. The system displays a success message to the admin.
Flow of Events for Extensions (Alternate Scenarios):	
	4a. Invalid Input: 1. The admin enters incorrect data in one or more fields. 2. The system identifies the incorrect field(s) and displays appropriate error messages. 3. The admin corrects the invalid data and resubmits the form (back to step 5 of the standard success scenario).
	6a. Server Validation Error: 1. The system sends the data to the server, and the server-side validation fails. 2. The server sends back an error message with details about the validation failures. 3. The system displays error messages to the admin.
	4. Admin revises the data on error messages and resubmits (back to step 5).
	7a. Backend Server Unavailable: 1. The backend server could not be connected by the system. 2. An error message is displayed by the system to the admin.
	7b. Database Error: 1. There is an error in updating the pharmacy details in the database. 2. An error response is given by the backend server. 3. An error message is displayed by the system to the admin.

TABLE 2.11.32 (UC-32) VIEW PHARMACISTS LIST






Use Case 32 – (UC – 32) View Pharmacists List	
Related Requirements	REQ32
Initiating Actor:	Admin
Actor's Goal:	To see a list of all pharmacists registered in the system.
Participating Actors:	Firebase Authentication
Preconditions:	<ul style="list-style-type: none"> The admin is logged into the system with administrative privileges. The admin navigates to the pharmacist management section.
Postconditions:	The system displays a list of all pharmacists, including relevant details (e.g., name, contact information, associated pharmacy).
Flow of Events for Main Success Scenario:	
	1. The admin navigates to the pharmacist management section. 2. The system makes a request for the list of pharmacists to the backend server. 3. The backend server retrieves the pharmacist details from the database. 4. The backend server sends the list of pharmacists back to the system. 5. The system gets the list of pharmacists. 6. The system displays the list of pharmacists to the admin.
Flow of Events for Extensions (Alternate Scenarios):	
	2a. Backend Server Unavailable: 1. The backend server is unavailable to the system. 2. The system displays an error message to the admin.
	3a. No Pharmacists Found: 1. The backend server retrieves no pharmacist data. 2. The system displays a message indicating that no pharmacists are registered.
	4a. Network Error During Fetch: 1. There is a network error while fetching the list of pharmacists. 2. The system displays an error message to the admin.
	6a. Display Error: 1. There is an error when the system is attempting to display the list. 2. The system displays a generic error message to the admin.

TABLE 2.11.33 (UC-33) ADD NEW PHARMACIST

Use Case 33 – (UC – 33) Add New Pharmacist	
Related Requirements	REQ33
Initiating Actor:	Admin
Actor's Goal:	To register a new pharmacist in the system.
Participating Actors:	Firebase Authentication
Preconditions:	<ul style="list-style-type: none"> The admin is logged into the system with administrative privileges. The admin navigates to the pharmacist management section. The system displays a form for adding a new pharmacist.
Postconditions:	<ul style="list-style-type: none"> The new pharmacist's information is stored in the system's database. The system confirms the successful addition to the admin. The new pharmacist appears in the pharmacist list (potentially requiring a refresh or automatic update).
Flow of Events for Main Success Scenario:	
	<ol style="list-style-type: none"> 1. The admin navigates to the pharmacist management page. 2. The admin clicks to create a new pharmacist. 3. The system displays a blank form to enter the pharmacist's details (e.g., name, email, contact information, affiliated pharmacy). 4. The admin enters the required details. 5. The admin submits the form. 6. The system validates the entered details (e.g., correct formats, mandatory fields, unique email). 7. If validation is successful, the system sends the new pharmacist details to the backend server for the insertion of a new record. 8. The backend server adds the new pharmacist details into the database. 9. The backend server sends a confirmation reply to the system. 10. The system displays a success message to the admin.
Flow of Events for Extensions (Alternate Scenarios):	
	<p>4a. Invalid Input:</p> <ol style="list-style-type: none"> 1. The admin enters incorrect data in one or more fields. 2. The system identifies the invalid field(s) and displays appropriate error messages. 3. The admin corrects the invalid data and resubmits the form (back to step 5 of the main success scenario). <p>6a. Validation Error on Server:</p> <ol style="list-style-type: none"> 1. The system sends the data to the server, but server-side validation fails. 2. The server sends an error response with details about the validation errors. 3. The system displays these error messages to the admin. 4. The admin corrects the data based on the error messages and resubmits (back to step 5). <p>7a. Backend Server Unavailable:</p> <ol style="list-style-type: none"> 1. The system fails to connect to the backend server. 2. The system displays an error message to the admin. <p>7b. Database Error:</p> <ol style="list-style-type: none"> 1. There is a failure in updating the pharmacy information in the database. 2. The backend server responds with an error. 3. The system displays an error message to the admin.

TABLE 2.11.34 (UC-34) UPDATE EXISTING PHARMACIST

Use Case 34 – (UC – 34) Update Existing Pharmacist	
Related Requirements	REQ34
Initiating Actor:	Admin
Actor's Goal:	To modify the details of an existing pharmacist in the system.
Participating Actors:	Firebase Authentication
Preconditions:	<ul style="list-style-type: none"> • The admin is logged into the system with administrative privileges. • The system displays a list of existing pharmacists. • The admin has selected a pharmacist to update. • The system displays the current details of the selected pharmacist in an editable form.
Postconditions:	<ul style="list-style-type: none"> • The details of the selected pharmacist are updated in the system's database. • The system confirms the successful update to The admin. • The updated pharmacist information is reflected in the pharmacist list (potentially requiring a refresh or automatic update).
Flow of Events for Main Success Scenario:	
→	1. The admin navigates to the pharmacist management page.
→	2. The admin selects a pharmacist from the list to update.
←	3. The system displays the editable form along with the existing pharmacist details.
→	4. The admin makes changes to the necessary fields.
→	5. The admin saves the changed form.
←	6. The system checks the changed information (e.g., correct formats, mandatory fields).
→	7. The system sends the changed information to the backend server for updating the pharmacist record.
←	8. The backend server updates the pharmacist information in the database.
←	9. The backend server sends confirmation to the system.
←	10. The system displays a success message to the admin.
Flow of Events for Extensions (Alternate Scenarios):	
	4a. Invalid Input:
→	1. The admin enters invalid data in one or more fields.
←	2. The system identifies the invalid field(s) and displays related error messages.
→	3. The admin corrects the invalid data and resubmits the form.
	6a. Validation Error on Server:
←	1. The system posts the data to the server, but server-side validation fails.
←	2. The server responds with an error response and details of the validation failures.
←	3. The system displays these error messages to the admin.
→	4. The admin corrects the data as per the error messages and resubmits.
	7a. Backend Server Unavailable:
←	1. The system cannot connect to the backend server.
←	2. The system displays an error message to the admin.
	7b. Database Error:
←	1. An error occurs while updating the pharmacy details in the database.
←	2. The backend server returns an error.
←	3. The system displays an error message to the admin.

TABLE 2.11.35 (UC-35) VIEW LAB/IMAGING CENTERS LIST




Use Case 35 – (UC – 35) View Lab/Imaging Centers List	
Related Requirements	REQ35
Initiating Actor:	Admin
Actor's Goal:	To see a list of all lab and imaging centers registered in the system.
Participating Actors:	Firebase Authentication
Preconditions:	<ul style="list-style-type: none"> The admin is logged into the system with administrative privileges The admin navigates to the lab/imaging center management section.
Postconditions:	<ul style="list-style-type: none"> The new lab/imaging center's information is stored in the system's database. The system confirms the successful addition to the admin. The new center appears in the lab/imaging center list (potentially requiring a refresh or automatic update).
Flow of Events for Main Success Scenario:	
	<ol style="list-style-type: none"> 1. The admin navigates to the lab/imaging center management page. 2. The system asks the backend server for the lab/imaging center list. 3. The backend server retrieves the lab/imaging center details from the database. 4. The backend server sends the list of lab/imaging centers to the system. 5. The system receives the list of lab/imaging centers. 6. The system displays the list of lab/imaging centers to the admin.
Flow of Events for Extensions (Alternate Scenarios):	
	<p>2a. Backend Server Unavailable:</p> <ol style="list-style-type: none"> 1. The system cannot connect to the backend server. 2. The system displays an error message to the admin. <p>3a. No Centers Found:</p> <ol style="list-style-type: none"> 1. The backend server does not find any lab/imaging center data. 2. The system displays a message that there are no centers registered currently. <p>4a. Network Error During Fetch:</p> <ol style="list-style-type: none"> 1. There is a network error while fetching the list of pharmacists. 2. The system displays an error message to the admin. <p>6a. Display Error:</p> <ol style="list-style-type: none"> 1. There is an error while displaying the list by the system. 2. The system displays a generic error message to the admin.

TABLE 2.11.36 (UC-36) ADD NEW LAB/IMAGING CENTER

Use Case 36 – (UC – 36) Add New Lab/Imaging Center	
Related Requirements	REQ36
Initiating Actor:	Admin
Actor's Goal:	To register a new lab or imaging center in the system.
Participating Actors:	Firebase Authentication
Preconditions:	<ul style="list-style-type: none"> The admin is logged into the system with administrative. The admin navigates to the lab/imaging center management section. The system displays a form for adding a new center.
Postconditions:	<ul style="list-style-type: none"> The system displays a list of all lab and imaging centers, including relevant details (e.g., name, address, contact information).
Flow of Events for Main Success Scenario:	
	<ol style="list-style-type: none"> 1. The admin navigates to the lab/imaging center management section. 2. The admin initiates the process of adding a new center. 3. The system provides an empty form to enter the center details (e.g., name, address, contact information, type of center). 4. The admin enters the required information. 5. The admin submits the form.

←	6. The system validates the input of information (e.g., correct formats, mandatory fields, unique name/identifier if present).
←	7. Upon successful validation, the system sends the new center details to the backend server to add a new record.
←	8. The backend server adds the new center details to the database.
←	9. The backend server sends a confirmation response back to the system.
	10. The system displays a success message to the admin.
Flow of Events for Extensions (Alternate Scenarios):	
	4a. Invalid Input:
→	1. The admin enters invalid information in one or more fields.
←	2. The system identifies the invalid field(s) and presents appropriate error messages.
→	3. The admin re-validates the invalid information and resubmits the form (back to step 5 of the main success scenario).
	6a. Validation Error on Server:
←	1. The system sends the data to the server, but the server-side validation fails.
←	2. The server returns an error response with details of the failures of validation.
←	3. The system informs the admin of such error messages.
→	4. The admin corrects the data based on such error messages and resubmits (goes back to step 5).
	7a. Backend Server Unavailable:
←	1. The backend server is inaccessible to the system.
←	2. The system displays an error message to the admin.
	7b. Database Error:
←	1. While updating the pharmacy information in the database, there is a mistake.
←	2. The backend server sends an error response.
←	3. The system displays an error message to the admin.

TABLE 2.11.37 (UC-37) UPDATE EXISTING LAB/IMAGING CENTER

Use Case 37 – (UC – 37) Update Existing Lab/Imaging Center	
Related Requirements	REQ37
Initiating Actor:	Admin
Actor's Goal:	To modify the details of an existing lab or imaging center in the system.
Participating Actors:	Firestore Authentication
Preconditions:	<ul style="list-style-type: none"> The admin is logged into the system with administrative privileges. The system displays a list of existing lab/imaging centers. The admin has selected a center to update. The system displays the current details of the selected center in an editable form.
Postconditions:	<ul style="list-style-type: none"> The details of the selected lab/imaging center are updated in the system's database. The system confirms the successful update to the admin. The updated center information is reflected in the center list (potentially requiring a refresh or automatic update).
Flow of Events for Main Success Scenario:	
→	1. The admin logs into the lab/imaging center management section.
→	2. The admin selects a center from the list to update.
→	3. The system displays the editable form with the current center details.
→	4. The admin updates the mandatory fields (e.g., name, address, contact, center type).
→	5. The admin submits the updated form.
←	6. The system validates for correct data (e.g., formats, mandatory fields).
→	7. The system forwards the updated data to the backend server to update the center record.
←	8. The backend server makes the center details available in the updated database.
←	9. The backend server sends a successful response to the system.
←	10. The system displays a success message to the admin.
Flow of Events for Extensions (Alternate Scenarios):	
	4a. Invalid Input:
→	1. The admin enters wrong data in one or more fields.
←	2. The system identifies the wrong field(s) and displays good error messages.
→	3. The admin corrects the wrong data and re-submits the form
	6a. Server-side Validation Error:
←	1. The system submits the data to the server, but server-side validation fails.

←	2. The server sends back an error message with details about the cause for the failures in validation.
←	3. The system shows error messages to the admin.
→	4. The admin corrects the data based on the error messages and re-submits (return to step 5).
	7a. Backend Server Down:
←	1. The backend server cannot be accessed by the system.
←	2. The admin displays an error message by the system.
	7b. Database Error:
←	1. Error is faced in updating the pharmacy information in the database.
←	2. The backend server returns an error response.
	3. An error message is displayed to the admin by the system.

TABLE 2.11.38 (UC-38) VIEW CENTER STAFF LIST

Use Case 38 – (UC – 38) View Center Staff List	
Related Requirements	REQ38
Initiating Actor:	Admin
Actor's Goal:	To see a list of all staff members associated with lab/imaging centers.
Participating Actors:	Firebase Authentication
Preconditions:	<ul style="list-style-type: none"> The admin is logged into the system with administrative privileges. The admin navigates to the center staff management section.
Postconditions:	<ul style="list-style-type: none"> The system displays a list of all center staff members, including relevant details (e.g., name, contact information, associated center)
Flow of Events for Main Success Scenario:	
→	1. The admin visits the center staff management section.
→	2. The system asks the list of center staff from the backend server.
→	3. The backend server retrieves the center staff details from the database.
←	4. The backend server sends the list of center staff back to the system.
←	5. The system retrieves the list of center staff.
←	6. The system displays the list of center staff to the admin.
Flow of Events for Extensions (Alternate Scenarios):	
	2a. Backend Server Unavailable:
←	1. The backend server is unable to connect.
←	2. The system displays an error message to the admin.
	3a. No Staff Found:
←	1. No center staff information is retrieved by the backend server.
←	2. The system displays a message indicating that no staff members are allocated to centers currently.
	4a. Network Error During Fetch:
←	1. The network contains an error when fetching the center staff members list.
←	2. The system displays an error message to the admin.
	6a. Display Error:
←	1. The system fails when displaying the list.
←	2. The admin receives a generic error message from the system.

TABLE 2.11.39 (UC-39) ADD NEW CENTER STAFF MEMBER

Use Case 39 – (UC – 39) Add New Center Staff Member	
Related Requirements	REQ39
Initiating Actor:	Admin
Actor's Goal:	To register a new staff member associated with a lab or imaging center.
Participating Actors:	Firebase Authentication
Preconditions:	<ul style="list-style-type: none"> • The admin is logged into the system with administrative privileges. • The admin navigates to the center staff management section. • The system displays a form for adding a new staff member.
Postconditions:	<ul style="list-style-type: none"> • The new center staff member's information is stored in the system's database. • The system confirms the successful addition to the admin. • The new staff member appears in the center staff list (potentially requiring a refresh or automatic update).
Flow of Events for Main Success Scenario:	
→	1. The admin logs in to the center staff management section.
→	2. The admin initiates the process of adding a new staff member.
←	3. The system displays a blank form to enter the details of the staff member
→	4. The admin enters the required details.
→	5. The admin saves the form.
←	6. The system verifies the details entered for correctness.
←	7. The system sends the new staff member details to the backend server to create a new record.
←	8. The backend server adds the new staff member details to the database.
←	9. The backend server sends a confirm response back to the system.
←	10. The system provides the admin with success notification.
Flow of Events for Extensions (Alternate Scenarios):	
→	4a. Invalid Input:
←	1. The admin enters invalid data in one or more fields.
→	2. The system identifies the invalid field(s) and displays appropriate error messages.
→	3. The admin corrects the invalid data and resubmits the form.
←	6a. Validation Error on Server:
←	1. The system sends the data to the server, but server-side validation fails.
←	2. The server returns an error along with details of the validation failures.
←	3. The system displays these error messages to the admin.
→	4. The admin corrects the data as per the error messages and resubmits (back to step 5).
←	7a. Backend Server Unavailable:
←	1. The system cannot connect to the backend server.
←	2. The system displays an error message to the admin.
←	7b. Database Error:
←	1. There is an error in updating the pharmacy data in the database.
←	2. The backend server returns an error response.
←	3. The system displays an error message to the admin.

TABLE 2.11.40 (UC-40) UPDATE EXISTING CENTER STAFF MEMBER

Use Case 40 – (UC – 40) Update Existing Center Staff Member	
Related Requirements	REQ40
Initiating Actor:	Admin
Actor's Goal:	To modify the details of an existing staff member associated with a lab or imaging center.
Participating Actors:	Firebase Authentication
Preconditions:	<ul style="list-style-type: none"> • The admin is logged into the system with administrative privileges. • The system displays a list of existing center staff members. • The admin has selected a staff member to update. • The system displays the current details of the selected staff member in an editable form.
Postconditions:	<ul style="list-style-type: none"> • The details of the selected center staff members are updated in the system's database. • The system confirms the successful update to the admin. • The updated staff member information is reflected in the center staff list (potentially requiring a refresh or automatic update).
Flow of Events for Main Success Scenario:	
→	1. The admin logs into the pharmacist management section.
→	2. The admin selects a center staff member to update from the list.
←	3. The system displays the editable form with the current staff member details.
→	4. The admin makes the necessary updates.
→	5. The admin submits the updated form.
←	6. The system validates the new information (e.g., ensures it is valid format, there are no missing required fields).
→	7. The system sends the new data to the backend server to update staff member records.
←	8. The backend server updates staff member information in the database.
←	9. The backend server sends a successful response to the system.
←	10. The system displays a success message to the admin.
Flow of Events for Extensions (Alternate Scenarios):	
→	4a. Invalid Input:
←	1. The admin enters erroneous data in one or more fields.
→	2. The system identifies the erroneous field(s) and displays corresponding error messages.
→	3. The admin corrects the erroneous data and resubmits the form.
←	6a. Validation Error on Server:
←	1. The system sends the data to the server, but server-side validation fails.
←	2. The server sends an error that contains details of the validation errors.
←	3. The system displays these error messages to the admin.
→	4. Admin corrects them as per the error messages and re-submits.
←	7a. Backend Server Unavailable:
←	1. The backend server is unavailable to the system.
←	2. The system displays an error message to the admin.
←	7b. Database Error:
←	1. Failure in update while changing the pharmacy details in the database.
←	2. Failure response from the backend server.
←	3. The system displays an error message to the admin.

2.12. TRACEABILITY MATRIX

TABLE 2.12.1 TRACEABILITY MATRIX

REQ	PW	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7	UC-8	UC-9	UC-10	UC-11	UC-12	UC-13	UC-14	UC-15
REQ1	5	X														
REQ2	5		X													
REQ3	3			X												
REQ4	3				X											
REQ5	4					X										
REQ6	4						X									
REQ7	4							X								
REQ8	5								X							
REQ9	4									X						
REQ10	4										X					
REQ11	3											X				
REQ12	4												X			
REQ13	4													X		
REQ14	4														X	
REQ15	4															X
Max PW		5	5	3	3	4	4	4	5	4	4	3	4	4	4	4
Total PW		5	5	3	3	4	4	4	5	4	4	3	4	4	4	4

REQ	PW	UC-16	UC-17	UC-18	UC-19	UC-20	UC-21	UC-22	UC-23	UC-24	UC-25	UC-26	UC-27
REQ16	4	X											
REQ17	4		X										
REQ18	4			X									
REQ19	4				X								
REQ20	5					X							
REQ21	5						X						
REQ22	5							X					
REQ23	5								X				
REQ24	5									X			
REQ25	5										X		
REQ26	5											X	
REQ27	5												X
Max PW		4	4	4	4	5	5	5	5	5	5	5	5
Total PW		4	4	4	4	5	5	5	5	5	5	5	5

REQ	PW	UC-28	UC-29	UC-30	UC-31	UC-32	UC-33	UC-34	UC-35	UC-36	UC-37	UC-38	UC-39	UC-40
REQ28	5	X												
REQ29	5		X											
REQ30	5			X										
REQ31	5				X									
REQ32	5					X								
REQ33	5						X							
REQ34	5							X						
REQ35	5								X					
REQ36	5									X				
REQ37	5										X			
REQ38	5											X		
REQ39	5												X	
REQ40	5													X
Max PW		5	5	5	5	5	5	5	5	5	5	5	5	5
Total PW		5	5	5	5	5	5	5	5	5	5	5	5	5

2.13. ACTIVITY DIAGRAMS

ICare—Patient Register an Account Activity Diagram

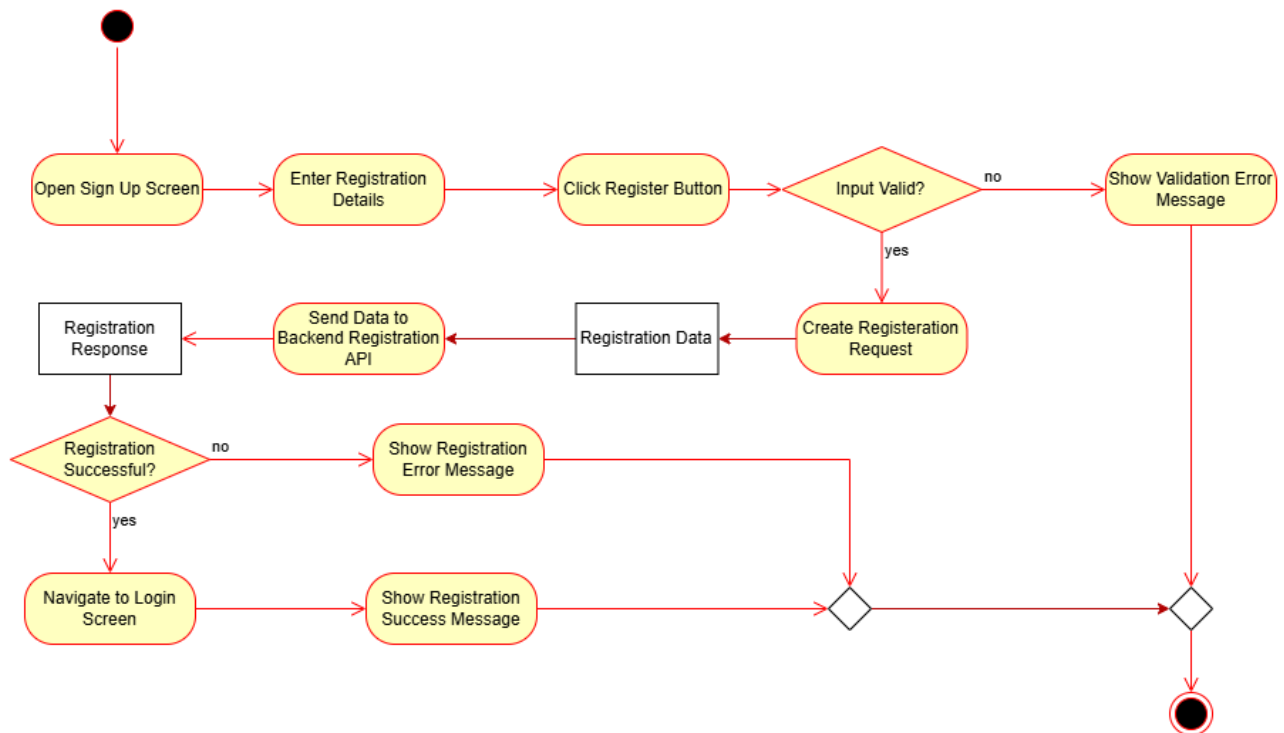


FIGURE 2.13.1 PATIENT REGISTER AN ACCOUNT ACTIVITY DIAGRAM

ICare - User Login Using Email & Password Activity Diagram

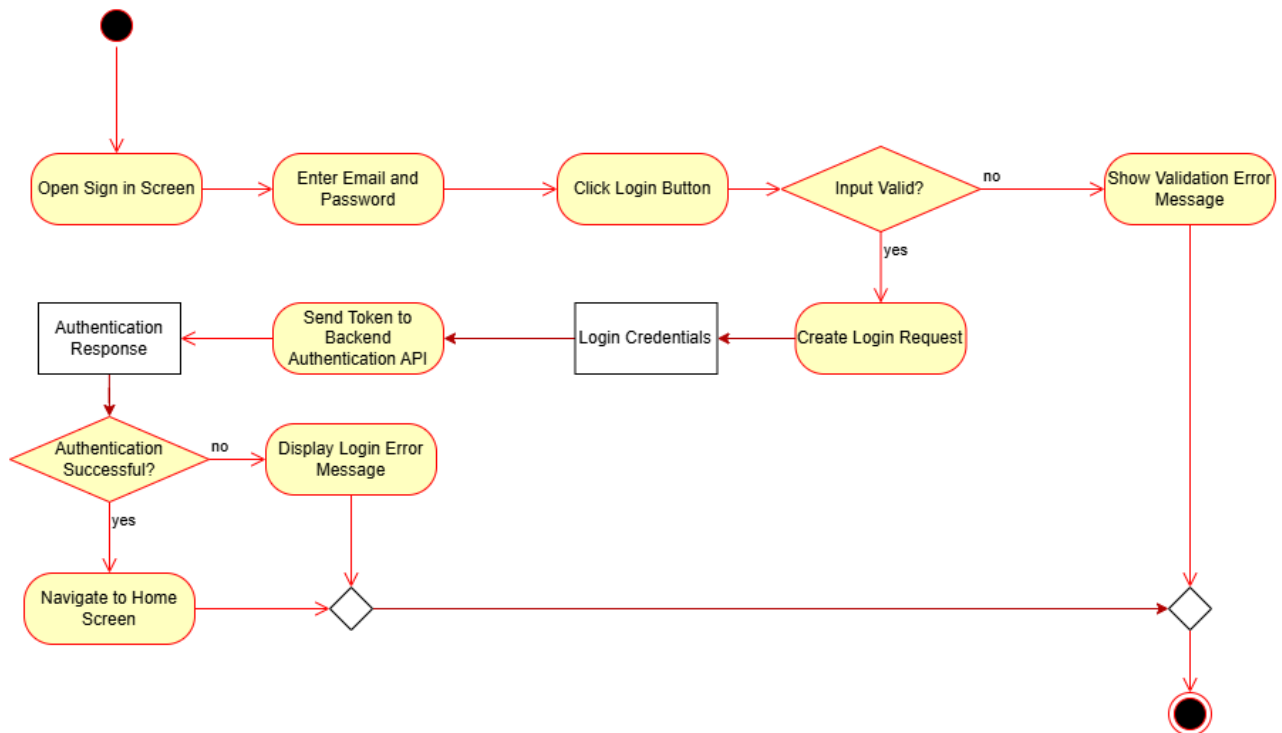


FIGURE 2.13.2 USER LOGIN USING EMAIL & PASSWORD ACTIVITY DIAGRAM

ICare - User Link Google Account Activity Diagram

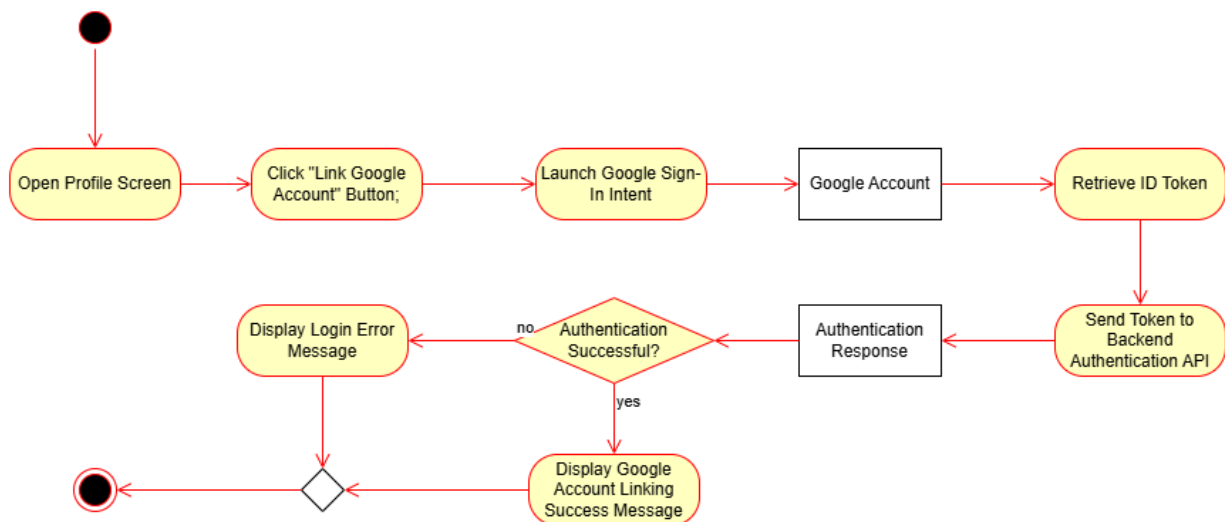


FIGURE 2.13.3 USER LINK GOOGLE ACCOUNT ACTIVITY DIAGRAM

ICare - User Login Using Google Account Activity Diagram

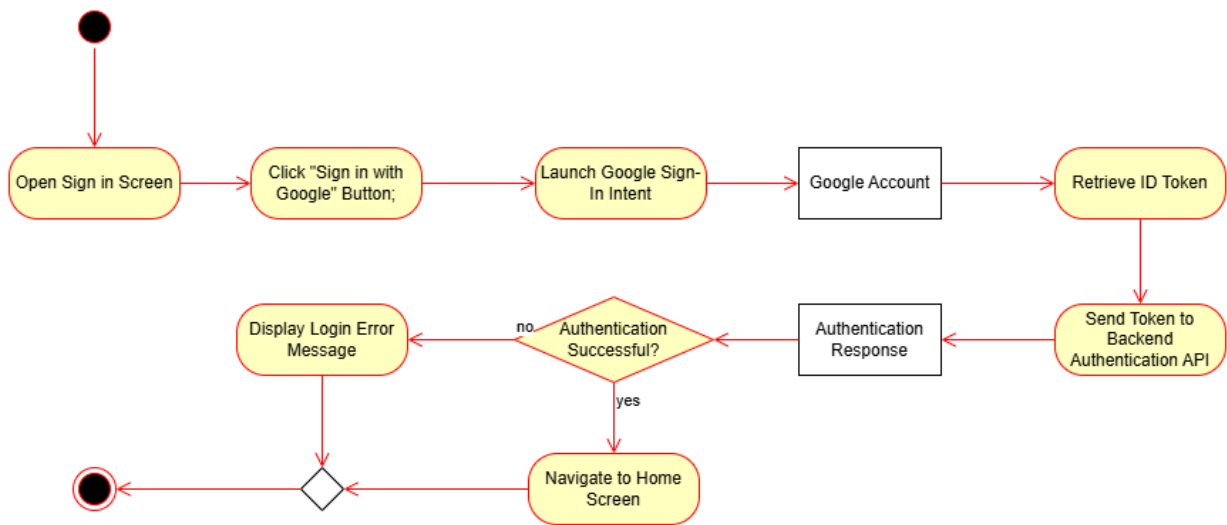


FIGURE 2.13.4 USER LOGIN USING GOOGLE ACCOUNT ACTIVITY DIAGRAM

ICare - Patient Book Appointment Activity Diagram

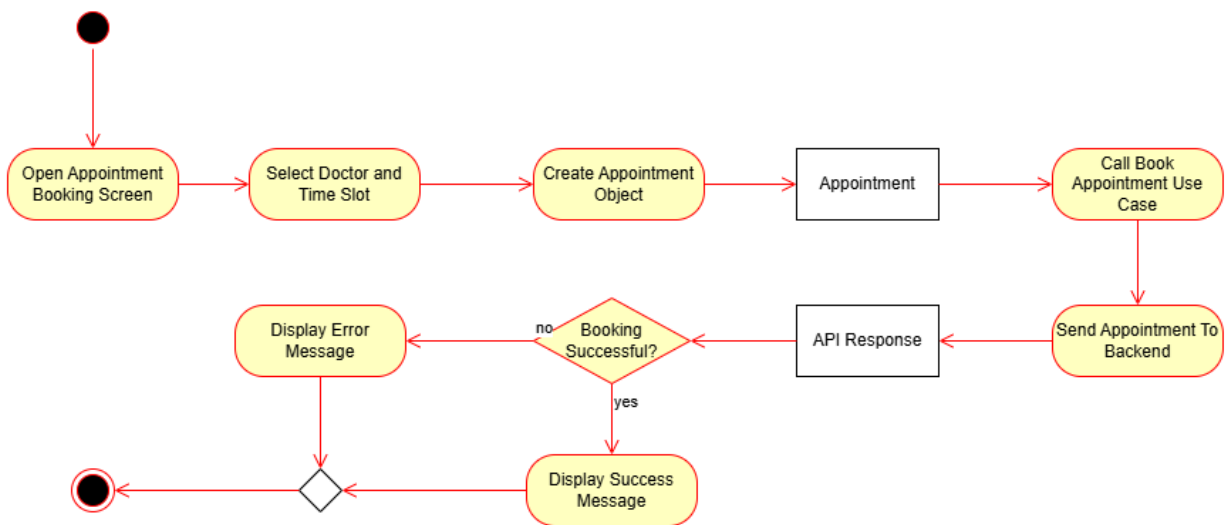


FIGURE 2.13.5 PATIENT BOOK APPOINTMENT ACTIVITY DIAGRAM

ICare - Clinic Staff Confirm Appointment Activity Diagram

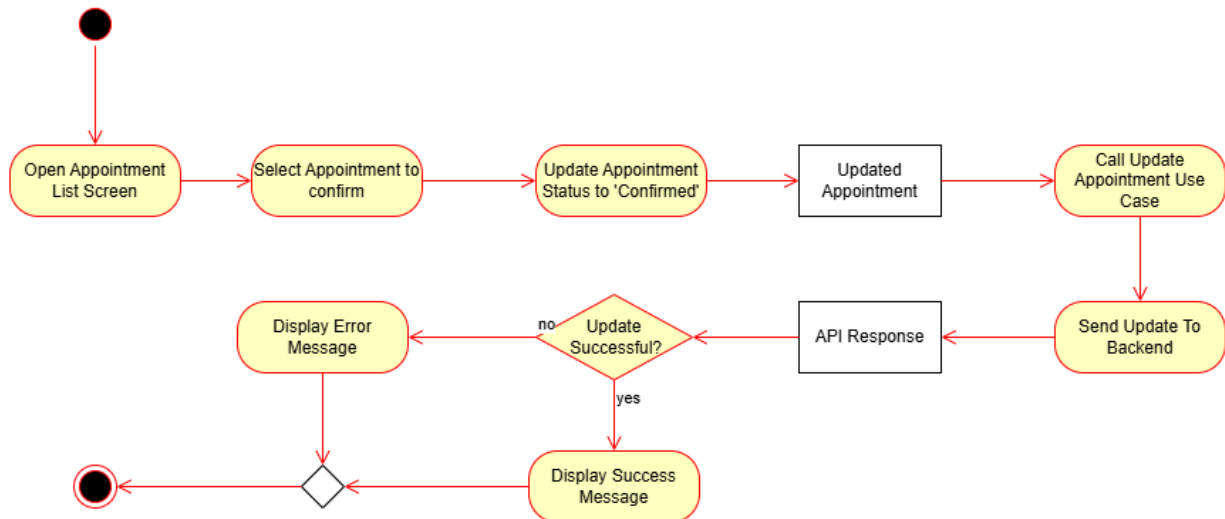


FIGURE 2.13.6 CLINIC STAFF CONFIRM APPOINTMENT ACTIVITY DIAGRAM

ICare - Doctor Add Consultation Activity Diagram

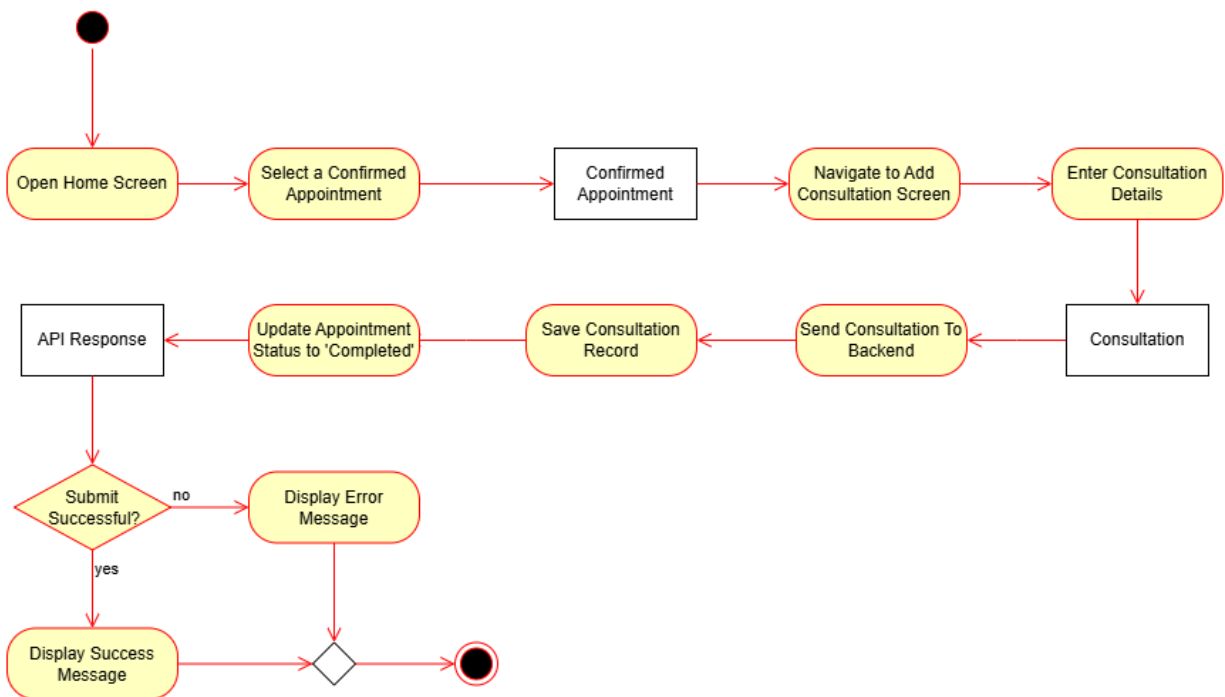


FIGURE 2.13.7 DOCTOR ADD CONSULTATION ACTIVITY DIAGRAM

ICare - Admin Add a New Entity(Clinic, Doctor, Clinic Staff, Pharmacy, Pharmacist, Center, or Center Staff) Activity Diagram

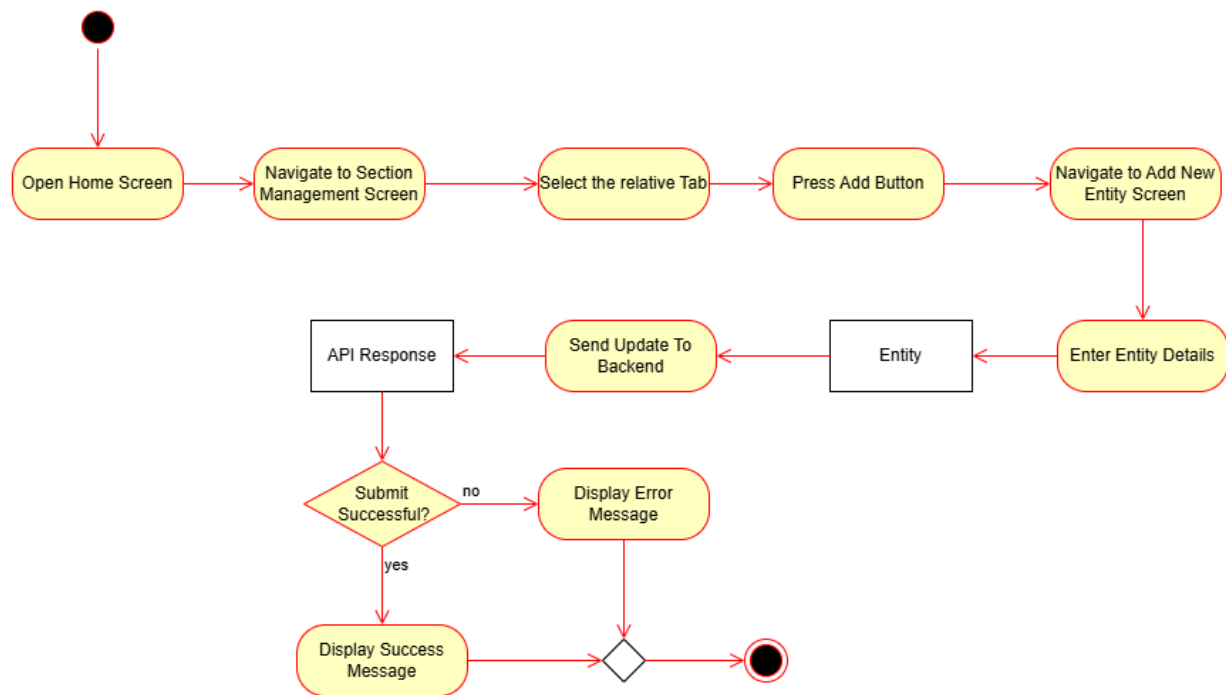


FIGURE 2.13.8 ADMIN ADD A NEW ENTITY (CLINIC, DOCTOR, CLINIC STAFF, PHARMACY, PHARMACIST, CENTER, OR CENTER STAFF) ACTIVITY DIAGRAM

ICare - Admin Update Existing Entity(Clinic, Doctor, Clinic Staff, Pharmacy, Pharmacist, Center, or Center Staff) Activity Diagram

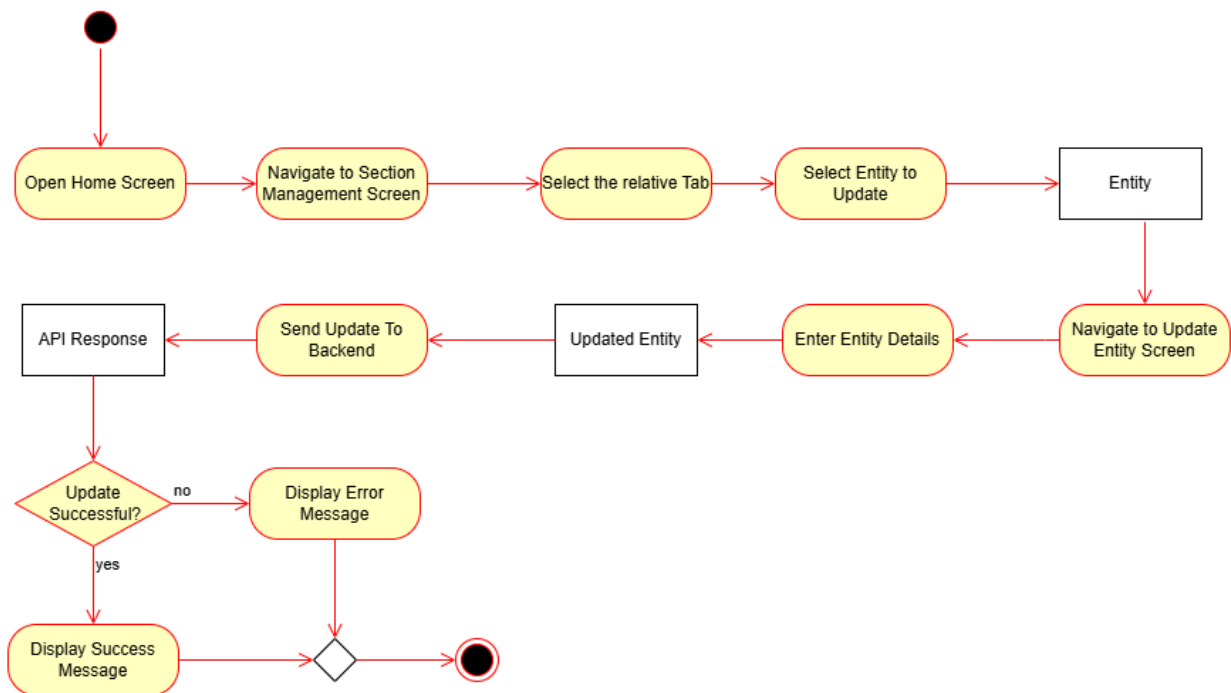


FIGURE 2.13.10 ADMIN UPDATE EXISTING ENTITY (CLINIC, DOCTOR, CLINIC STAFF, PHARMACY, PHARMACIST, CENTER, OR CENTER STAFF) ACTIVITY DIAGRAM

ICare - Admin View Cached Entity (Clinic, Doctor, Pharmacy, Center) List Activity Diagram

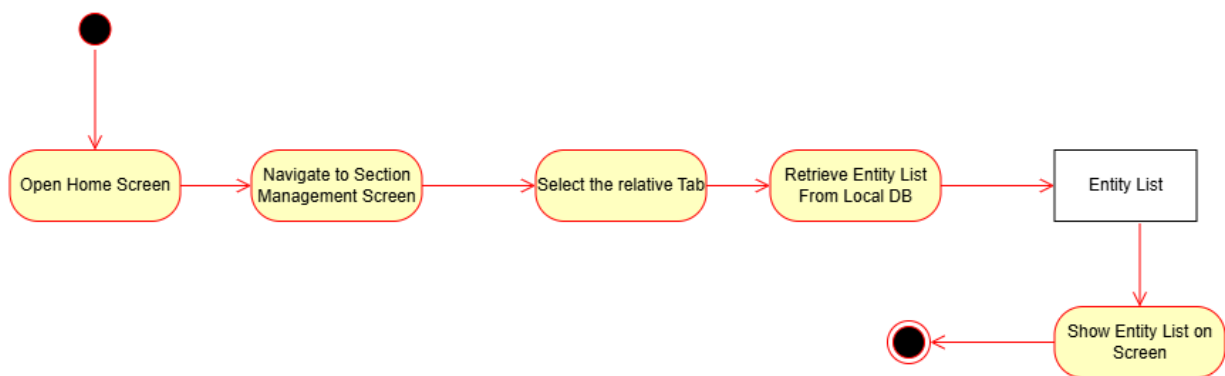


FIGURE 2.13.9 ADMIN VIEW CACHED ENTITY (CLINIC, DOCTOR, PHARMACY, CENTER) LIST ACTIVITY DIAGRAM

ICare - Admin View Non-Cashable Entity (Clinic Staff, Pharmacist, or Center Staff) List Activity Diagram

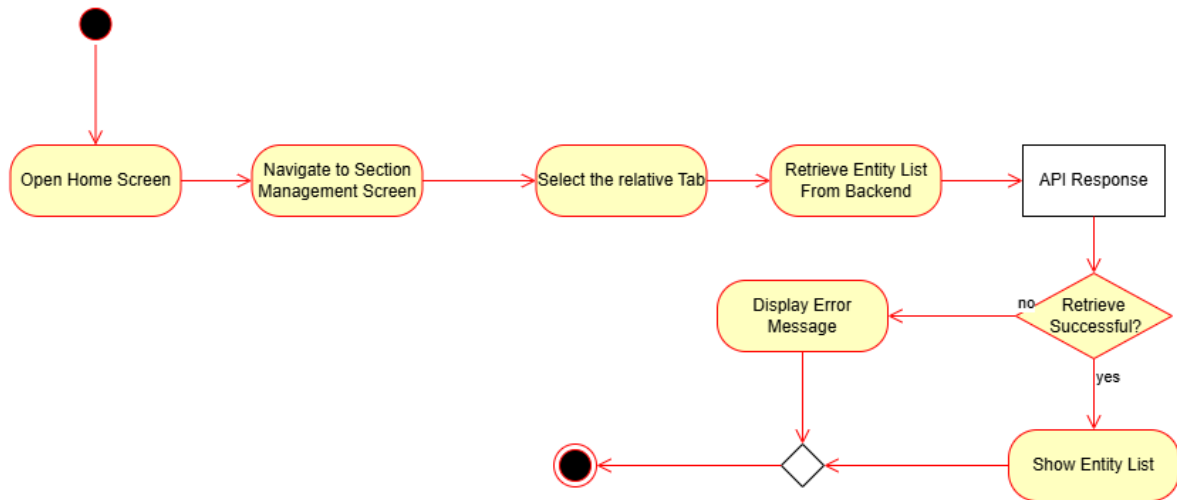


FIGURE 2.13.11 ADMIN VIEW NON-CASHABLE ENTITY (CLINIC STAFF, PHARMACIST, OR CENTER STAFF) LIST ACTIVITY DIAGRAM

ICare - User Fetch Data (Clinics, Pharmacies, Centers, and Doctors) From The Backend Activity Diagram

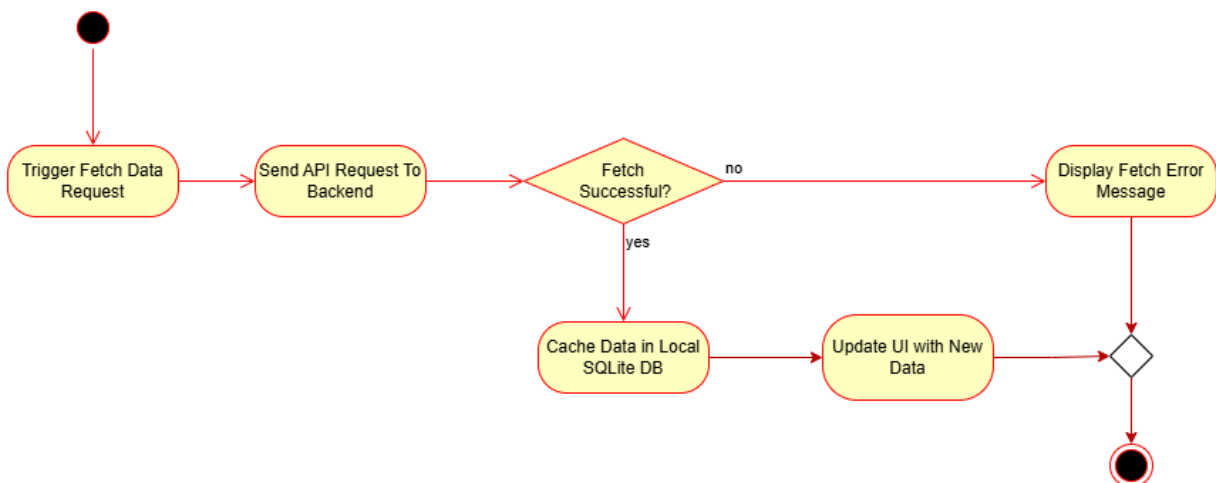
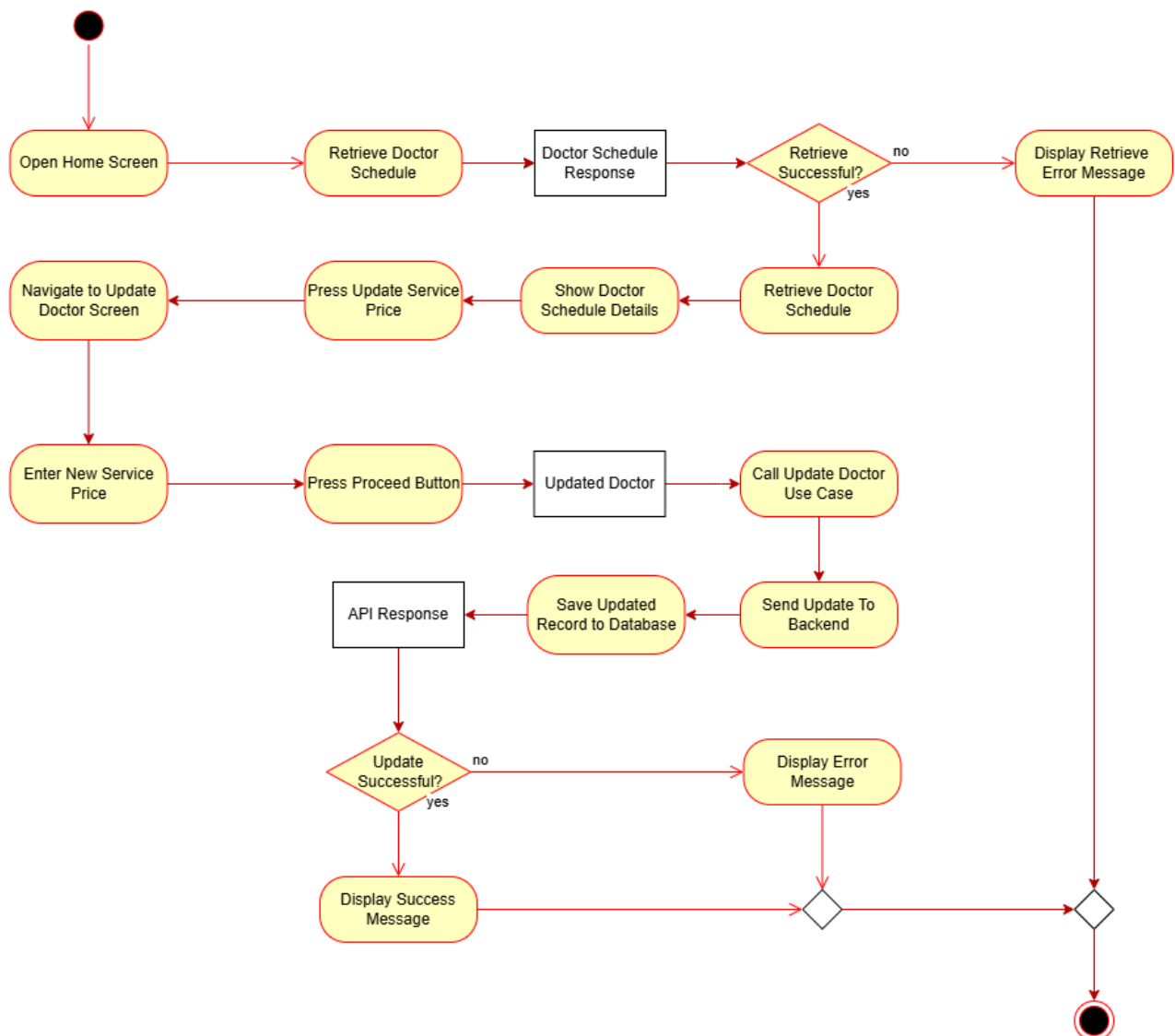


FIGURE 2.13.12 USER FETCH DATA (CLINICS, PHARMACIES, CENTERS, AND DOCTORS) FROM THE BACKEND ACTIVITY DIAGRAM

FIGURE 2.13.13 DOCTOR VIEW HIS SCHEDULE AND UPDATE HIS SERVICE PRICE ACTIVITY DIAGRAM

ICare - Dctor View His Schedule and Update His Service Price Activity Diagram



ICare - Patient Re-schedule Appointment Activity Diagram

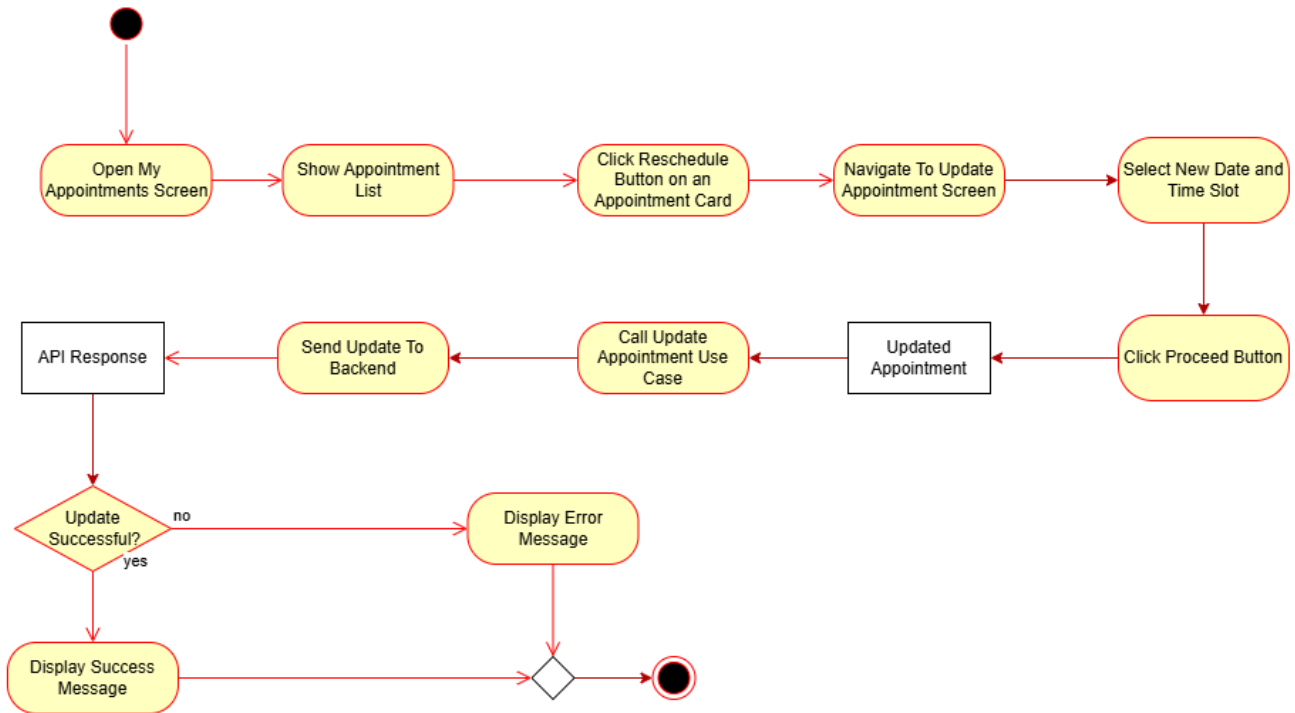


FIGURE 2.13.14 PATIENT RE-SCHEDULE APPOINTMENT ACTIVITY DIAGRAM

ICare - Patient View Top Doctors Activity Diagram



FIGURE 2.13.15 PATIENT VIEW TOP DOCTORS ACTIVITY DIAGRAM

ICare - Patient Cancel Appointment Activity Diagram

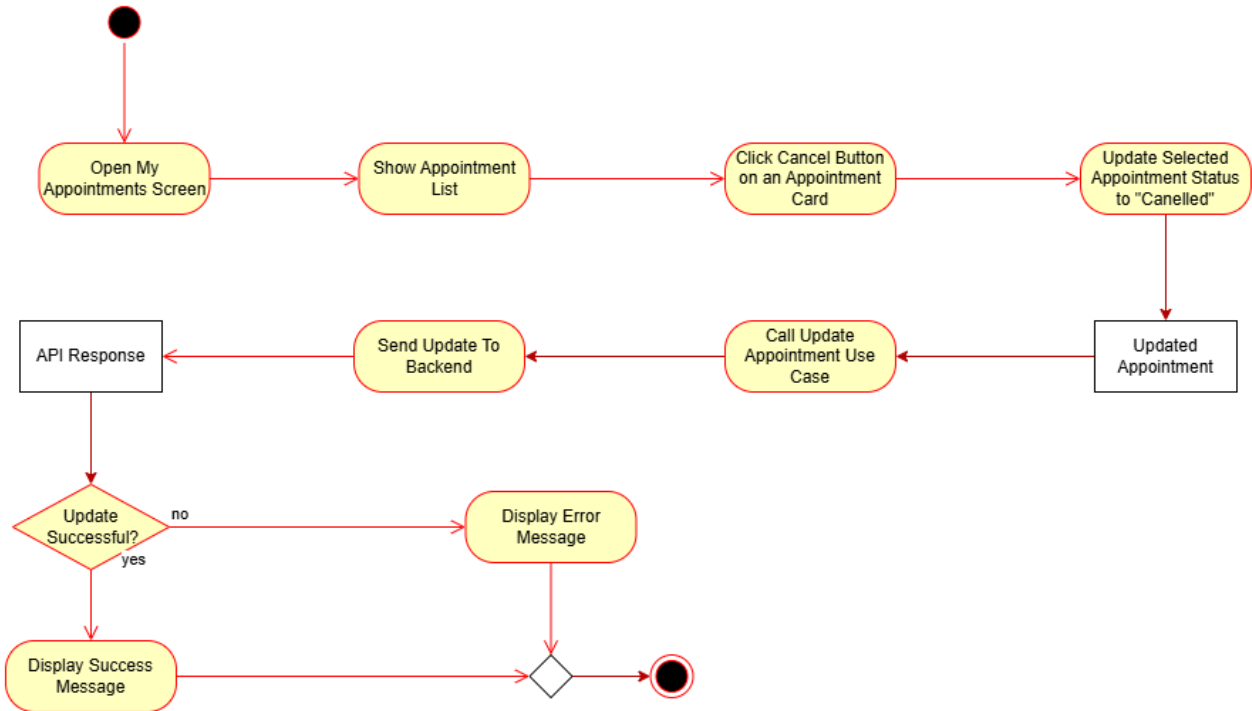


FIGURE 2.13.16 PATIENT CANCEL APPOINTMENT ACTIVITY DIAGRAM

ICare - Patient View Cached Entity (Doctor, Pharmacy, Center) List Activity Diagram

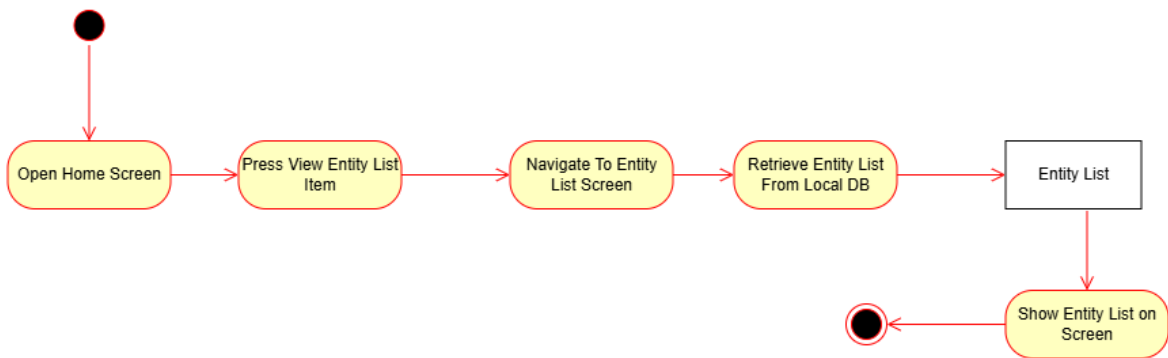


FIGURE 2.13.17 PATIENT VIEW CACHED ENTITY (DOCTOR, PHARMACY, CENTER) LIST ACTIVITY DIAGRAM

ICare - Doctor View Medical Record and Update Consultation Activity Diagram

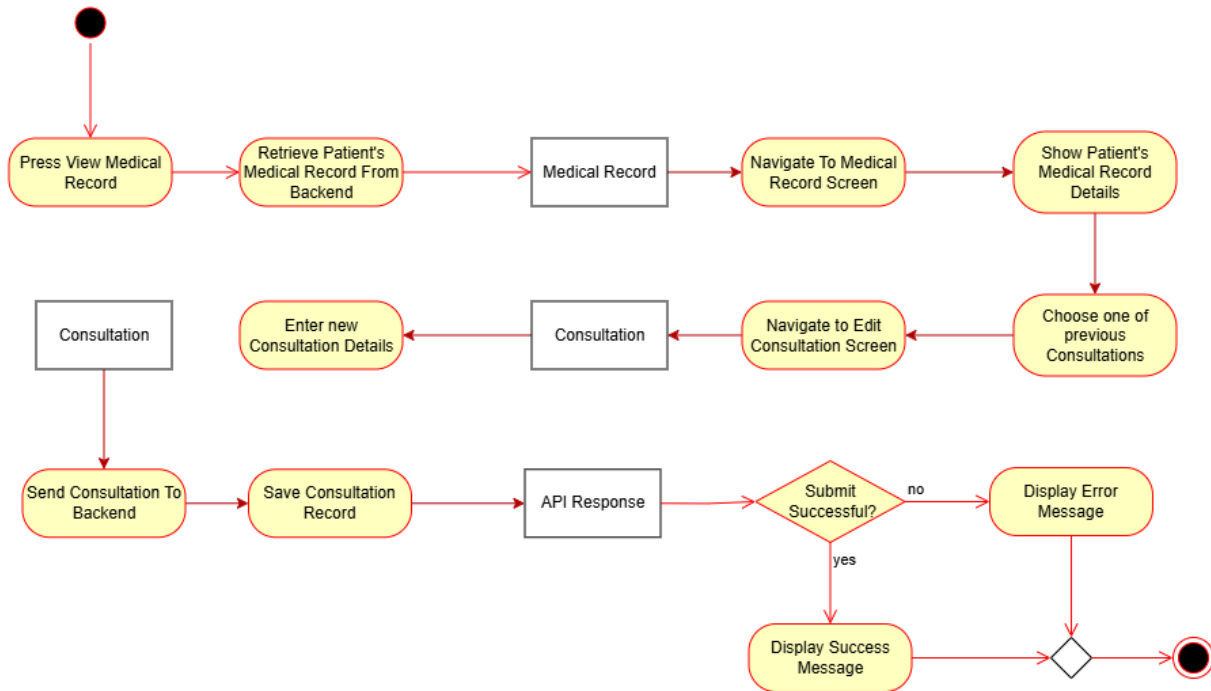


FIGURE 2.13.18 DOCTOR VIEW MEDICAL RECORD AND UPDATE CONSULTATION ACTIVITY DIAGRAM

ICare - Clinic Staff View Pending and Confirmed Appointments Activity Diagram

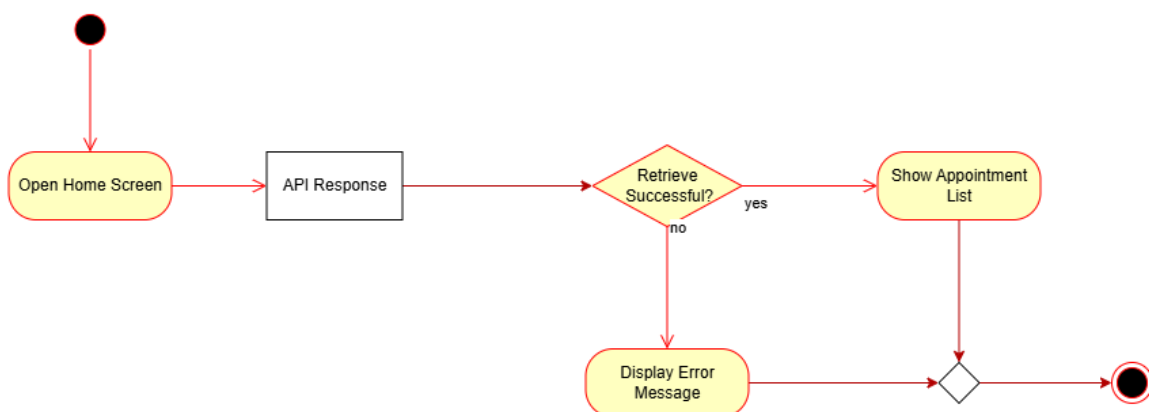


FIGURE 2.13.19 CLINIC STAFF VIEW PENDING AND CONFIRMED APPOINTMENTS ACTIVITY DIAGRAM

2.14. SEQUENCE DIAGRAMS

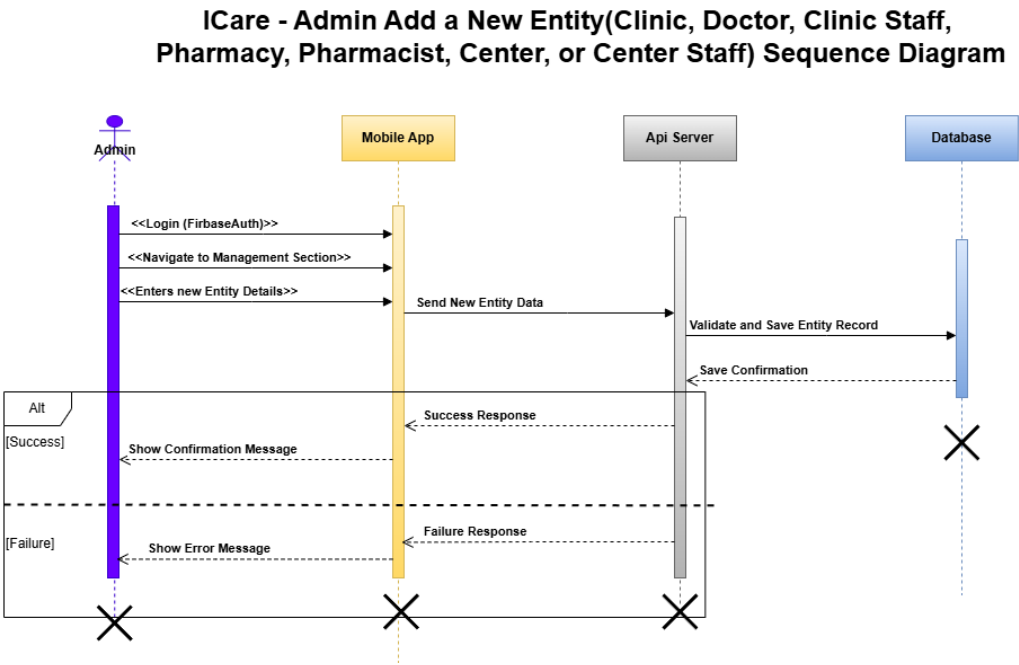


FIGURE 2.14.1 ADMIN ADD A NEW ENTITY(CLINIC, DOCTOR, CLINIC STAFF, PHARMACY, PHARMACIST, CENTER, OR CENTER STAFF) SEQUENCE DIAGRAM

ICare - Admin View Cached Entity (Clinic, Doctor, Pharmacy, Center) List Sequence Diagram

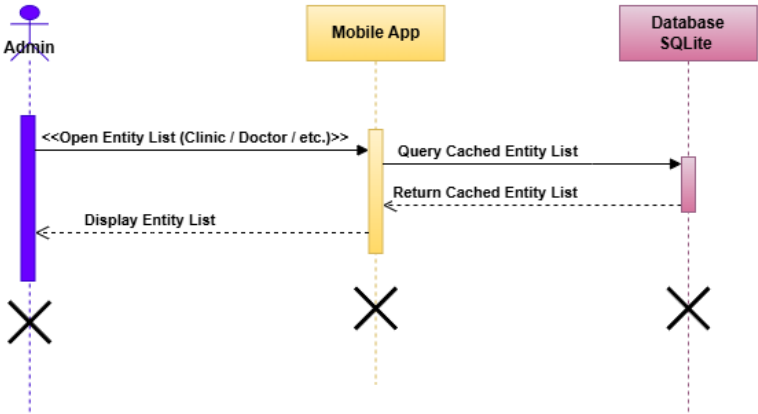


FIGURE 2.14.2 ADMIN VIEW Cached ENTITY (CLINIC, DOCTOR, PHARMACY, CENTER) LIST SEQUENCE DIAGRAM

ICare - Admin Update Existing Entity(Clinic, Doctor, Clinic Staff, Pharmacy, Pharmacist, Center, or Center Staff) Sequence Diagram

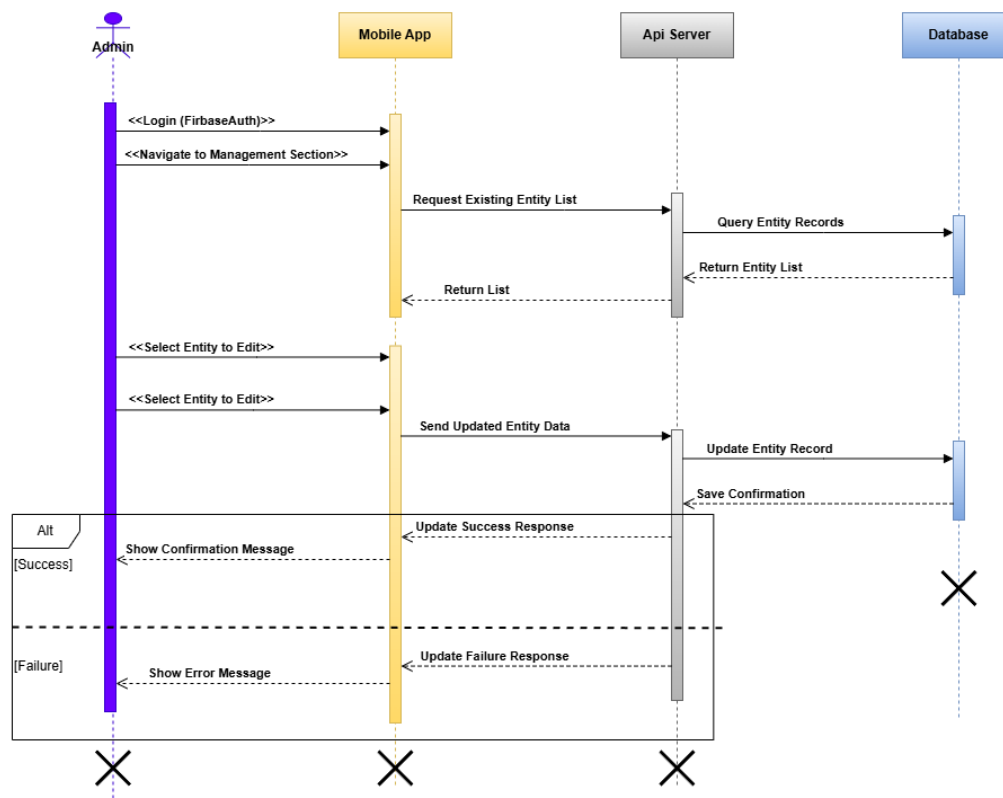


FIGURE 2.14.3 ADMIN UPDATE EXISTING ENTITY(CLINIC, DOCTOR, CLINIC STAFF, PHARMACY, PHARMACIST, CENTER, OR CENTER STAFF) SEQUENCE DIAGRAM

ICare - Admin View Non-Cashable Entity (Clinic Staff, Pharmacist, or Center Staff) List Sequence Diagram

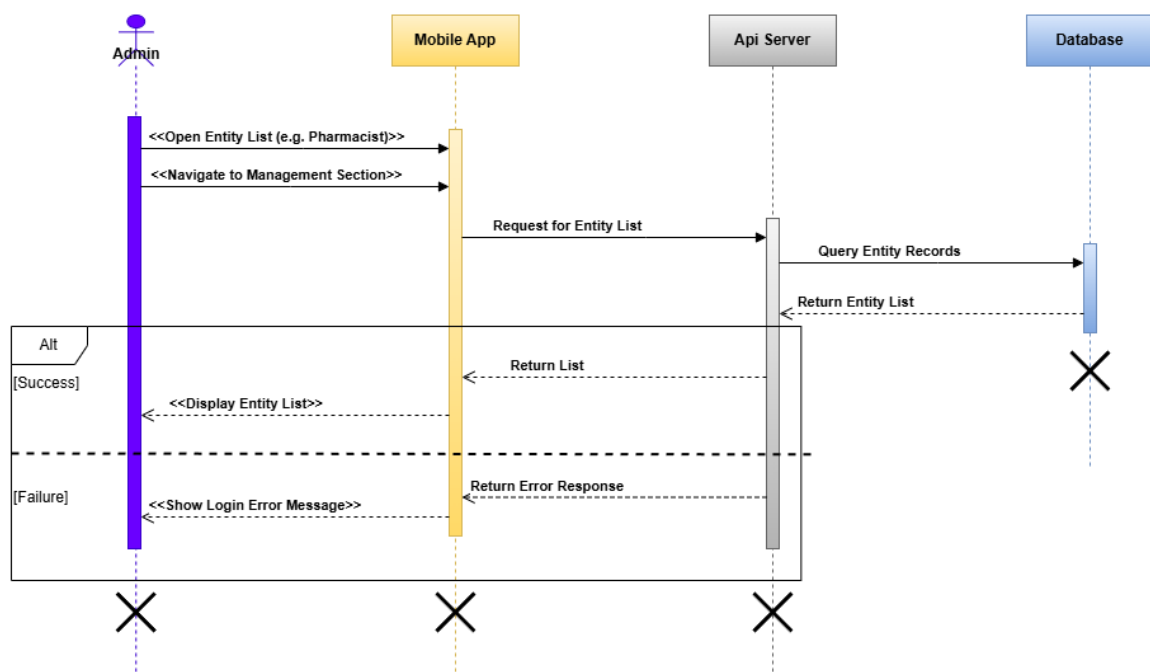


FIGURE 2.14.4 ADMIN VIEW NON-CASHABLE ENTITY (CLINIC STAFF, PHARMACIST, OR CENTER STAFF) LIST SEQUENCE DIAGRAM

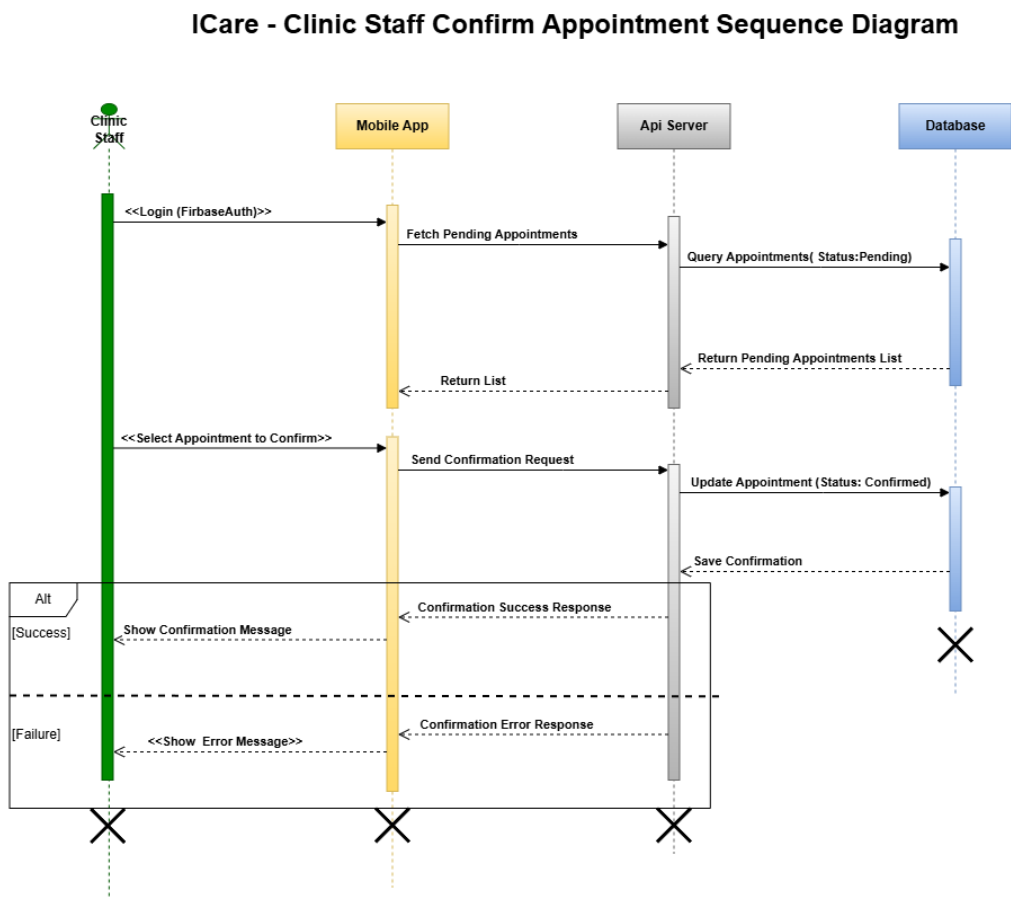


FIGURE 2.14.5 CLINIC STAFF CONFIRM APPOINTMENT SEQUENCE DIAGRAM

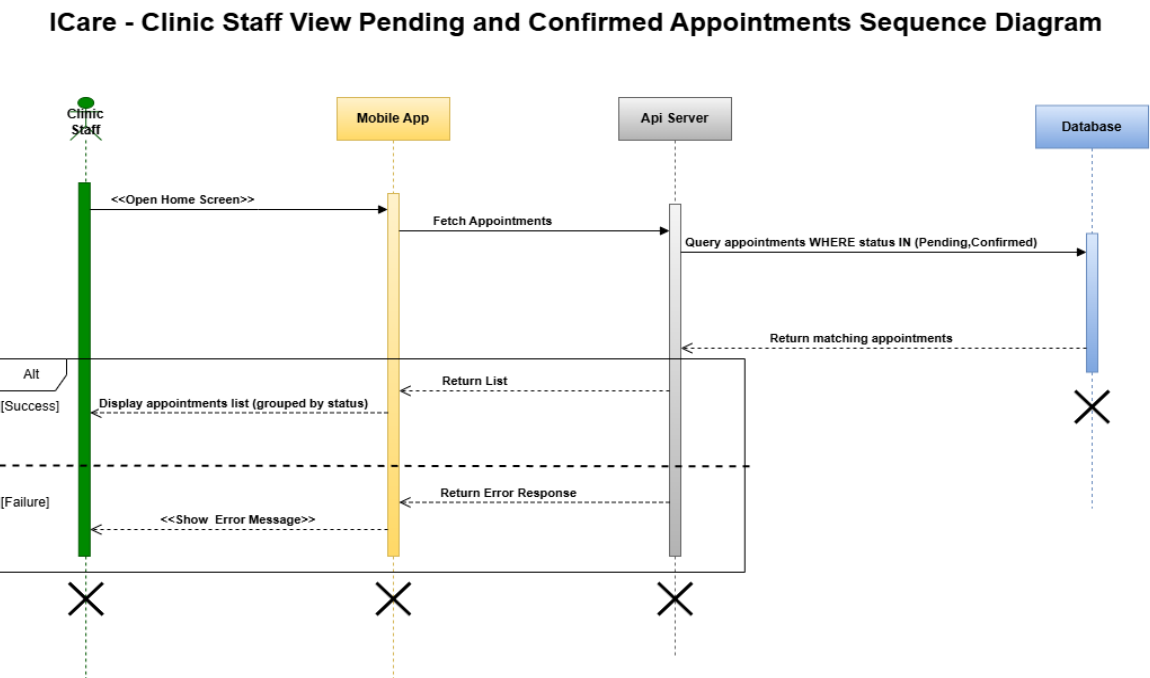


FIGURE 2.14.6 CLINIC STAFF VIEW PENDING AND CONFIRMED APPOINTMENTS SEQUENCE DIAGRAM

ICare - Doctor Add Consultation Sequence Diagram

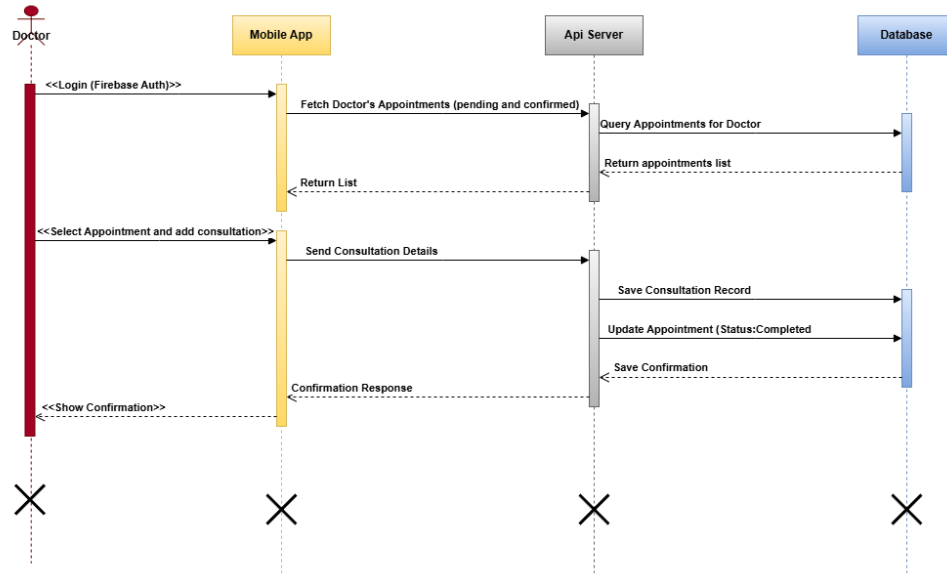


FIGURE 2.14.7 DOCTOR ADD CONSULTATION SEQUENCE DIAGRAM

ICare - Doctor View Medical Record and Update Consultation Sequence Diagram

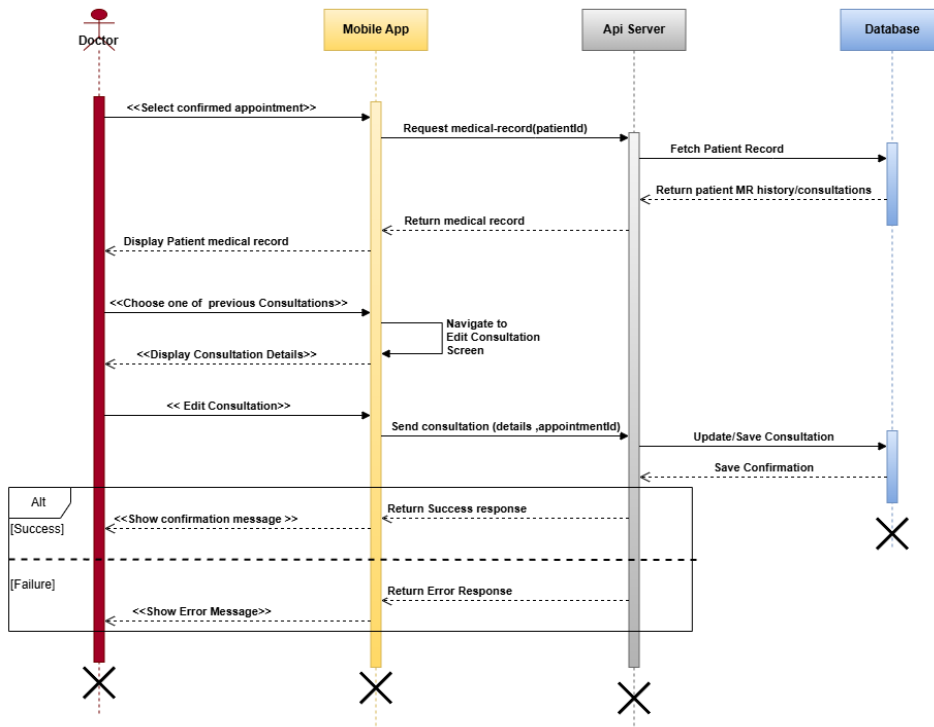


FIGURE 2.14.8 DOCTOR VIEW MEDICAL RECORD AND UPDATE CONSULTATION SEQUENCE DIAGRAM

ICare - Patient View Cached Entity (Doctor, Pharmacy, Center) List Sequence Diagram

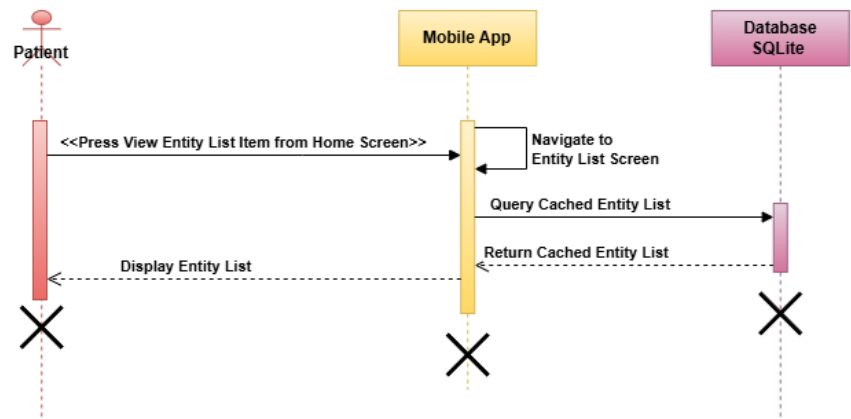


FIGURE 2.14.9 PATIENT VIEW CACHED ENTITY (DOCTOR, PHARMACY, CENTER) LIST SEQUENCE DIAGRAM

ICare - Patient View Top Doctors Sequence Diagram

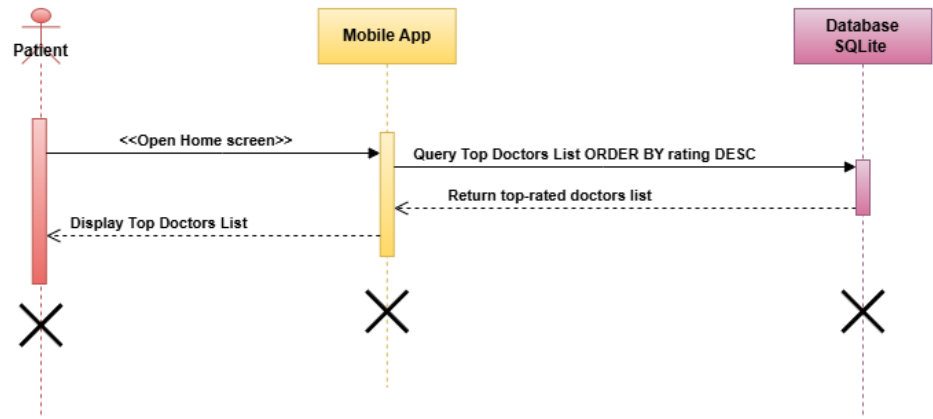


FIGURE 2.14.10 PATIENT VIEW TOP DOCTORS SEQUENCE DIAGRAM

ICare - User Fetch Data (Clinics, Pharmacies, Centers, and Doctors) From The Backend Sequence Diagram

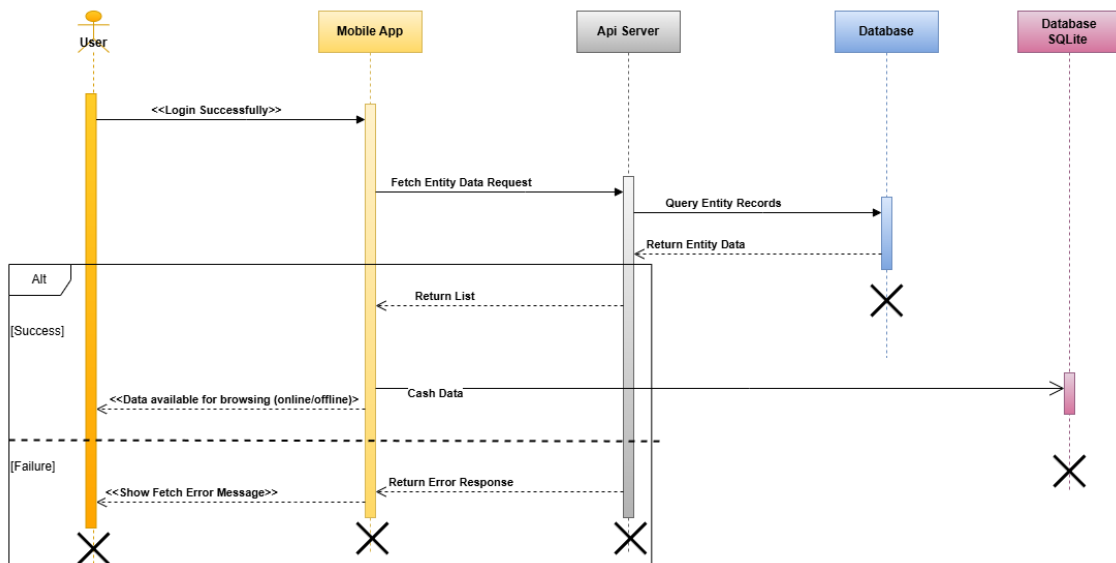


FIGURE 2.14.11 USER FETCH DATA (CLINICS, PHARMACIES, CENTERS, AND DOCTORS) FROM THE BACKEND SEQUENCE DIAGRAM

ICare - User Link Google Account Sequence Diagram

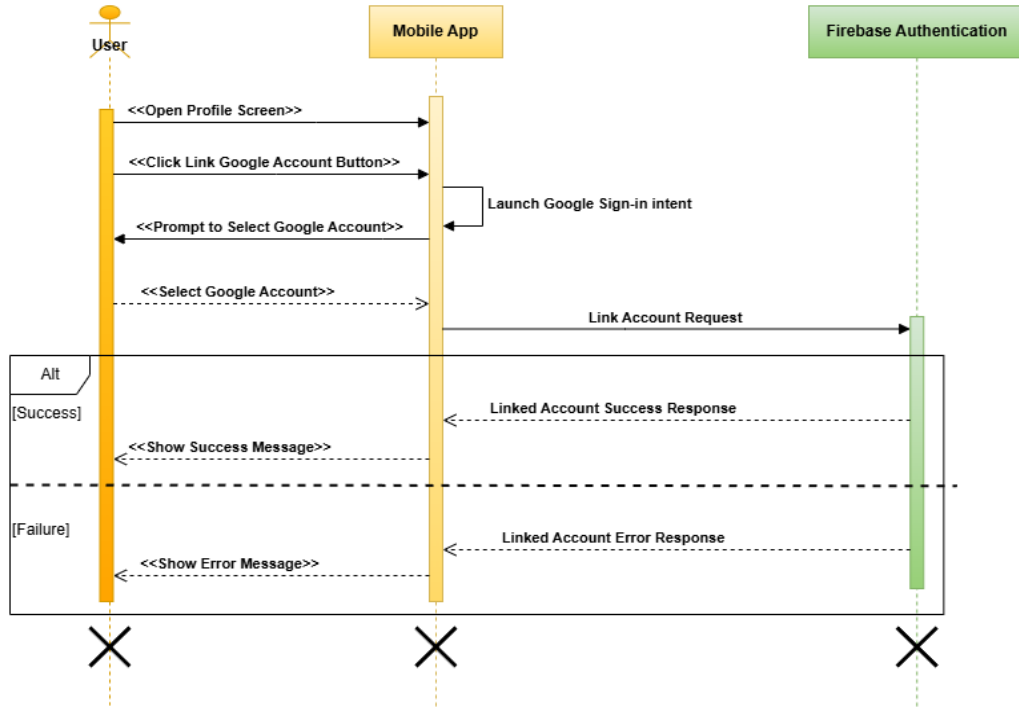


FIGURE 2.14.12 USER LINK GOOGLE ACCOUNT SEQUENCE DIAGRAM

ICare - User Login Using Google Account Sequence Diagram

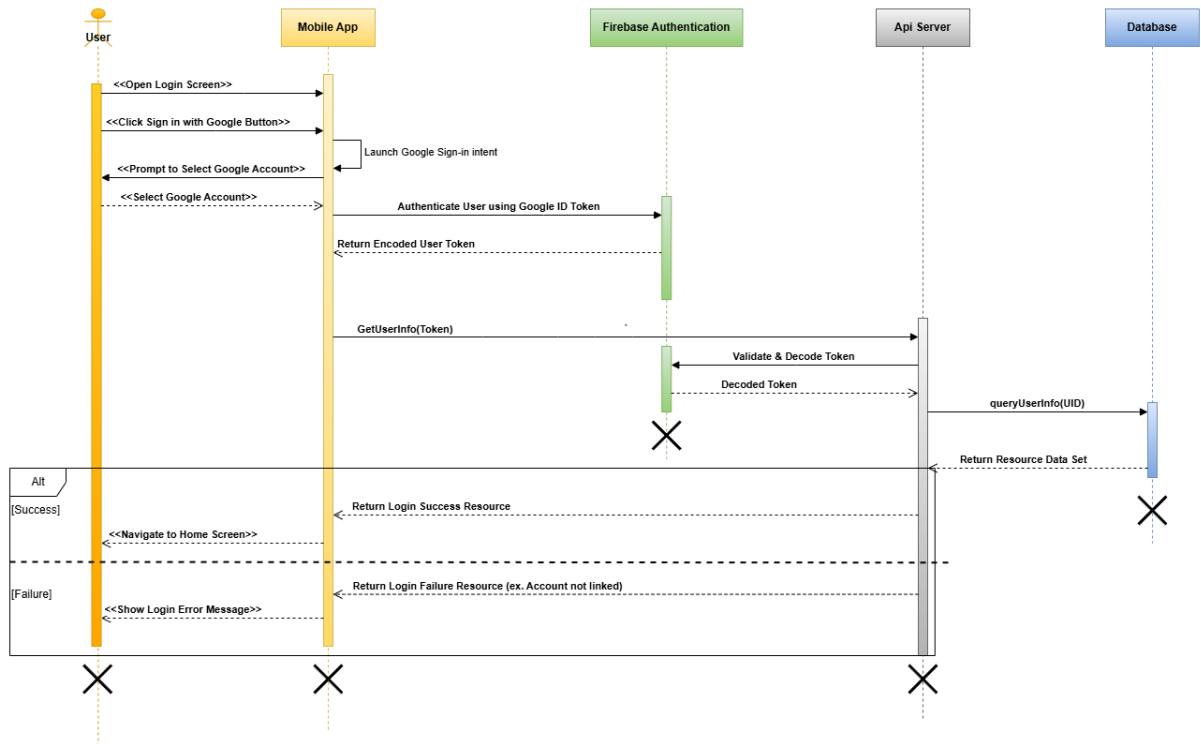


FIGURE 2.14.13 USER LOGIN USING GOOGLE ACCOUNT SEQUENCE DIAGRAM

ICare - Doctor View His Schedule and Update His Service Price Sequence Diagram

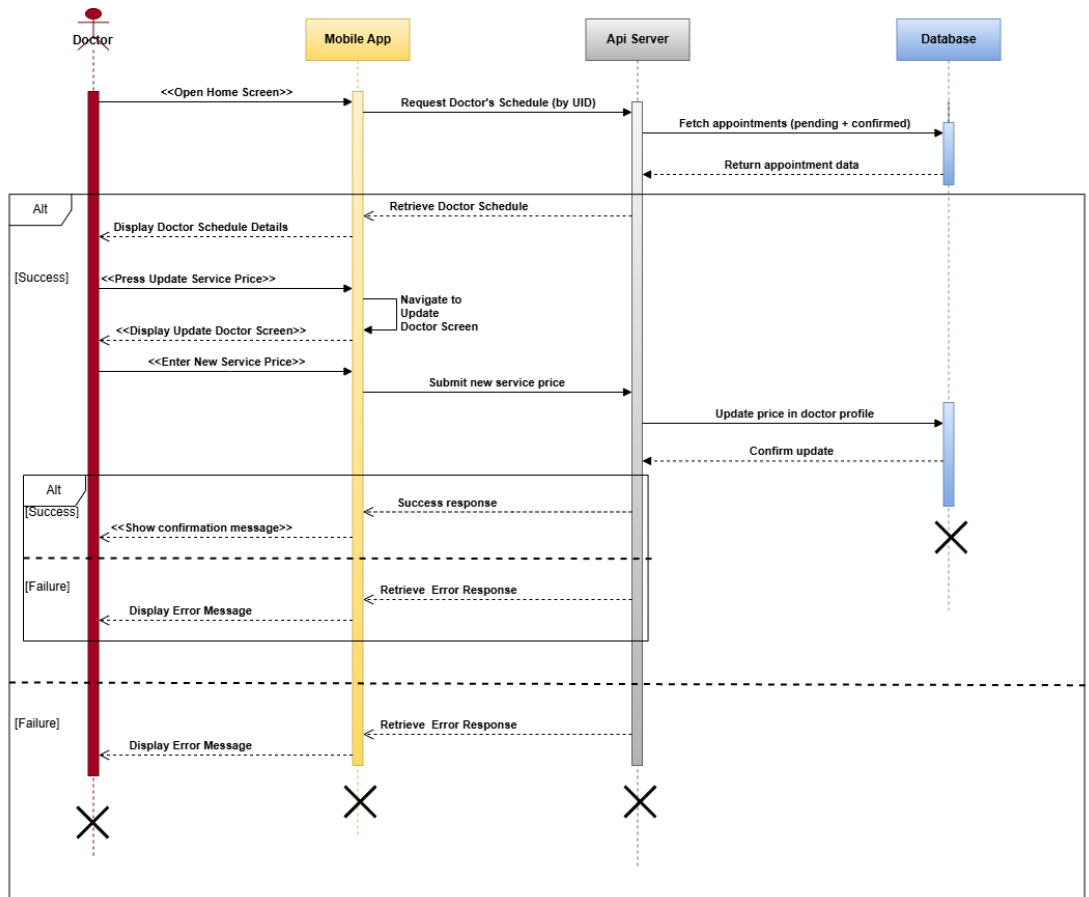


FIGURE 2.14.14 DOCTOR VIEW HIS SCHEDULE AND UPDATE HIS SERVICE PRICE SEQUENCE DIAGRAM

ICare - User Login Using Email & Password Activity Diagram

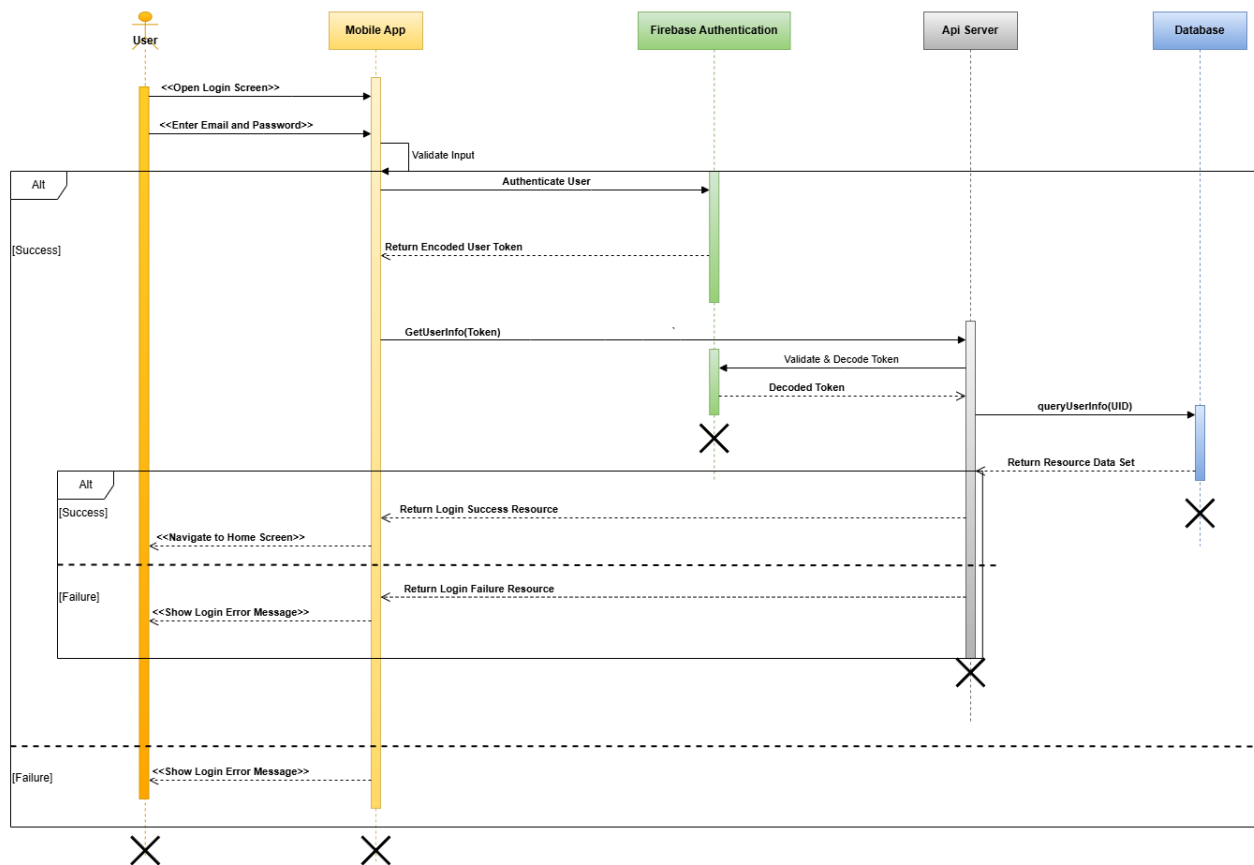


FIGURE 2.14.15 USER LOGIN USING EMAIL & PASSWORD ACTIVITY DIAGRAM

ICare—Patient Booking Appointment Sequence Diagram

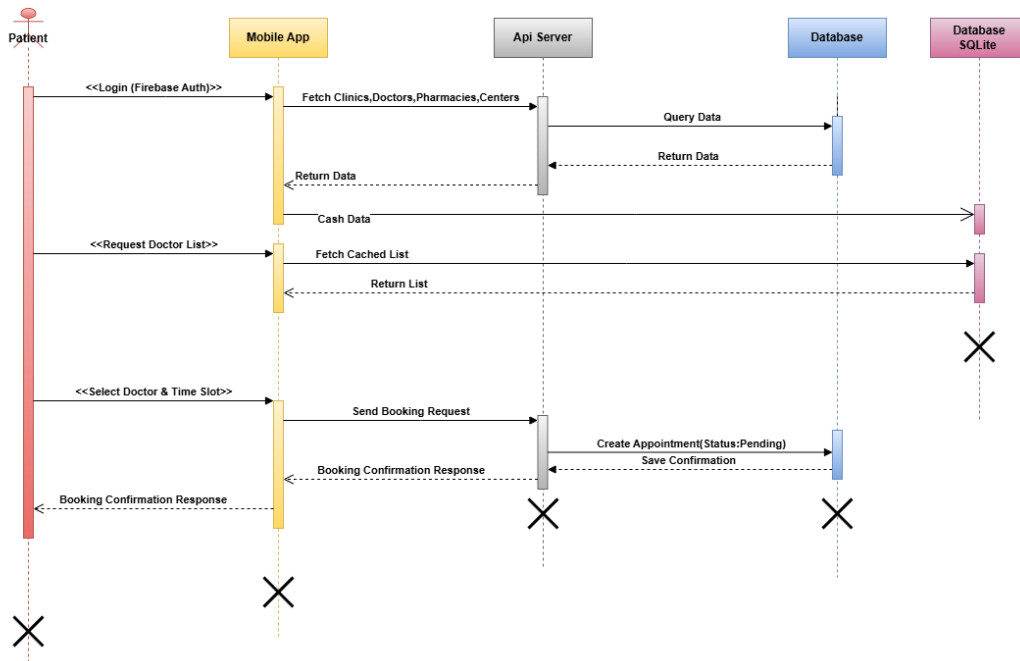


FIGURE 2.14.16 PATIENT BOOKING APPOINTMENT SEQUENCE DIAGRAM

ICare—Patient Canceling An Appointment Sequence Diagram

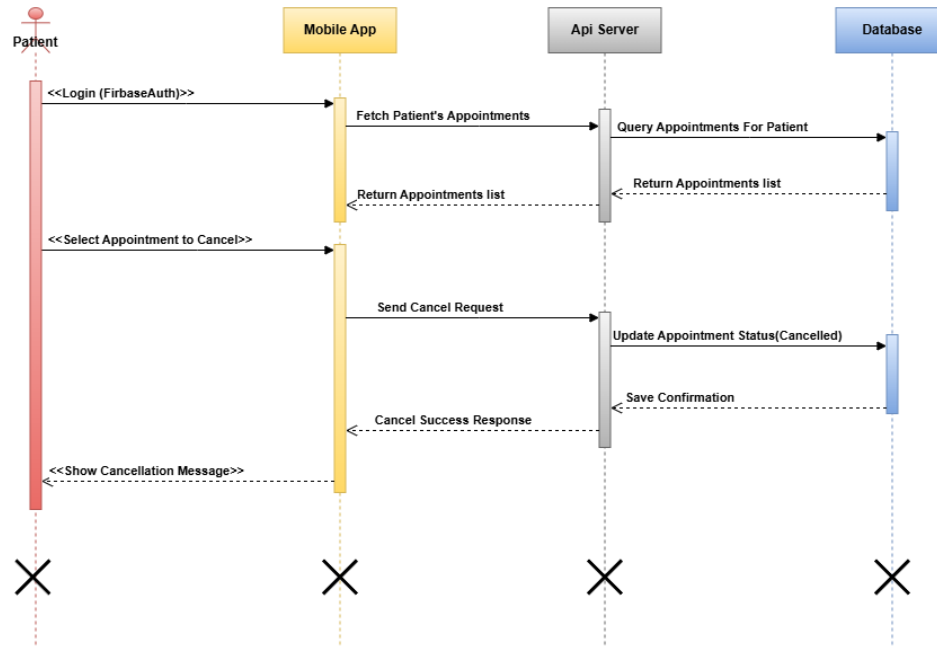


FIGURE 2.14.17 PATIENT CANCELING AN APPOINTMENT SEQUENCE DIAGRAM

ICare—Patient Register an Account Sequence Diagram

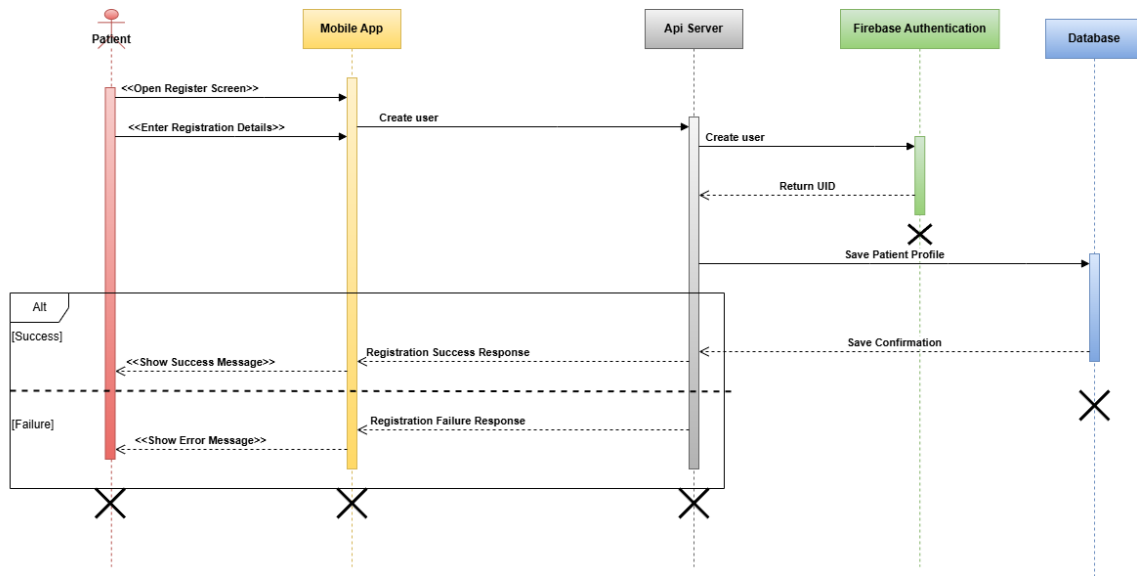


FIGURE 2.14.18 PATIENT REGISTER AN ACCOUNT SEQUENCE DIAGRAM

ICare—Patient Rescheduling An Appointment Sequence Diagram

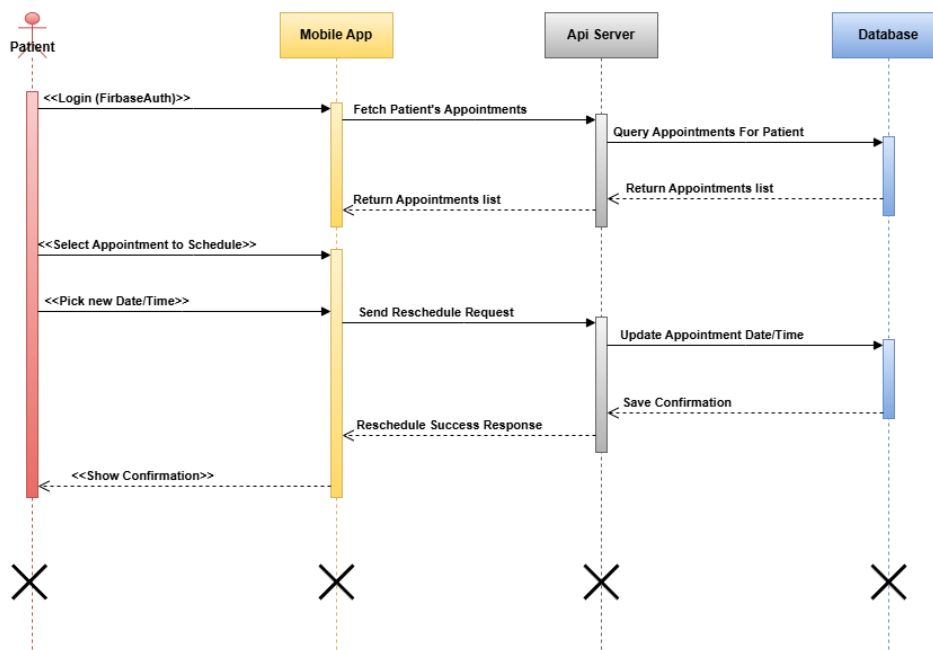


FIGURE 2.14.19 PATIENT RESCHEDULING AN APPOINTMENT SEQUENCE DIAGRAM

2.15. WIRE FRAMES PROTOTYPING

2.15.1. WIREFRAMES SCREENS SAMPLE

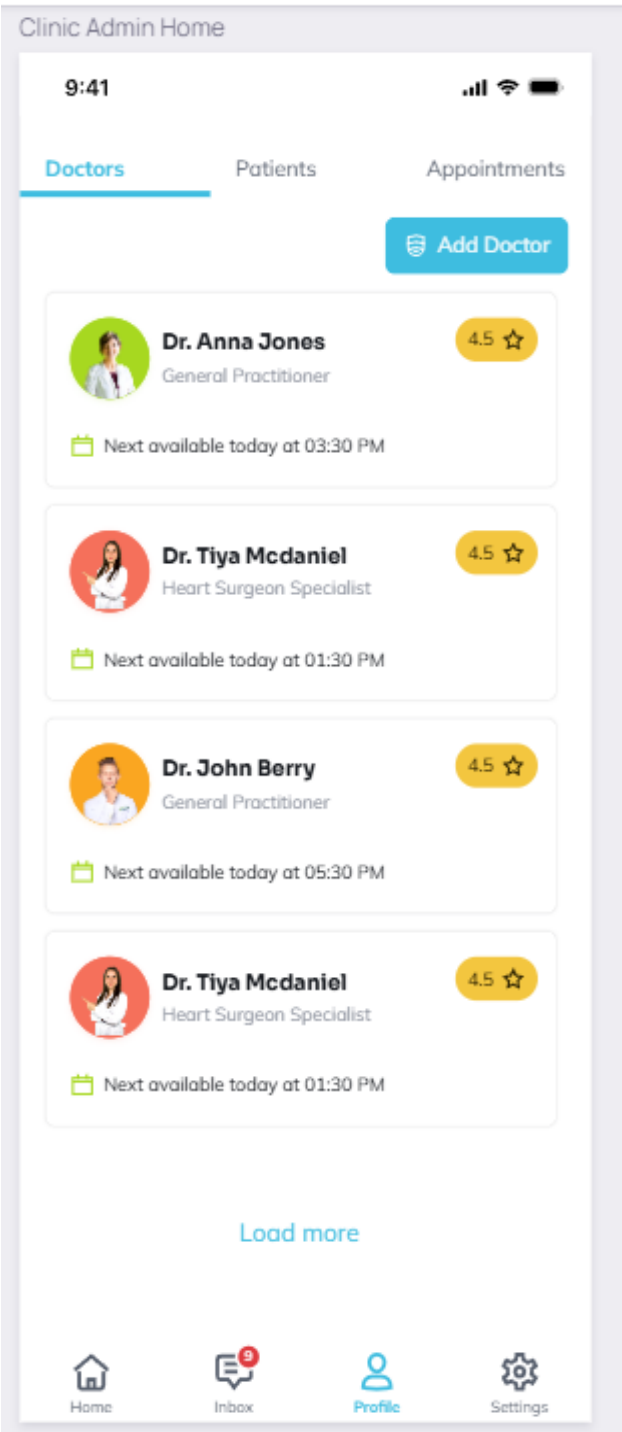


FIGURE 2.15.1.1 CLINIC ADMIN HOME

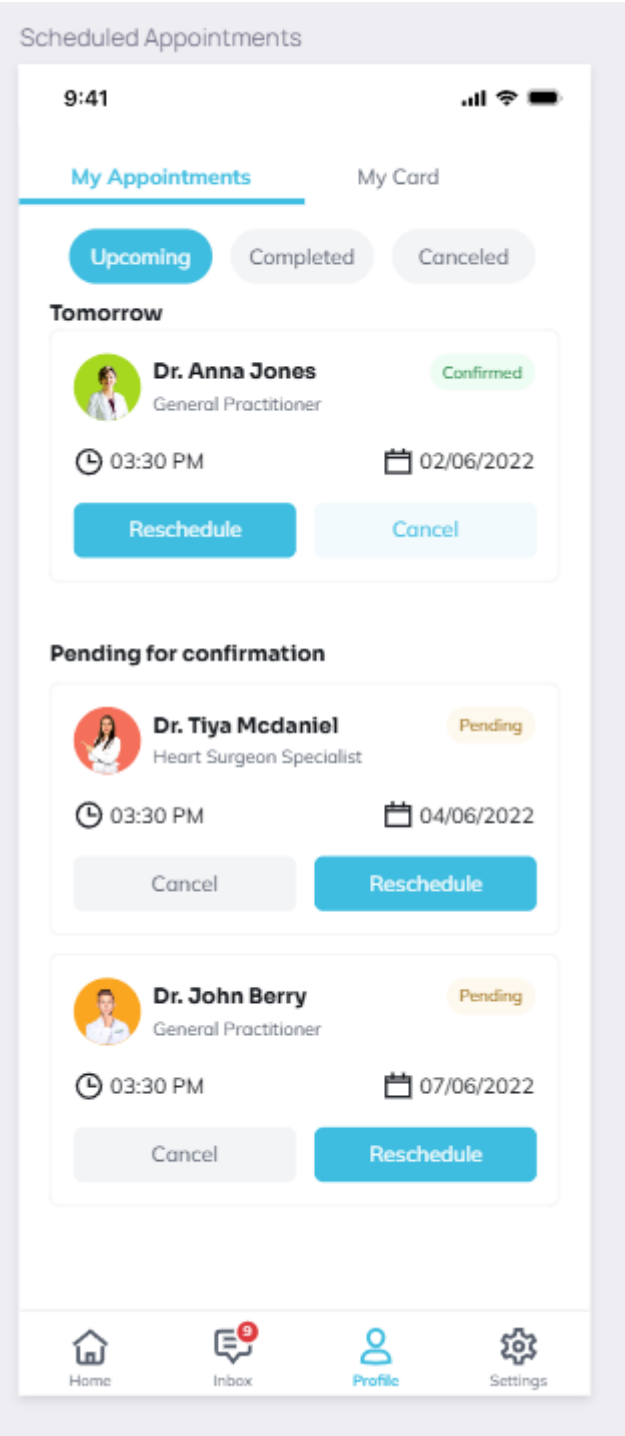


FIGURE 2.15.1.2 SCHEDULED APPOINTMENTS

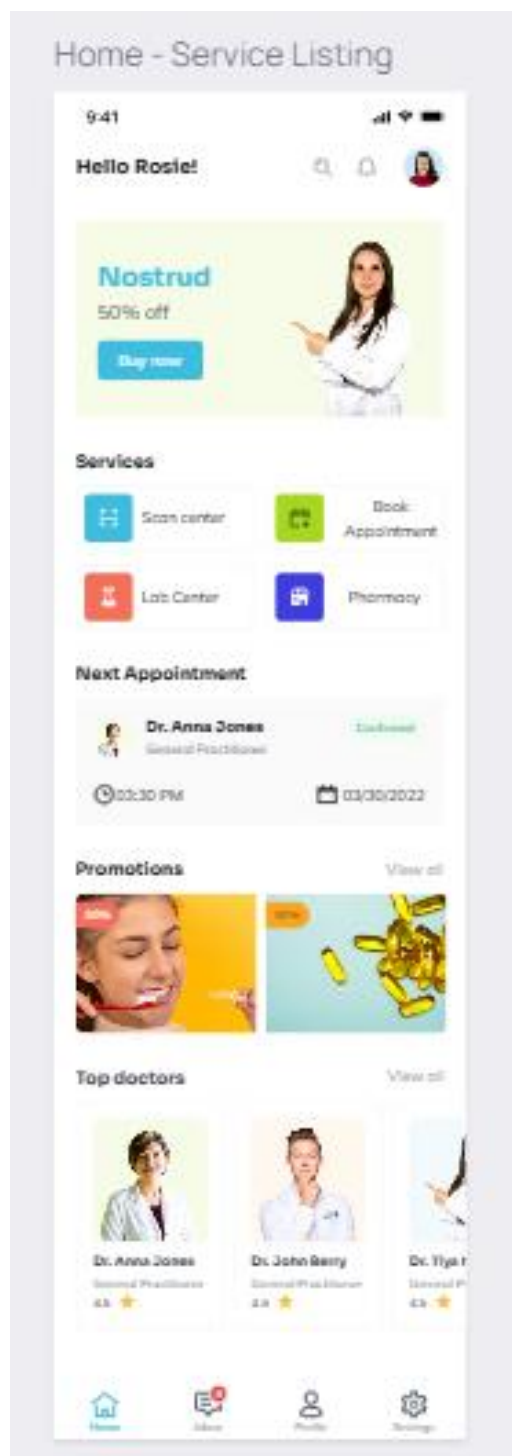


FIGURE 2.15.1.3 HOME SERVICE LISTING

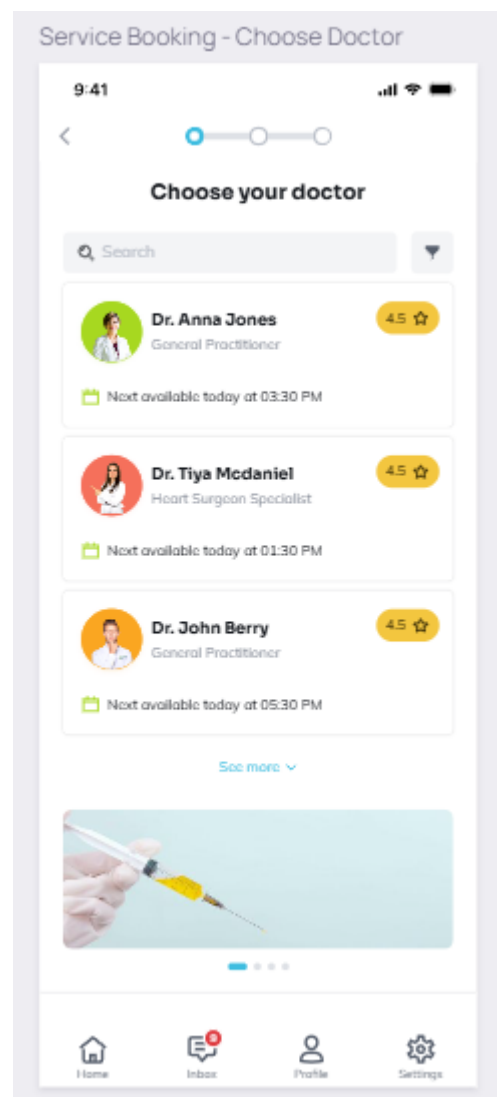


FIGURE 2.15.1.4 SERVICE BOOKING - CHOOSE DOCTOR

2.16. NON-FUNCTIONAL REQUIREMENTS

2.15.1 PERFORMANCE REQUIREMENTS:

- ✓ The system shall respond to user actions within 2 seconds for local operations, such as navigating screens and accessing cached data.
- ✓ The system shall respond within 5 seconds for network-based operations like fetching data from the backend when the network connection is stable.
- ✓ The system shall allow offline access to cached data, including clinics, doctors, pharmacies, and imaging centers, through SQLite to improve responsiveness when connectivity is poor.
- ✓ The system shall efficiently manage multiple user interactions, ensuring stability for up to 100 concurrent users during peak hours.

2.15.2 SECURITY REQUIREMENTS:

- ✓ The system shall enforce secure user authentication using Firebase Authentication, which supports both email/password and Google account integration.
- ✓ All communications between the mobile app and backend services shall be encrypted using HTTPS (SSL/TLS).
- ✓ Sensitive data, including patient personal details, appointment information, and medical consultations, shall be transmitted and stored securely, following best practices for Android and backend security.

2.15.3 AVAILABILITY & RELIABILITY REQUIREMENTS:

- ✓ The system shall maintain 99.5% uptime for backend services to ensure reliable availability of healthcare services.
- ✓ The system shall provide proper error handling and retry methods for network-dependent operations.
- ✓ If there is a network failure, the system shall allow read-only access to cached data until the connection is restored.

2.15.4 USABILITY REQUIREMENTS:

- ✓ The system shall offer a user-friendly Android interface with clear navigation that suits the specific role of the logged-in user, whether they are a Patient, Doctor, Clinic Staff, or Admin.
- ✓ All screens and forms shall follow consistent design guidelines to make them easy to use for individuals with different levels of digital literacy.
- ✓ Input forms shall give immediate feedback for incorrect or missing fields.

2.15.5 MAINTAINABILITY & SCALABILITY REQUIREMENTS:

- ✓ The system shall be designed using Clean Architecture and the MVVM pattern. This approach will keep different concerns separate and make it easier to improve in the future.
- ✓ The application shall be organized by feature, such as feature-auth, feature-home, and feature-appointments. This will help with maintenance and testing.
- ✓ The backend shall be developed with Spring Boot REST APIs. It will be scalable to support future growth in the number of healthcare providers and patients.

2.15.6 COMPATIBILITY REQUIREMENTS:

- ✓ The system shall be compatible with Android 6.0 (API level 23) and newer.
- ✓ The system shall work reliably on devices with various screen sizes and hardware specifications.

2.15.7 BACKUP & DATA INTEGRITY REQUIREMENTS:

- ✓ The system shall make regular backups of the backend database to prevent data loss.
- ✓ The system shall ensure data consistency and integrity when multiple user roles access the system at the same time, such as when one user schedules an appointment while another retrieves the same appointment list.

CHAPTER 3

DESIGN

3. CHAPTER 3

3.1. COMPONENT DIAGRAM

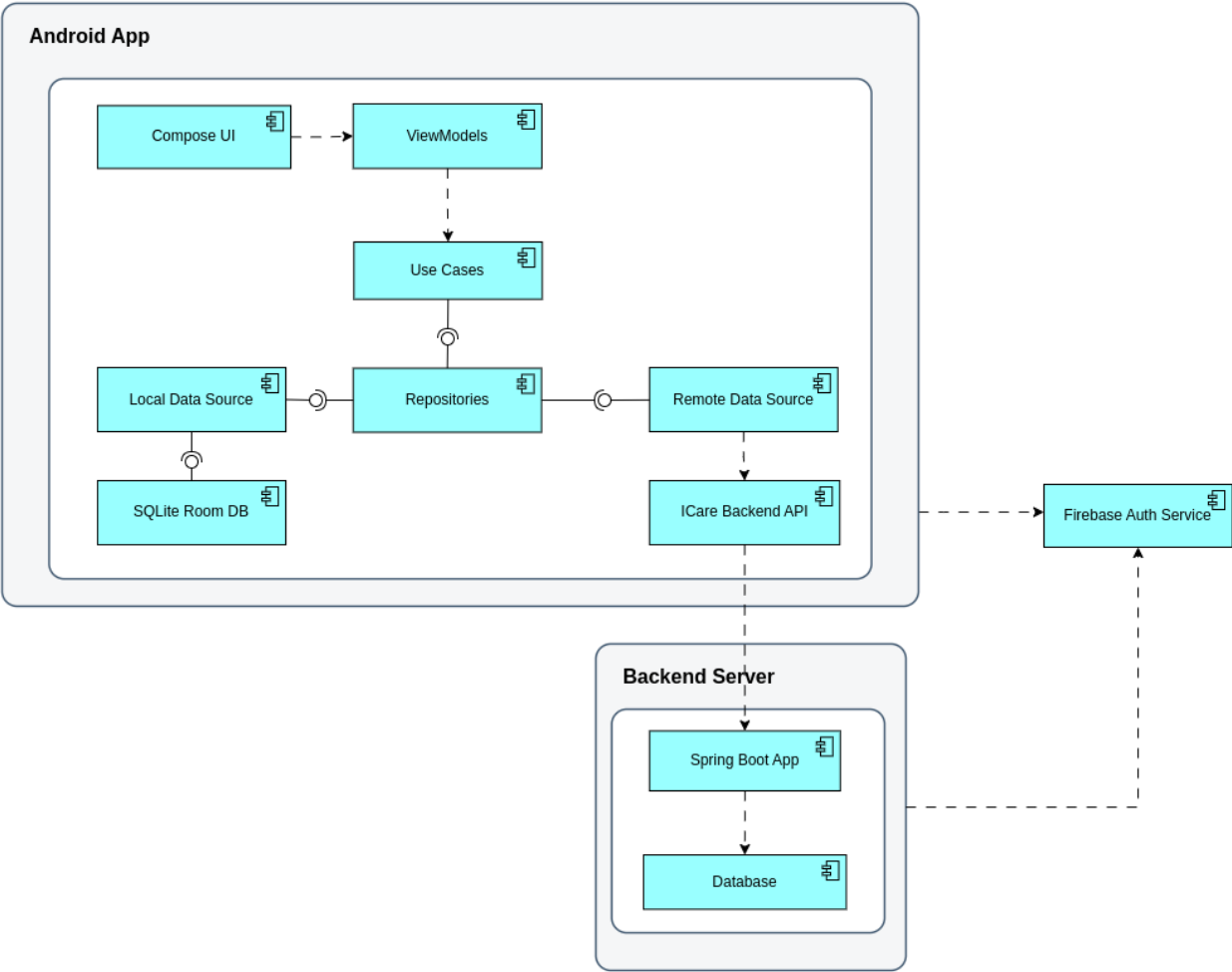


FIGURE 3.1.1 COMPONENT DIAGRAM

3.2. CLASS DIAGRAM

ICare Class Diagram

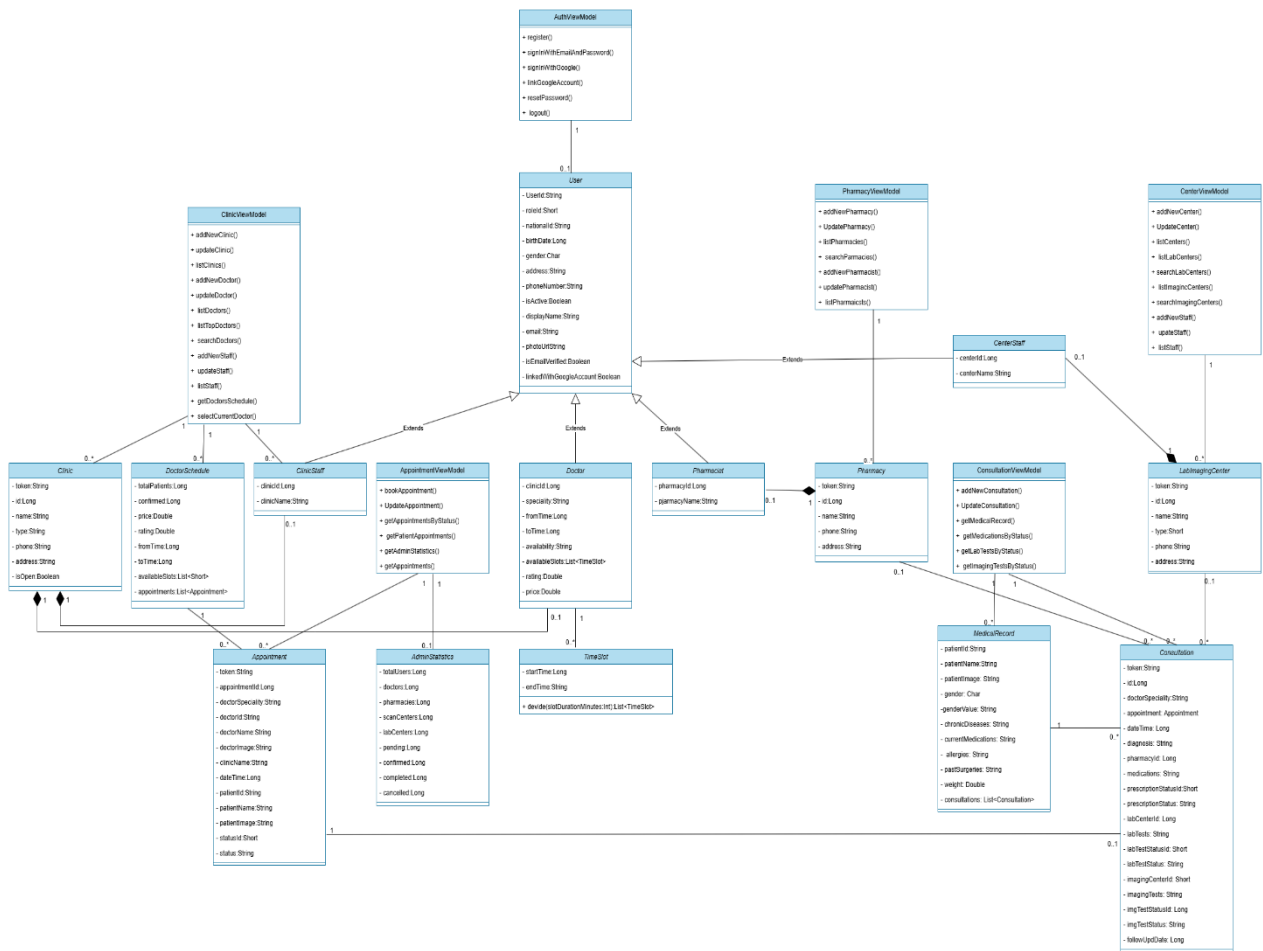


FIGURE 3.2.1 CLASS DIAGRAM

3.3. ENHANCED ENTITY RELATIONSHIP DIAGRAM (EERD)

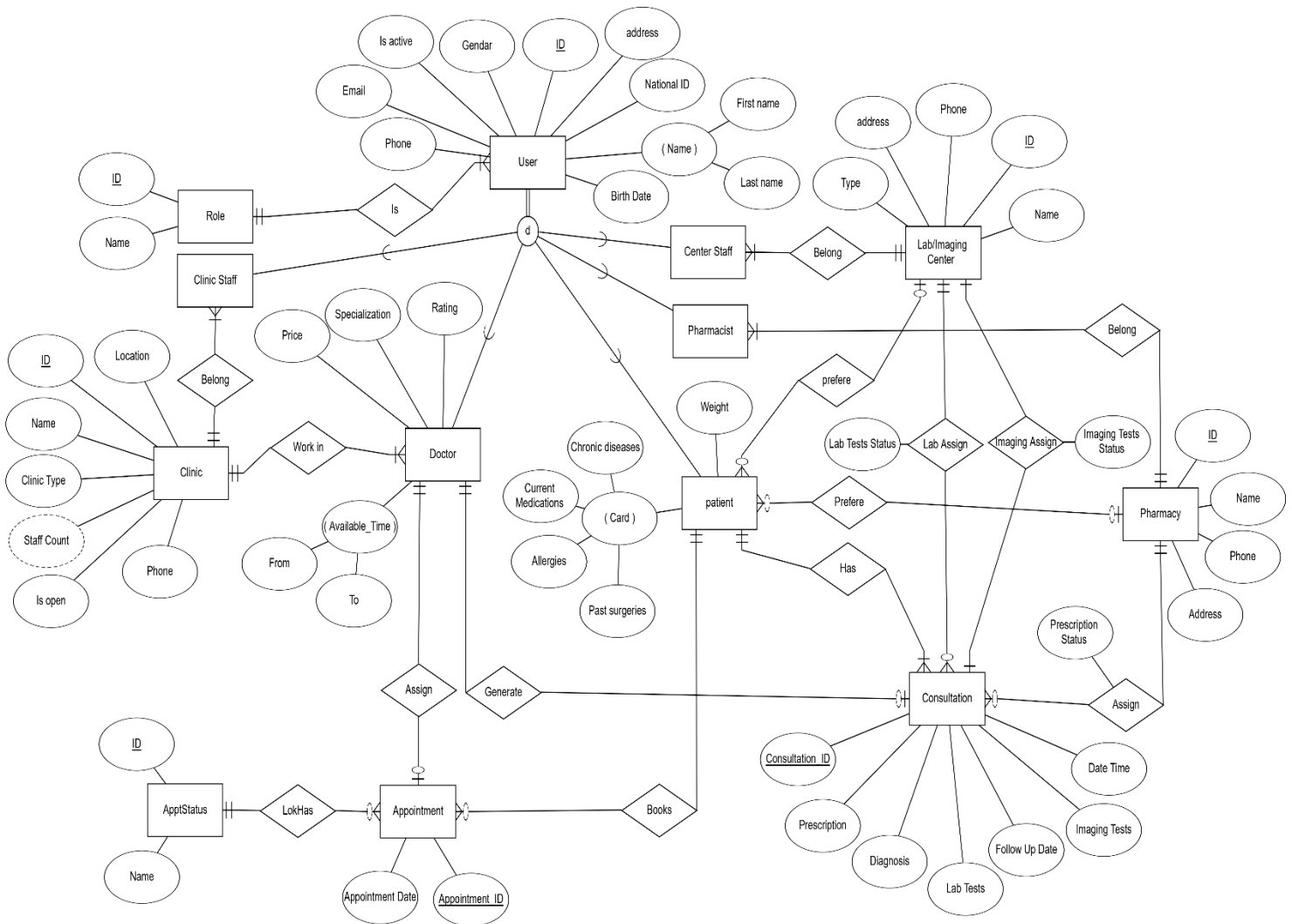


FIGURE 3.3.1 ENHANCED ENTITY RELATIONSHIP DIAGRAM (EERD)

3.4. MAPPING

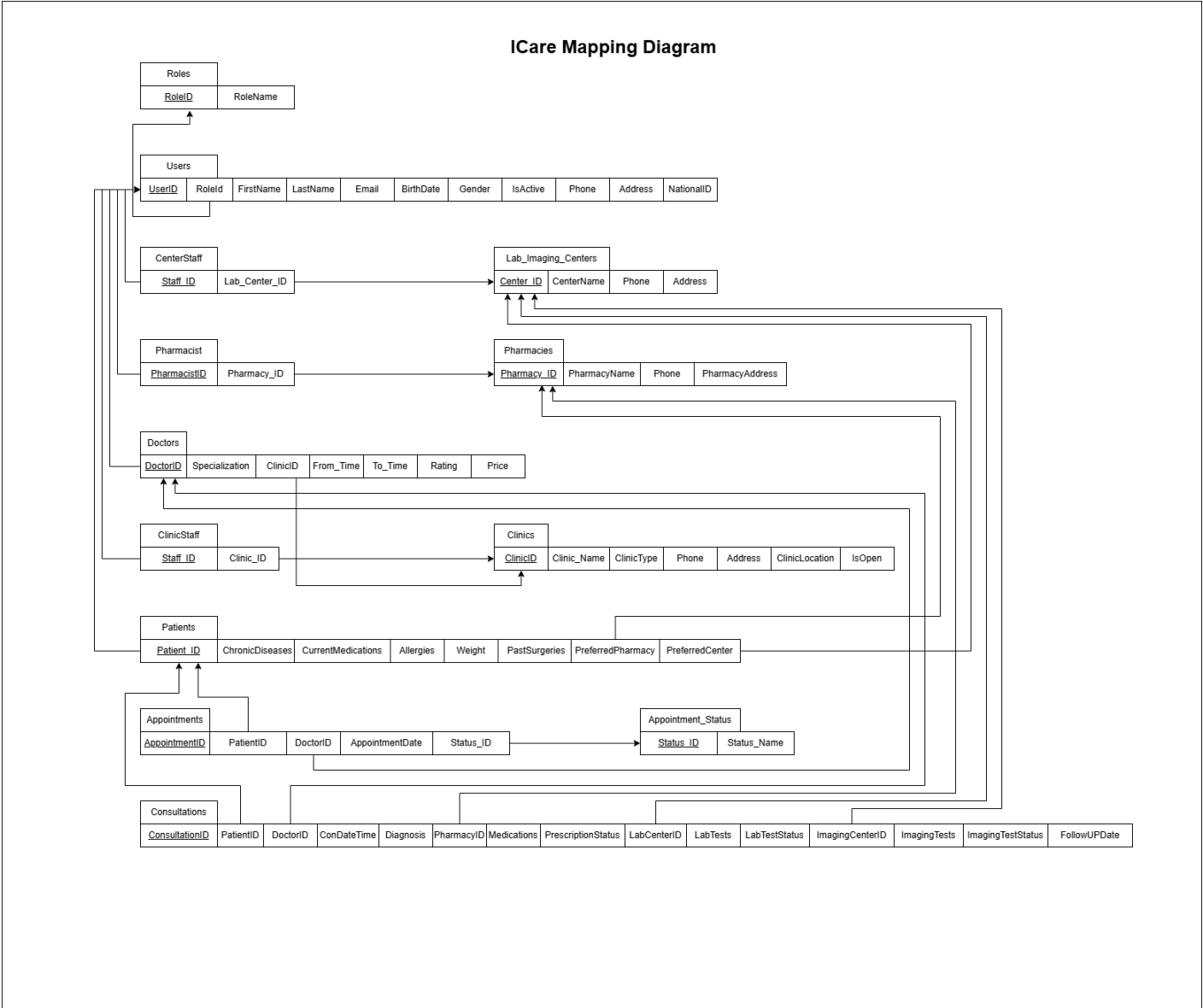


FIGURE 3.4.1 MAPPING SCHEME

CHAPTER 4

TESTING

4. CHAPTER 4

4.1. TESTING (USER ACCEPTANCE TEST)

4.1.1. TEST CASES

TABLE 4.1.1.1 TC-1

Test Case Identifier	TC-1
Use Case tested	UC-1
Pass /Fail criteria	The test is successful if a new patient creates an account with valid data and is taken to the login screen.
Input data	Name, Email, Password, Phone Number
Test Procedure	Expected Result
Step 1. Patient accesses the registration form.	System displays the form fields.
Step 2. Patient enters personal details: name, email, password, and phone number.	The system accepts the input.
Step 3. Patient submits the registration form.	The system validates the data (format, uniqueness, password policy).

TABLE 4.1.1.2 TC-2

Test Case Identifier	TC-2
Use Case tested	UC-2
Pass /Fail criteria	The test passes if the patient logs in with valid credentials and is directed to their home screen.
Input data	Registered Email, Password
Test Procedure	Expected Result
Step 1. Patient opens the ICare application.	The system displays the login screen.
Step 2. Patient enters a valid email and password.	The system accepts the input.
Step 3. Patient taps the Login button.	The system authenticates the credentials through Firebase Authentication.
Step 4. If the credentials are valid, the system allows access to the patient.	Patient is redirected to the patient's home screen.

TABLE 4.1.1.3 TC-3

Test Case Identifier	TC-3
Use Case tested	UC-3
Pass /Fail criteria	The test passes when the user successfully connects a Google account to their ICare account.
Input data	Valid Google account credentials
Test Procedure	Expected Result
Step 1. Log in via email/password.	User accesses profile screen.
Step 2. Tap "Link Google Account".	The system shows Google account list.
Step 3. Choose an account and confirm.	The system links Google account.
Step 4. Confirmation shown to user.	Link success message appears.

TABLE 4.1.1.4 TC-4

Test Case Identifier	TC-4
Use Case tested	UC-4
Pass /Fail criteria	The test passes when the user logs in with a linked Google account and accesses their dashboard.
Input data	Valid Google account credentials
Test Procedure	Expected Result
Step 1. Open login screen.	The system displays a login screen.
Step 2. Select Google login option.	Google account list appears.
Step 3. Select a valid linked account.	System validates and logs in.
Step 4. User accesses the dashboard.	User lands on appropriate dashboard screen.

TABLE 4.1.1.5 TC-5

Test Case Identifier	TC-5
Use Case tested	UC-5
Pass /Fail criteria	The test is successful if the system logs out the user and takes them to the login screen.
Input data	None
Test Procedure	Expected Result
Step 1. User taps Logout from the profile screen.	The system logs out the user and exits the app.

TABLE 4.1.1.6 TC-6

Test Case Identifier	TC-6
Use Case tested	UC-6
Pass /Fail criteria	The test is successful if the system retrieves and shows the list of clinics after logging in.
Input data	Valid login credentials
Test Procedure	Expected Result
Step 1. User logs into ICare app.	The system authenticates user.
Step 2. The system retrieves clinics from the backend.	The clinics list returned successfully.
Step 3. Clinics list displayed to user.	User sees the full list in UI.

TABLE 4.1.1.7 TC-7

Test Case Identifier	TC-7
Use Case tested	UC-7
Pass /Fail criteria	The test passes if the system retrieves and shows the doctors' list successfully after logging in.
Input data	Valid login credentials
Test Procedure	Expected Result
Step 1. User logs into ICare app.	The system authenticates user.
Step 2. The system retrieves doctors from the backend.	Doctors list returned successfully.
Step 3. The doctors list displayed to user.	User sees the full list in UI.

TABLE 4.1.1.8 TC-8

Test Case Identifier	TC-8
Use Case tested	UC-8
Pass /Fail criteria	The test passes if the patient books an appointment and gets confirmation.
Input data	Patient details, selected doctor, date, time
Test Procedure	Expected Result
Step 1. The patient selects the doctor and date.	System displays available times.
Step 2. The patient chooses a time slot and confirms the booking.	System validates and books appointment.
Step 3. System confirms booking.	Confirmation message displayed to patient.

TABLE 4.1.1.9 TC-9

Test Case Identifier	TC-9
Use Case tested	UC-9
Pass /Fail criteria	The test passes if the patient successfully reschedules an existing appointment.
Input data	Existing appointment ID, new date and time
Test Procedure	Expected Result
Step 1. Patient selects existing appointment.	Appointment details displayed.
Step 2. Patient selects new date and time.	System checks availability.
Step 3. Patient confirms reschedule.	System updates appointment and confirms.

TABLE 4.1.1.10 TC-10

Test Case Identifier	TC-10
Use Case tested	UC-10
Pass /Fail criteria	The test passes if the patient successfully cancels an existing appointment.
Input data	Appointment ID
Test Procedure	Expected Result
Step 1. Patient selects appointment from the list.	Appointment details displayed.
Step 2. Patient clicks Cancel option.	System prompts for confirmation.
Step 3. Patient confirms cancellation.	System removes appointment and confirms to patient.

TABLE 4.1.1.11 TC-11

Test Case Identifier	TC-11
Use Case tested	UC-11
Pass /Fail criteria	The test is successful if the patient can view a list of top doctors
Input data	None
Test Procedure	Expected Result
Step 1. Patient navigates to the home screen.	The system displays home screen content.
Step 2. Patient scrolls down to the Top Doctors section at the bottom.	The system fetches top-rated doctors.
Step 3. The system displays top doctors list.	A list of top-rated doctors appears.

TABLE 4.1.1.12 TC-12

Test Case Identifier	TC-12
Use Case tested	UC-12
Pass /Fail criteria	The test passes if the patient can view a list of contracted pharmacies.
Input data	None
Test Procedure	Expected Result
Step 1. Patient goes to the Pharmacies section.	System displays available pharmacies.
Step 2. System fetches contracted pharmacies list.	Contracted pharmacies data returned.
Step 3. System displays list to the patient.	List appears with pharmacy names and contact info.

TABLE 4.1.1.13 TC-13

Test Case Identifier	TC-13
Use Case tested	UC-13
Pass /Fail criteria	The test passes if the patient can view a list of contracted lab centers.
Input data	None
Test Procedure	Expected Result
Step 1. Patient goes to Lab Centers section.	The system displays lab center options.
Step 2. The system retrieves contracted lab centers.	Lab centers data returned.
Step 3. System displays lab centers list.	List appears with center names and details.

TABLE 4.1.1.14 TC-14

Test Case Identifier	TC-14
Use Case tested	UC-14
Pass /Fail criteria	The test is considered a pass if the patient can see a list of imaging (scanning) centers.
Input data	None
Test Procedure	Expected Result
Step 1. Patient goes to the Imaging Centers section.	The system displays imaging center options.
Step 2. The system retrieves contracted imaging centers.	Imaging centers data returned.
Step 3. The system shows the list of imaging centers.	List appears with center names and details.

TABLE 4.1.1.15 TC-15

Test Case Identifier	TC-15
Use Case tested	UC-15
Pass /Fail criteria	The test passes if clinical staff can see a list of pending and confirmed appointments.
Input data	Valid clinical staff login
Test Procedure	Expected Result
Step 1. Clinical staff member logs in.	The system authenticates staff member.
Step 2. The staff member goes to the Appointments section.	The system displays an appointment list.
Step 3. The system retrieves and displays pending and confirmed appointments.	Appointments categorized and listed correctly.

TABLE 4.1.1.16 TC-16

Test Case Identifier	TC-16
Use Case tested	UC-16
Pass /Fail criteria	The test is successful if the clinical staff confirms a pending appointment.
Input data	Appointment ID
Test Procedure	Expected Result
Step 1. Clinical staff logs in.	System authenticates staff member.
Step 2. Staff goes to the Appointments section.	Pending and confirmed appointments listed.
Step 3. Staff selects a pending appointment and clicks Confirm.	System updates status to Confirmed.
Step 4. Confirmation message displayed.	Appointment marked confirmed in the system.

TABLE 4.1.1.17 TC-17

Test Case Identifier	TC-17
Use Case tested	UC-17
Pass /Fail criteria	The test passes if the doctor can view their own appointment schedule.
Input data	Valid doctor login
Test Procedure	Expected Result
Step 1. Doctor logs in.	The system authenticates doctor.
Step 2. The doctor goes to My Appointments.	The system fetches the doctor's appointment list.
Step 3. The appointments list is shown and categorized by status.	Correct appointments appear in list view.

TABLE 4.1.1.18 TC-18

Test Case Identifier	TC-18
Use Case tested	UC-18
Pass /Fail criteria	The test passes if the doctor successfully updates their service price.
Input data	New service price
Test Procedure	Expected Result
Step 1. Doctor logs in and navigates to Profile Settings.	System loads profile info.
Step 2. Doctor updates service price and taps Save.	System validates and updates price.
Step 3. Confirmation message displayed.	New price reflected in the system.

TABLE 4.1.1.19 TC-19

Test Case Identifier	TC-19
Use Case tested	UC-19
Pass /Fail criteria	The test passes if the doctor successfully adds a consultation for a finished appointment.
Input data	Appointment ID, consultation notes
Test Procedure	Expected Result
Step 1. The doctor logs in and chooses a finished appointment.	Appointment details displayed.
Step 2. Doctor adds consultation notes and submits.	System saves consultation details.
Step 3. Confirmation message displayed.	Consultation linked to appointment successfully.

TABLE 4.1.1.20 TC-20

Test Case Identifier	TC-20
Use Case tested	UC-20
Pass /Fail criteria	The test passes if the admin can view the list of registered clinics.
Input data	Valid admin login
Test Procedure	Expected Result
Step 1. Admin logs in.	System authenticates admin.
Step 2. The admin goes to Clinics Management.	Clinics list retrieved from backend.
Step 3. Clinics list displayed with details.	List appears with correct clinic data.

TABLE 4.1.1.21 TC-21

Test Case Identifier	TC-21
Use Case tested	UC-21
Pass /Fail criteria	The test succeeds if the admin can successfully add a new clinic.
Input data	Clinic Name, Location, Contact Info
Test Procedure	Expected Result
Step 1. The admin logs in and goes to Clinics Management.	Clinics management page opens.
Step 2. The admin selects the Add New Clinic option.	Add clinic form displayed.
Step 3. The admin fills in the clinic details and submits.	System validates and adds the clinic.
Step 4. Confirmation message displayed.	The new clinic appears in the clinics list.

TABLE 4.1.1.22 TC-22

Test Case Identifier	TC-22
Use Case tested	UC-22
Pass /Fail criteria	The test passes if the admin can successfully update the information of an existing clinic.
Input data	Updated clinic details
Test Procedure	Expected Result
Step 1. Admin navigates to Clinics Management.	Clinics list displayed.
Step 2. Admin selects a clinic to update.	Clinic details form opens.
Step 3. Admin edits details and submits.	The system validates and updates the clinic.
Step 4. Confirmation message displayed.	Updated info reflected in clinics list.

TABLE 4.1.1.23 TC-23

Test Case Identifier	TC-23
Use Case tested	UC-23
Pass /Fail criteria	The test passes if the admin can view the list of registered doctors.
Input data	None
Test Procedure	Expected Result
Step 1. Admin logs in and navigates to Doctors Management.	List of registered doctors displayed.

TABLE 4.1.1.24 TC-24

Test Case Identifier	TC-24
Use Case tested	UC-24
Pass /Fail criteria	The test passes if the admin can add a new doctor successfully.
Input data	Doctor Name, Specialization, Contact Info
Test Procedure	Expected Result
Step 1. Admin selects Add New Doctor.	Add doctor form appears.
Step 2. Admin fills in doctor details and submits.	System validates and adds the doctor.
Step 3. Confirmation message displayed.	The new doctor appears in the doctors list.

TABLE 4.1.1.25 TC-25

Test Case Identifier	TC-25
Use Case tested	UC-25
Pass /Fail criteria	The test passes if the admin can successfully update a doctor's information.
Input data	Updated doctor details
Test Procedure	Expected Result
Step 1. Admin selects a doctor to update.	Doctor's profile form displayed.
Step 2. Admin updates details and submits.	System validates and updates info.
Step 3. Confirmation message displayed.	Updated info appears in doctors list.

TABLE 4.1.1.26 TC-26

Test Case Identifier	TC-26
Use Case tested	UC-26
Pass /Fail criteria	The test is successful if the admin can view the list of clinic staff members.
Input data	None
Test Procedure	Expected Result
Step 1. Admin logs in and navigates to Clinic Staff Management.	System displays clinic staff list.

TABLE 4.1.1.27 TC-27

Test Case Identifier	TC-27
Use Case tested	UC-27
Pass /Fail criteria	The test is successful if the admin can add a new clinic staff member.
Input data	Name, Email, Phone, Assigned Clinic
Test Procedure	Expected Result
Step 1. Admin selects Add New Clinic Staff.	Add staff form displayed.
Step 2. The admin fills in the staff details and submits.	The system validates and adds staff.
Step 3. Confirmation message displayed.	The new staff member is listed among the staff members.

TABLE 4.1.1.28 TC-28

Test Case Identifier	TC-28
Use Case tested	UC-28
Pass /Fail criteria	The test is successful if the admin can update the information of a current clinic staff member.
Input data	Updated staff details
Test Procedure	Expected Result
Step 1. Admin selects a staff member to update.	Staff details form displayed.
Step 2. Admin edits details and submits.	The system validates and updates staff info.
Step 3. Confirmation message displayed.	Updated info reflected in staff list.

TABLE 4.1.1.29 TC-29

Test Case Identifier	TC-29
Use Case tested	UC-29
Pass /Fail criteria	The test passes if the admin can view the list of pharmacies successfully.
Input data	None
Test Procedure	Expected Result
Step 1. The admin logs in and goes to Pharmacy Management.	List of registered pharmacies displayed.

TABLE 4.1.1.30 TC-30

Test Case Identifier	TC-30
Use Case tested	UC-30
Pass /Fail criteria	The test is successful if the clinical staff confirms a pending appointment.
Input data	Appointment ID
Test Procedure	Expected Result
Step 1. Clinical staff member logs in.	System authenticates staff.
Step 2. Staff member goes to Appointments.	Appointment list displayed.
Step 3. The staff selects a pending appointment and clicks Confirm.	System updates status to Confirmed and displays a confirmation message.

TABLE 4.1.1.31 TC-31

Test Case Identifier	TC-31
Use Case tested	UC-31
Pass /Fail criteria	The test is successful if the doctor adds a consultation for a completed appointment.
Input data	Appointment ID, consultation notes
Test Procedure	Expected Result
Step 1. The doctor logs in and goes to My Appointments.	Appointment list displayed.
Step 2. The doctor chooses a finished appointment.	Appointment details open.
Step 3. The doctor writes down the consultation notes and submits them.	System saves notes and confirms addition.

TABLE 4.1.1.32 TC-32

Test Case Identifier	TC-32
Use Case tested	UC-32
Pass /Fail criteria	The test passes if the admin can view the list of pharmacists successfully.
Input data	None
Test Procedure	Expected Result
Step 1. Admin logs in and navigates to Pharmacist Management.	List of pharmacists displayed.

TABLE 4.1.1.33 TC-33

Test Case Identifier	TC-33
Use Case tested	UC-33
Pass /Fail criteria	The test is successful if the admin can add a new pharmacist.
Input data	Name, Email, Phone, Assigned Pharmacy
Test Procedure	Expected Result
Step 1. Admin selects Add New Pharmacist.	Add the pharmacist form displayed.
Step 2. Admin fills in details and submits.	The system validates and adds pharmacist.
Step 3. Confirmation message displayed.	The new pharmacist appears in the list.

TABLE 4.1.1.34 TC-34

Test Case Identifier	TC-34
Use Case tested	UC-34
Pass /Fail criteria	The test passes if the admin can successfully update a pharmacist's information.
Input data	Updated pharmacist details
Test Procedure	Expected Result
Step 1. Admin selects a pharmacist to update.	Pharmacist details form displayed.
Step 2. Admin edits details and submits.	System validates and updates info.
Step 3. Confirmation message displayed.	Updated pharmacist info shown in list.

TABLE 4.1.1.35 TC-35

Test Case Identifier	TC-35
Use Case tested	UC-35
Pass /Fail criteria	The test passes if the admin can view the list of lab/imaging centers successfully.
Input data	None
Test Procedure	Expected Result
Step 1. The admin goes to Lab/Imaging Center Management.	List of centers displayed.

TABLE 4.1.1.36 TC-36

Test Case Identifier	TC-36
Use Case tested	UC-36
Pass /Fail criteria	The test is successful if the admin can add a new lab or imaging center.
Input data	Center Name, Type, Contact Info
Test Procedure	Expected Result
Step 1. Admin selects Add New Center.	Add center form displayed.
Step 2. Admin fills in details and submits.	The system validates and adds center.
Step 3. Confirmation message displayed.	The new center appears in list.

TABLE 4.1.1.37 TC-37

Test Case Identifier	TC-37
Use Case tested	UC-37
Pass /Fail criteria	The test passes if the admin can update the details of an existing lab or imaging center.
Input data	Updated center details
Test Procedure	Expected Result
Step 1. Admin selects a center to update.	Center details form displayed.
Step 2. Admin updates details and submits.	System validates and updates center info.
Step 3. Confirmation message displayed.	Updated center info reflected in list.

TABLE 4.1.1.38 TC-38

Test Case Identifier	TC-38
Use Case tested	UC-38
Pass /Fail criteria	The test passes if the admin can successfully view the list of center staff.
Input data	None
<u>Test Procedure</u>	<u>Expected Result</u>
Step 1. Admin goes to Center Staff Management.	List of staff members displayed.

TABLE 4.1.1.39 TC-39

Test Case Identifier	TC-39
Use Case tested	UC-39
Pass /Fail criteria	The test passes if the admin can add a new center staff member successfully.
Input data	Name, Email, Phone, Assigned Center
<u>Test Procedure</u>	<u>Expected Result</u>
Step 1. Admin selects Add New Center Staff.	Add staff form displayed.
Step 2. Admin fills in details and submits.	The system validates and adds staff member.
Step 3. Confirmation message displayed.	The new staff member appears on the staff list.

TABLE 4.1.1.40 TC-40

Test Case Identifier	TC-40
Use Case tested	UC-40
Pass /Fail criteria	The test passes if the admin can update the information of an existing center staff member.
Input data	Updated staff details
<u>Test Procedure</u>	<u>Expected Result</u>
Step 1. The admin chooses a staff member to update.	Staff details form displayed.
Step 2. Admin updates details and submits.	System validates and updates info.
Step 3. Confirmation message displayed.	Updated staff info shown in list.

5. FUTURE WORK

5.1 Insurance companies' integration

Enhance compatibility with insurance providers to streamline claims processing and coverage verification.

5.2. Telemedicine Integration

Virtual Consultations: Support video consultations for patients who are unable to attend in-person consultations due to distance or other constraints.

6. REFERENCES

- [1] <https://kotlinlang.org>
- [2] <https://www.w3schools.com/sql>
- [3] <https://www.microsoft.com/en-us/sql-server/sql-server-2017>
- [4] <https://spring.io/projects/spring-boot>
- [5] <https://developer.android.com/guide>
- [6] <https://firebase.google.com/docs/auth>
- [7] <https://square.github.io/retrofit>
- [8] <https://developer.android.com/training/data-storage/room>
- [9] <https://insert-koin.io>
- [10] <https://github.com>
- [11] <https://www.jetbrains.com/idea>
- [12] <https://developer.android.com/studio>
- [13] <https://www.atlassian.com/software/jira>
- [14] <https://www.microsoft.com/en-us/microsoft-365/microsoft-office>
- [15] <https://app.visily.ai/projects/22934966-00df-4aee-a82b-bd408cb6be94/boards/1601159>