MediCare

LEVEL_4 [2025]

CS Department

Supervised by :

Dr. Seham Elaw Amer

Project Team Members

Name	Role	Department
Abdelrahman Abdel Samee	Back-End	CS
Mohamed Ahmed Ali	Back-End	CS
Nehal Tarek	Back-End	CS
Nada Mohamed	Front-End	CS
Mayada Mohamed	Front-End	CS
Abdallah Adel	UI/UX	CS
Ahmed Mansour	Mobile Developer	CS
Ali Maher Mohamed	Mobile Developer	CS

Contents

Figures

List of Figures

Tables

List of Tables

Abbreviations

Symbol	Expression
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
ERD	Entity Relation Diagram
DFD	Data Flow Diagram
SDLC	Software Development Life cycle

Acknowledgment

First and foremost, we would express our deepest gratitude and appreciation to our Advisor Dr. Seham for her support, outstanding guidance, and encouragement throughout our graduation project.

As well as Engineer. Dr.Elnomery Zanaty for his continuous support and help as he was always there for us providing more than the needed time and effort from him since the very beginning of our project.

We would like to thank our families, especially our parents. We hope they are proud of us in our last year of education, we hope that we will start giving back to the community very soon! Thank you for supporting us and providing us with the needed time, eort, encouragement, patience, and assistance over the years.

Finally, our faculty for providing us with the courses that guided us in the right direction in our lives and the help of all the professors that left a great impact on our lives,

Thank you.

Chapter 1: Introduction

In this chapter, we are going to discuss and go deeper into the overview of the project and know more about its scope and limitations and explain some terminologies we will find throughout the document.

Introduction

MediCare, is an application and website designed to enhance the healthcare experience by integrating a variety of medical services into a single, comprehensive platform. The application allows users to easily book doctor appointments, view their specializations, and access a transparent rating system that helps them make well-informed decisions. The project prioritizes credibility, ensuring that all information about doctors is verified for a secure and reliable experience. By leveraging modern technologies and a streamlined design, MediCare provides an innovative solution that meets user needs and improves the quality of healthcare services.

Chapter 2: Problem Definition

In this chapter, we will explore the core problem that the project aims to address.
We will delve into the challenges and issues faced in medical field that led to the development of this application. .

Problem Definition

Accessing reliable healthcare services remains a challenge for many individuals due to fragmented systems, lack of verified information, and limited accessibility to medical professionals. Patients often struggle to find accurate information about doctors, such as their specializations, availability, or credibility. Furthermore, there is no centralized platform that offers a seamless and trustworthy way to book appointments, check doctor ratings, and manage healthcare needs efficiently. These challenges can lead to delays in treatment, frustration for patients, and inefficiencies in the healthcare system.

Issues and Objectives

Issue		Weight
1.	Lack of a unified platform that consolidates essential healthcare features like booking and reviews.	10
2.	Difficulty in accessing verified and accurate information about doctors	9
3.	Limited tools for efficient and convenient healthcare management.	8

Objectives	
1.	Ensure all information displayed about doctors is verified and credible to enhance user trust.
2.	Implement a transparent and user-friendly rating system to help patients make informed decisions.
3.	Provide an intuitive, technology-driven solution that streamlines healthcare access and improves user satisfaction.

Requirements and constrains

Requirements

1. Obtaining Official Licenses from Doctors and Pharmacies
2. Collaborate with pharmacies to provide a legitimate list of medicines and services..

Constraints

1. Ensure compatibility with major operating systems (Windows, macOS, Android, iOS) and browsers
2. Manual Verification: Doctor credential verification may require manual processes, leading to potential delays.
3. Internet Dependence: The system requires reliable internet connectivity..
4. Medical Expertise: Limited availability of medical professionals for consultation, feedback, and testing..

Chapter 3:Project Analysis

In this section,

we will analyze the proposed solution in terms of its feasibility, objectives, and expected outcomes.

We will examine the requirements of the project

and evaluate the different approaches

Analysis

Project Analysis

➢ What Is Agile?

The Agile methodology is a project management approach that involves breaking the project into phases and emphasizes continuous collaboration and improvement.

Teams follow a cycle of planning, executing, and evaluating.

➢ Why Choose Agile?

Using Agile in Our project, especially in software development, can offer several benefits that align with the dynamic nature of modern projects, especially in the context of our MediCare application. Here's why Agile would be a great fit:

1. Flexibility and Adaptability

- Rapid Changes: In healthcare, requirements can change quickly, whether due to regulatory changes, new technologies, or user feedback. Agile allows you to adapt to these changes without derailing the entire project.
- Iterative Improvements: You can continuously improve the features of the MediCare app as you receive feedback from users, stakeholders, or testing phases.

2. Customer-Centric Approach

- Continuous Feedback: Agile emphasizes frequent collaboration with stakeholders (such as doctors, patients, or healthcare professionals), ensuring that the app meets their real-time needs and expectations. This is crucial for a project like MediCare, where user experience and accuracy are key.
- Faster Delivery of Features: With Agile, you can prioritize the most important features (such as doctor ratings, appointment scheduling) and deliver them incrementally, ensuring early value to users.

3. Improved Quality

- Testing and Feedback Loops: Agile encourages continuous testing throughout the development cycle, helping identify issues early on and maintain high quality in your project. For an app in the medical field, where reliability and accuracy are paramount, this is especially beneficial.
- Refinement of Features: Each iteration or sprint provides an opportunity to refine features, ensuring that they meet both user and technical requirements.

4. Enhanced Collaboration

- Cross-functional Teamwork: Agile promotes teamwork among developers, designers, product managers, and other stakeholders. This improves communication and ensures that all aspects of the MediCare project (technical, design, usability) are aligned.
- Transparency: Agile methodologies encourage regular meetings (like sprint reviews and stand-ups), making the development process transparent to all team members and stakeholders, which is vital in a collaborative project like MediCare.

5. Risk Management

- Early Identification of Risks: Agile's iterative approach helps to identify potential risks early in the project, allowing you to address them before they become significant issues.
- Frequent Releases: Each sprint produces a working version of the product, so you can catch potential problems early in the development cycle and make necessary adjustments.

6. Faster Time to Market

- Quick Iterations: Agile enables you to deliver a basic version of the MediCare app quickly (e.g., basic appointment scheduling or doctor information), allowing you to get valuable user feedback sooner and make improvements in the next cycle.
- Customer Value First: By breaking down the project into manageable pieces and releasing them incrementally, Agile ensures that users get the most critical features first.

7. Continuous Improvement

- Post-Release Enhancements: Once the basic functionalities of the app are delivered, Agile allows for continuous improvement through new sprints, ensuring that the MediCare app evolves based on user needs and market demands.

Conclusion:

For MediCare project, Agile helps ensure that the development process remains flexible, user-focused, and efficient. It allows for rapid adjustments, minimizes risks, and delivers valuable features early, which is especially beneficial in the fast-paced and constantly evolving healthcare domain.

➢ Stages of Agile

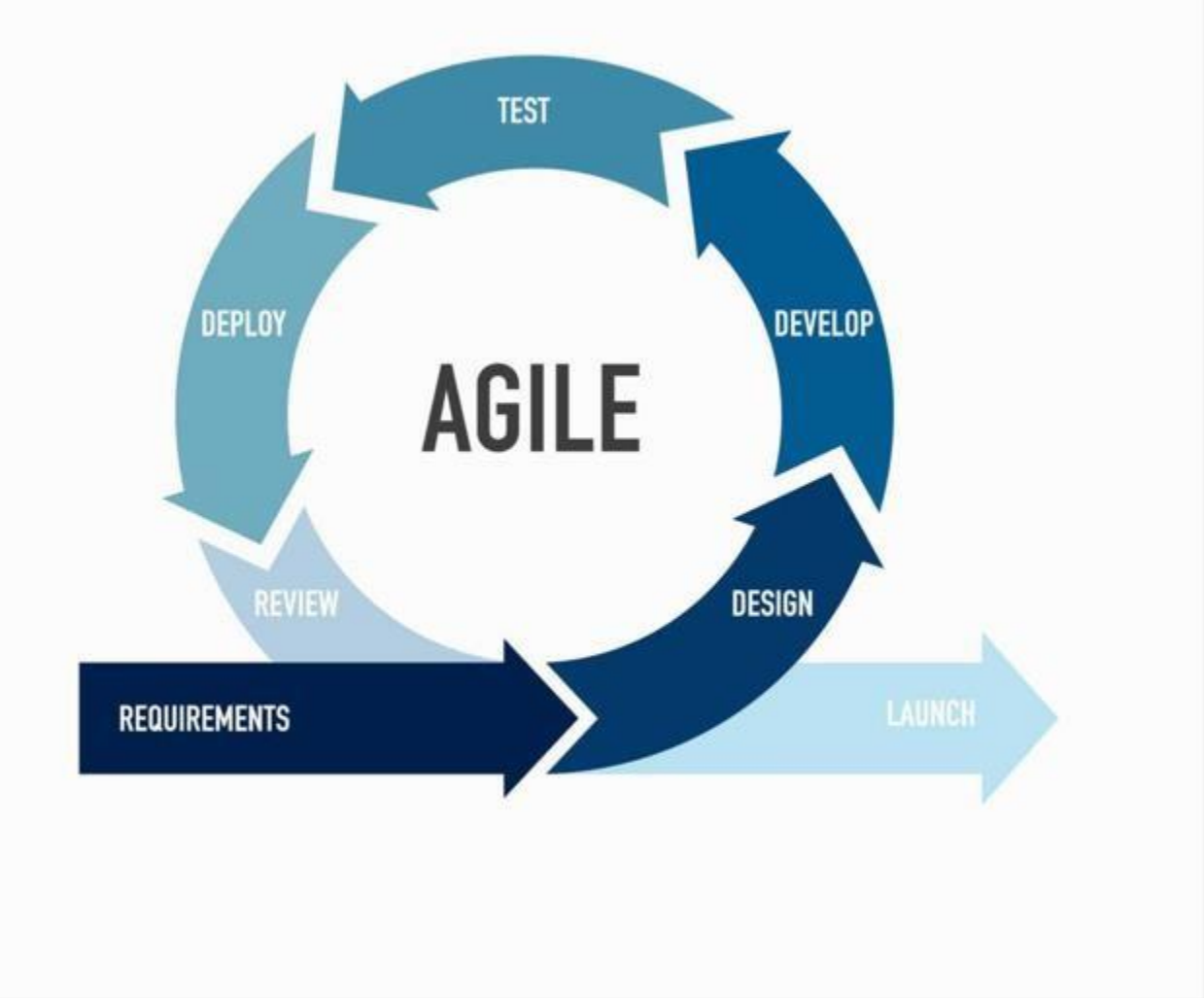


Figure 3.1 Agile Development

Requirement ⟶ In Agile, you begin by identifying the key features and functionalities, much like the requirement phase. This aligns with the Sprint Planning phase, where you plan what to deliver in each sprint, breaking down the tasks into user stories.

Design → The design phase can overlap with development in Agile, as design decisions are continuously made during the development process. During sprints, you focus on designing and iterating the user interface and architecture, and developers work on coding the features.

Development → The development phase is where the main work happens. In Agile, this is the Sprint Execution phase, where developers implement features and continuously test them.

Testing → In Agile, testing happens continuously throughout the sprints. You can think of your testing phase as part of Sprint Review and ongoing testing to ensure the application works as expected.

Deployment ⟶ Deployment aligns with the Release phase in Agile, where features and updates are deployed in smaller increments. After testing and approvals, the application is released or deployed to users.

Review ⟶ Finally, the Review phase is part of the Sprint Review and Retrospective. This is where you evaluate what went well, what can be improved, and plan the next steps for future sprints.

Scenario

➢ Use Case Scenario

Use case name: MediCare System.	Unique ID: UC-BD-01
Area: : MediCare System - End-to-End Functionality.	
Actor(s): Patients, Doctors, Admins.	
Stakeholders: Patients, Doctors, Admins.	
Description : MediCare system provides features for patients to search for doctors, hospitals, pharmacies, and labs; book appointments; and interact with the system via various functionalities. Doctors manage appointments, blogs, and offers, while admins oversee the entire system.	
Triggering Event: When a user logs into the system to perform actions like searching, booking, or managing resources.	
Trigger type: External , Temporal.	
Patient Use Cases	
Steps performed (Main Path)	**Information for Steps**
1. The patient registers on the system.	Fill out the registration form with name, contact, email, password, and more.
2. The patient logs into the system.	Enter email and password on the login page.
3. The patient searches for doctors, hospitals, pharmacies, or labs.	Uses filters (specialty, price, location) to refine results.
4. The patient views search results.	Details include names, locations, services, reviews, prices, and ratings.
5. The patient selects a doctor and books an appointment.	Select time slots and confirms the appointment. Payment is made online or at the clinic.
6. The patient views hospital or care center information.	Information includes services, departments, consultation fees, and operating hours.
7. The patient rates a pharmacy or lab after service.	Rates from 1 to 5 stars. The average rating is updated.
Doctor use cases	
1 . The doctor registers on the system.	Fill out the registration form with name, contact details, specialty, email, and password.

2. The doctor logs into the system.	Enter email and password on the login page.
3. The doctor views their dashboard.	Views upcoming appointments, patient details, and schedule summary.
4. The doctor manages appointments.	Updates appointment statuses (e.g., completed, rescheduled, or canceled).
5. The doctor creates blogs.	Adds title, content, and optional links, then publishes the blog for public viewing.
6. The doctor manages offers.	Creates, edits, or deletes promotional offers such as discounts or packages with start/end dates.
7. The doctor updates clinic information.	Edits or adds clinic details such as address, contact info, working hours, and consultation fees.
8. The doctor reviews patient feedback.	Views and responds to ratings or feedback to improve services.
Admin Use Cases Summary	
1. The admin logs into the system.	logs in with email and password to access the admin dashboard.
2. The admin manages users.	views, edits, or deletes user accounts (patients and doctors).
3. The admin manages doctors and clinics.	adds or updates doctor profiles and clinic information.
4. The admin manages hospitals, pharmacies, and labs.	adds or updates hospitals, pharmacies, and lab details.
5. The admin manages content.	approves, edits, or deletes blogs and promotional offers.
6. The admin manages offers and discounts.	oversees doctor-created offers, ensuring accuracy and validity.
7. The admin monitors appointments and bookings.	views and manages appointments, resolving conflicts or cancellations.
8. The admin manages feedback and ratings.	monitors and resolves issues with reviews and ratings.
9. The admin generates system reports.	generates reports on system usage, activities, and payments.
10. The admin manages media.	Admin uploads and organizes media content related to doctors, clinics, and offers.

Postconditions

- Patients successfully book appointments, search for doctors, and access detailed healthcare information.
- Doctors maintain accurate schedules and manage offers effectively.

• Admins ensure system reliability, secure transactions, and up-to-date records for smooth operation
Assumptions • All users have stable internet access for seamless interaction with the system. • The system incorporates a secure and efficient payment gateway. • Healthcare providers (doctors, hospitals, and labs) regularly update their schedules and availability
Questions 1. Should the system send automated appointment reminders via email and/or SMS for better user engagement? 2. Should patients have the ability to cancel appointments independently without requiring admin approval?

1.Use Case: Google Login

- **Name**: Log in with Google
- **Description**: A user logs into the system using their Google account, creating or accessing their account and earning daily login points.
- **Actor**: User
- **Preconditions**:
  - User has a valid Google account.
  - Google ID token is provided and verifiable.
- **Steps**:
    - User submits Google ID token after signing in via Google.
    - System validates token using Google Client API.
    - System finds or creates a user based on email, updating name, google_id, and avatar.
    - System awards 10 points for daily login if 24 hours have passed since last_visit.
    - System generates and returns a JWT access_token and user details.
- **Outcome**: User is authenticated; account created/updated; token issued.
- **Exceptions**: Invalid or unverifiable Google token returns error.

2. Use Case: User Registration

- **Name**: Register a New User
- **Description**: A user signs up by providing personal details and receives a verification email.
- **Actor**: User
- **Preconditions**:
  - User has a unique, valid email address.
- **Steps**:
    - Users submit name, email, phone, address, birth date, and password.
    - System validates input (e.g., unique email).
    - System creates account and sends verification email with token.
    - System returns success message.
- **Outcome**: Account created; verification email sent.
- **Exceptions**: Invalid input (e.g., duplicate email) returns error.

3. Use Case: User Login

- **Name**: Log in to the System
- **Description**: A user logs in with email and password to access the system.
- **Actor**: Registered User
- **Preconditions**:
  - User has a verified account (email_verified_at not null).
  - Correct email and password provided.
- **Steps**:
    - User enters email and password.
    - System validates credentials and email verification.
    - System issues JWT token and awards 10 points for daily login if applicable.
    - System returns token and user details.
- **Outcome**: User authenticated; token issued.
- **Exceptions**: Invalid credentials or unverified email return errors.

4. Use Case: Email Verification

- **Name**: Verify User Email
- **Description**: A user verifies their email using a link to activate their account.
- **Actor**: Registered User
- **Preconditions**:
  - User has a valid, non-expired verification_token.
- **Steps**:
    - User clicks verification link in email.
    - System validates token and marks email verified.
    - System redirects to login page.
- **Outcome**: Email verified; user can log in.
- **Exceptions**: Invalid/expired token returns error.

5. Use Case: Resend Verification Email

- **Name**: Resend Verification Email
- **Description**: A user requests a new verification email.
- **Actor**: Registered User
- **Preconditions**:
  - User has an unverified account (email_verified_at null).
  - Email exists in system.
- **Steps**:
    - User submits email for resend.
    - System checks unverified status and sends new email.
    - System returns success message.
- **Outcome**: New verification email sent.
- **Exceptions**: Invalid/already verified email returns error.

6. Use Case: Forgot Password

- **Name**: Request Password Reset
- **Description**: A user requests a password reset link.
- **Actor**: Registered User
- **Preconditions**:
  - User's email exists in system.
- **Steps**:
    - User submits email.
    - System validates email and sends reset link with token.
    - System returns success message.
- **Outcome**: Reset email sent.
- **Exceptions**: Invalid email or email failure returns error.

7. Use Case: Reset Password

- **Name**: Reset User Password
- **Description**: A user resets their password using a reset link.
- **Actor**: Registered User
- **Preconditions**:
  - User has a valid, non-expired reset_token.
- **Steps**:
    - User submits email, token, and new password.
    - System validates token and updates password.
    - System returns success message.
- **Outcome**: Password updated; user can log in.
- **Exceptions**: Invalid/expired token returns error.

8. Use Case: Get Account Details

- **Name**: Retrieve Account Information
- **Description**: A logged-in user views their account details.
- **Actor**: Authenticated User
- **Preconditions**:
  - User is authenticated with valid JWT token.
- **Steps**:
    - User requests account details.
    - System returns user data (e.g., name, email).
- **Outcome**: User receives account details.
- **Exceptions**: Invalid token returns error.

9. Use Case: Logout

- **Name**: Log Out
- **Description**: A user logs out, ending their session.
- **Actor**: Authenticated User
- **Preconditions**:
  - User is authenticated with valid JWT token.
- **Steps**:
    - User sends logout request.
    - System invalidates token and returns success message.
- **Outcome**: Session ended.
- **Exceptions**: Invalid token returns error.

10. Use Case: Delete Account

- **Name**: Delete User Account
- **Description**: A user deletes their account.
- **Actor**: Authenticated User
- **Preconditions**:
  - User is authenticated with valid JWT token.
  - Non-OAuth users must provide correct password.
- **Steps**:
    - User submits deletion request (with password if non-OAuth).
    - System verifies credentials, logs out, and deletes account.
    - System returns success message.
- **Outcome**: Account deleted.
- **Exceptions**: Invalid token/password returns error.

11. Use Case: View Available Appointments

- **Name**: Get Available Appointments
- **Description**: A user views available appointment slots for a doctor on a specific day.
- **Actor**: User
- **Preconditions**:
  - Doctor ID exists in the doctors table.
  - Specified day is valid.
- **Steps**:
    - User selects a doctor and day.
    - System retrieves unbooked appointments for the doctor and day.
    - System returns list of available appointments.
- **Outcome**: User receives available appointment slots.
- **Exceptions**: Invalid doctor ID returns empty or error response.

12. Use Case: Create Reservation

- **Name**: Book a Reservation
- **Description**: A user books an appointment with a doctor, applying a discount based on points.
- **Actor**: Authenticated User
- **Preconditions**:
  - User is authenticated with a valid JWT token.
  - Doctor, clinic, and appointment IDs exist; appointment is unbooked.
  - User has sufficient points for discount.
- **Steps**:
  - User submits doctor, clinic, appointment IDs, and status.
  - System validates input and checks appointment availability.
  - System calculates discount (points/10 * 5), deducts points, and creates reservation.
  - System marks appointment as booked and sends confirmation email.
  - System returns reservation details.
- **Outcome**: Reservation created; appointment booked; email sent.
- **Exceptions**: Invalid input, booked appointment, or insufficient points return errors..

13. Use Case: Confirm Reservation (User)

- **Name**: Confirm User Reservation
- **Description**: A user confirms their pending reservation.
- **Actor**: Authenticated User
- **Preconditions**:
  - User is authenticated and owns the reservation.
  - Reservation exists and is in pending status.
- **Steps**:
    - User requests to confirm reservation.
    - System verifies user ownership and pending status.
    - System updates reservation to confirmed and marks appointment as booked.
    - System returns confirmation message.

- **Outcome**: Reservation confirmed; appointment booked.
- **Exceptions**: Unauthorized user or non-pending reservation returns error

14. Use Case: Cancel Reservation (User)

- **Name**: Cancel User Reservation
- **Description**: A user cancels their reservation.
- **Actor**: Authenticated User
- **Preconditions**:
  - User is authenticated and owns the reservation.
  - Reservation exists.
- **Steps**:
    - User requests to cancel reservation.
    - System verifies user ownership.
    - System deletes related notifications, marks appointment as unbooked, and deletes reservation.
    - System returns cancellation message.
- **Outcome**: Reservation canceled; appointment freed.
- **Exceptions**: Unauthorized user or non-existent reservation returns error.

15. Use Case: View User Reservations

- **Name**: Get User Reservations
- **Description**: A user views their reservation history.
- **Actor**: Authenticated User
- **Preconditions**:
  - User is authenticated with a valid JWT token.
- **Steps**:
    - User requests their reservations.
    - System retrieves user's reservations with doctor, clinic, and appointment details.
    - System returns reservation list.
- **Outcome**: User receives reservation history.
- **Exceptions**: Unauthorized user returns error.

16. Use Case: View Doctor Reservations

- **Name**: Get Doctor Reservations
- **Description**: A doctor views all their reservations.
- **Actor**: Authenticated Doctor
- **Preconditions**:
  - Doctor is authenticated with a valid token and has doctor role.
- **Steps**:
    - Doctor requests their reservations.
    - System retrieves reservations linked to the doctor with user, clinic, and appointment details.
    - System returns reservation list.

- **Outcome**: Doctor receives reservation list.
- **Exceptions**: Unauthorized or non-doctor user returns error.

17. Use Case: View Specific Reservation (Doctor)

- **Name**: Get Reservation by ID
- **Description**: A doctor views details of a specific reservation.
- **Actor**: Authenticated Doctor
- **Preconditions**:
  - Doctor is authenticated with a valid token and has doctor role.
  - Reservation exists and belongs to the doctor.
- **Steps**:
    - Doctor requests reservation by ID.
    - System retrieves reservation with user, clinic, and appointment details.
    - System returns reservation data.
- **Outcome**: Doctor receives reservation details.
- **Exceptions**: Unauthorized, non-doctor, or non-existent reservation returns error.

18. Use Case: Mark Notification as Read

- **Name**: Mark Notification as Read
- **Description**: A user or doctor marks a reservation-related notification as read.
- **Actor**: Authenticated User or Doctor
- **Preconditions**:
  - User or doctor is authenticated.
  - Notification exists and belongs to the user/doctor.
- **Steps**:
    - User/doctor requests to mark notification as read.
    - System verifies notification ownership and marks it as read.
    - System returns success message.
- **Outcome**: Notification marked as read.
- **Exceptions**: Unauthorized or non-existent notification returns error.

19. Use Case: Confirm Reservation (Doctor)

- **Name**: Doctor Confirms Reservation
- **Description**: A doctor confirms a pending reservation.
- **Actor**: Authenticated Doctor
- **Preconditions**:
  - Doctor is authenticated with a valid token and has doctor role.
  - Reservation exists, belongs to the doctor, and is pending.
- **Steps**:
    - Doctor requests to confirm reservation.
    - System quas doctor ownership and pending status.
    - System updates reservation to confirmed and marks appointment as booked.
    - System returns confirmation message.

- **Outcome**: Reservation confirmed; appointment booked.
- **Exceptions**: Unauthorized, non-doctor, or non-pending reservation returns error.

20. Use Case: Cancel Reservation (Doctor)

- **Name**: Doctor Cancels Reservation
- **Description**: A doctor cancels a reservation.
- **Actor**: Authenticated Doctor
- **Preconditions**:
  - Doctor is authenticated with a valid token and has doctor role.
  - Reservation exists, belongs to the doctor, and is not already canceled.
- **Steps**:
    - Doctor requests to cancel reservation.
    - System verifies doctor ownership.
    - System deletes notifications, marks appointment as unbooked, and sets reservation to canceled.
    - System returns cancellation message.
- **Outcome**: Reservation canceled; appointment freed.
- **Exceptions**: Unauthorized, non-doctor, or already canceled reservation returns error

.

21. Use Case: Mark Reservation as Visited

- **Name**: Mark Reservation as Visited
- **Description**: A doctor marks a confirmed reservation as visited after the appointment.
- **Actor**: Authenticated Doctor
- **Preconditions**:
  - Doctor is authenticated with a valid token and has doctor role.
  - Reservation exists, belongs to the doctor, and is confirmed.
- **Steps**:
    - Doctor requests to mark reservation as visited.
    - System verifies doctor ownership and confirmed status.
    - System updates reservation to visited.
    - System returns success message.
- **Outcome**: Reservation marked as visited.
- **Exceptions**: Unauthorized, non-doctor, or non-confirmed reservation returns error

22. Use Case: Initiate Payment

- **Name**: Create Payment Intent
- **Description**: A user initiates a payment for a reservation using a card.
- **Actor**: Authenticated User
- **Preconditions**:
  - User is authenticated with a valid JWT token.
  - Reservation ID exists in the reservations table.
  - Stripe API is configured with a valid secret key.

- **Steps**:
    - User submits reservation ID and payment amount.
    - System validates input (amount > 0, valid reservation ID).
    - System creates a Stripe PaymentIntent with amount (in piastres) and currency (EGP).
    - System updates reservation with payment_intent_id, currency (EGP), and pending status.
    - System returns clientSecret for frontend card processing.
- **Outcome**: Payment intent created; user can complete payment on frontend.
- **Exceptions**: Invalid reservation, insufficient amount, or Stripe API errors return error.

23. Use Case: Update Payment Status

- **Name**: Confirm Payment Status
- **Description**: The system updates a reservation's payment status after Stripe confirms payment success.
- **Actor**: Authenticated User
- **Preconditions**:
  - User is authenticated with a valid JWT token.
  - Reservation exists with a valid payment_intent_id.
  - Stripe API is configured with a valid secret key.
- **Steps**:
    - User submits payment_intent_id after frontend payment attempt.
    - System retrieves PaymentIntent from Stripe to check status.
    - If status is succeeded, system updates reservation to succeeded and sends confirmation email.
    - System returns success message.
- **Outcome**: Payment status updated; user receives confirmation email.
- **Exceptions**: Non-existent reservation, non-succeeded payment, or Stripe API errors return error.

24. Use Case: Analyze Symptoms

- **Name**: Perform AI Health Analysis
- **Description**: A user submits symptoms (text) and/or a medical image for AI analysis to receive a probable diagnosis, recommended specialization, and advice.
- **Actor**: User
- **Preconditions**:
  - User provides either text describing symptoms or an image (or both).
  - Image, if provided, is a valid image file (e.g., JPEG, max 5MB).
- **Steps**:
    - User submits symptoms (text) and/or uploads an image via analysis form.
    - System validates input (text nullable, image optional, valid format).
    - System builds a prompt combining symptoms and/or image analysis instructions.
    - System calls Gemini API (gemini-2.0-flash) with prompt and image data (if provided).
    - System parses response into JSON format (diagnosis, specialization, advice, confidence, medications).
    - System returns analysis results with confidence message and medication warning.
- **Outcome**: User receives AI-generated health analysis with diagnosis, advice, and warnings.
- **Exceptions**: Missing input, invalid image, or Gemini API errors return error responses.

25. Use Case: Analyze Prescription Image

- **Name**: Analyze Prescription
- **Description**: A user uploads a prescription image for AI analysis to extract medication details, prescription summary, and advice.
- **Actor**: User
- **Preconditions**:
  - System is configured with a valid GEMINI_API_KEY.
  - User provides a valid image file (e.g., JPEG, max 5MB) of a prescription.
- **Steps**:
    - User uploads prescription image via analysis form.
    - System validates image (required, valid format).
    - System builds a prompt to extract medications, prescription details, and advice.
    - System calls Gemini API (gemini-2.0-flash) with prompt and image data.
    - System parses response into JSON (medications, prescription details, advice, confidence).
    - System returns analysis with confidence message and warning.
- **Outcome**: User receives prescription details, medications, and warnings.
- **Exceptions**: Invalid image, unclear prescription, or Gemini API errors return error.

26. Use Case: Get Medicine Details

- **Name**: Retrieve Medicine Information
- **Description**: A user requests detailed information about a specific medicine, including indications, dosage, side effects, and precautions.
- **Actor**: User
- **Preconditions**:
  - User provides a request with medicine name.
- **Steps**:
    - User submits a medicine name for analysis.
    - System builds a prompt for detailed medicine information in Arabic.
    - System calls Gemini API (gemini-2.0-flash) with the prompt.
    - System parses JSON response (indications, dosage, side effects, precautions, additional info).
    - System returns medicine details.
- **Outcome**: User receives detailed medicine information.
- **Exceptions**: Invalid medicine name or Gemini API errors return error.

27. Use Case: Analyze Lab Test Report

- **Name**: Analyze Lab Test
- **Description**: A user uploads a lab test report (image or PDF) for AI analysis to extract test results, interpretation, and recommendations.
- **Actor**: User
- **Preconditions**:
  - User provides a valid file (JPEG, PNG, or PDF, max 5MB) of a lab test report.
- **Steps**:
    - User uploads lab test report file via analysis form.
    - System validates file (required, valid format).
    - System builds a prompt to extract test results, interpretation, and recommendations.
    - System calls Gemini API (gemini-2.0-flash) with prompt and file data.
    - System parses response into JSON (test results, interpretation, recommendations).
    - System returns analysis with warning.
- **Outcome**: User receives lab test analysis with results, interpretation, and warnings.
- **Exceptions**: Invalid file, unclear report, or Gemini API errors return error.

Diagrams

Analysis Diagrams

- ✓ Use Case Diagram
- ✓ Context Diagram
- ✓ Data Flow Diagram
- ✓ Sequence Diagram
- ✓ Entity-Relationship Diagram
- ✓ Mapping

➢ Use Case Diagram

Figure 3.2Use Case Diagram

➢ Context Diagram

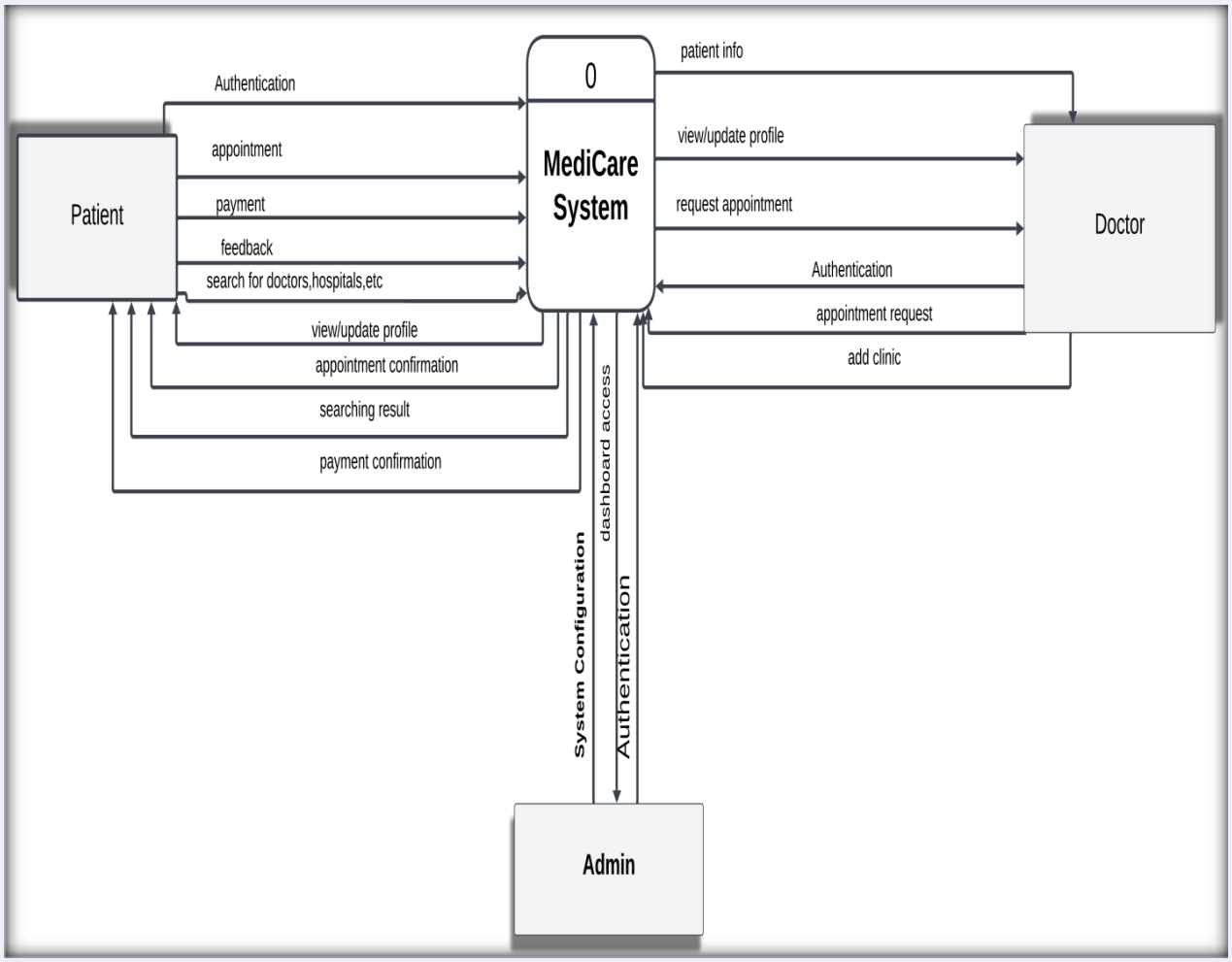


Figure 3.3 Context Diagram

➢ Data Flow Diagram

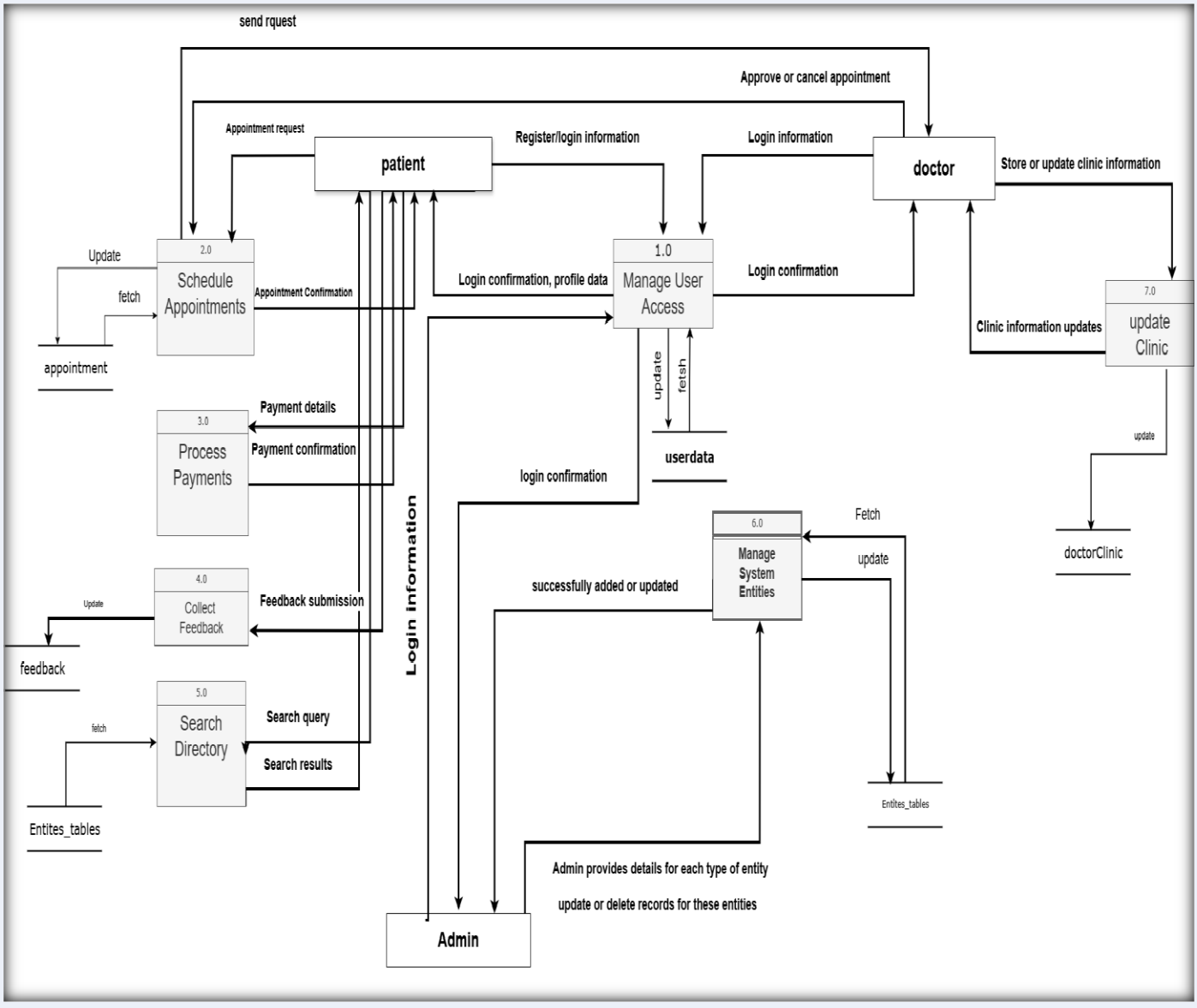


Figure 3.4 Data flow Diagram

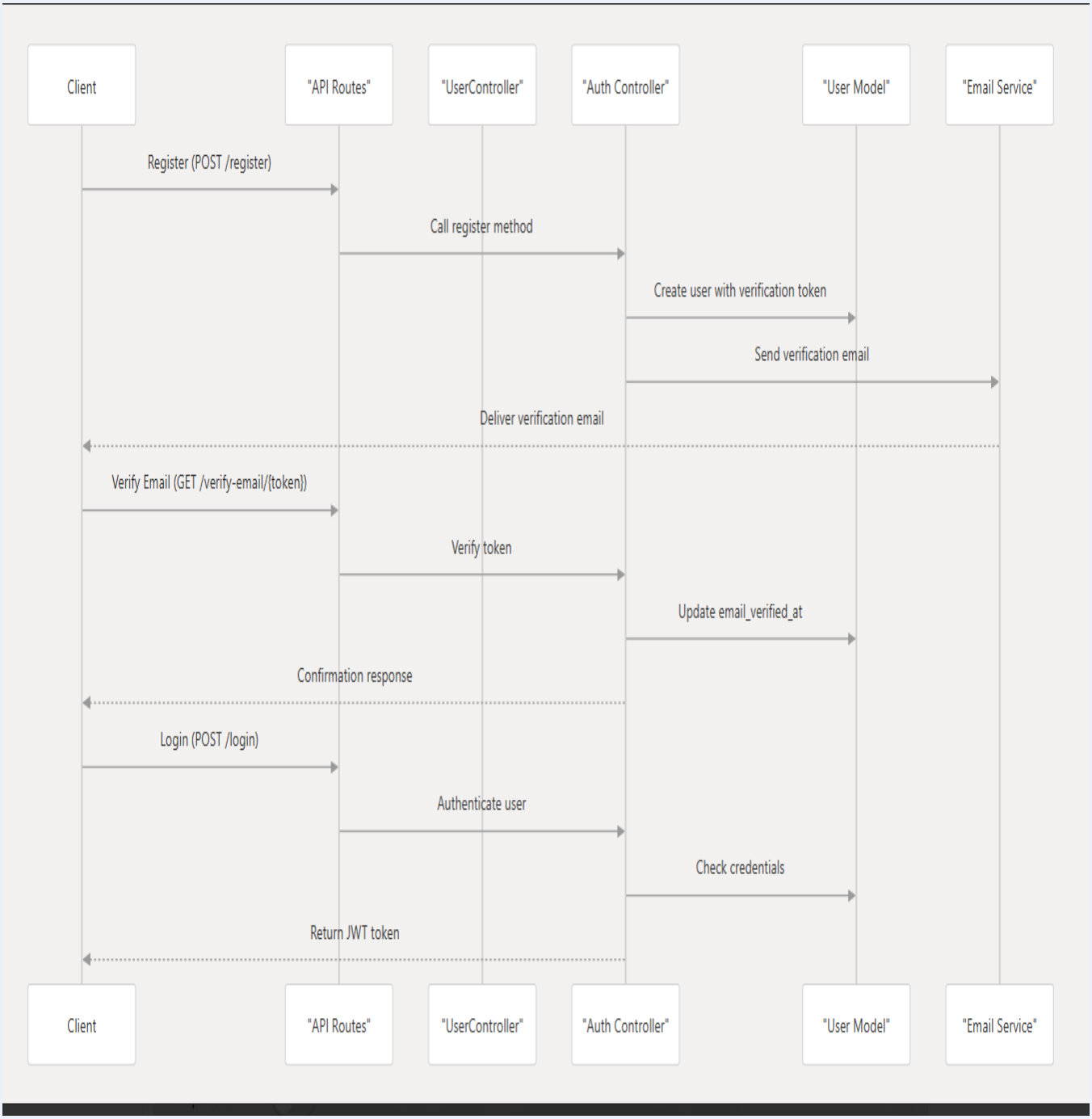


Figure 3.5 sequence diagram for Authentication System

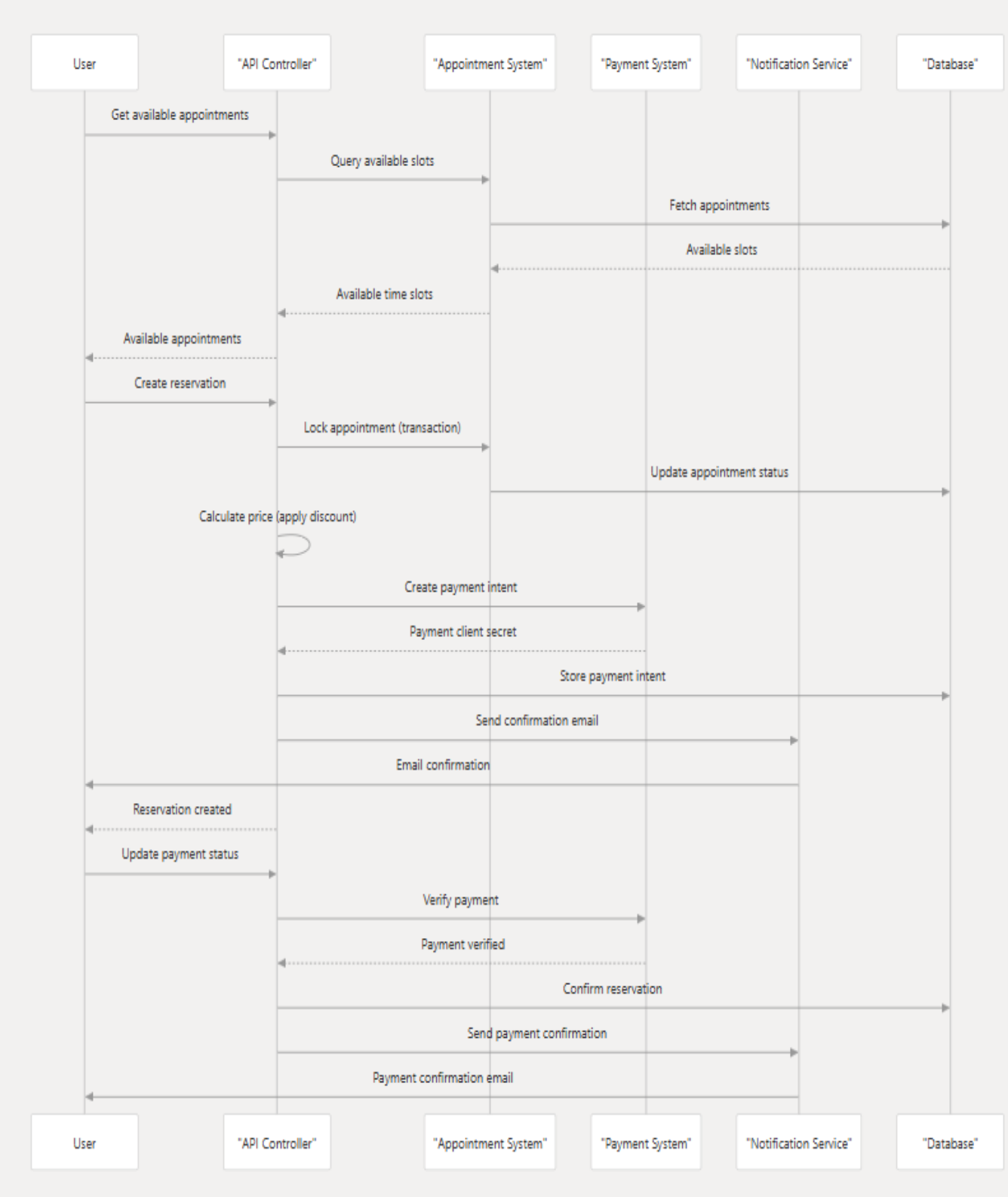


Figure 3.6 Sequence diagram for reservation system

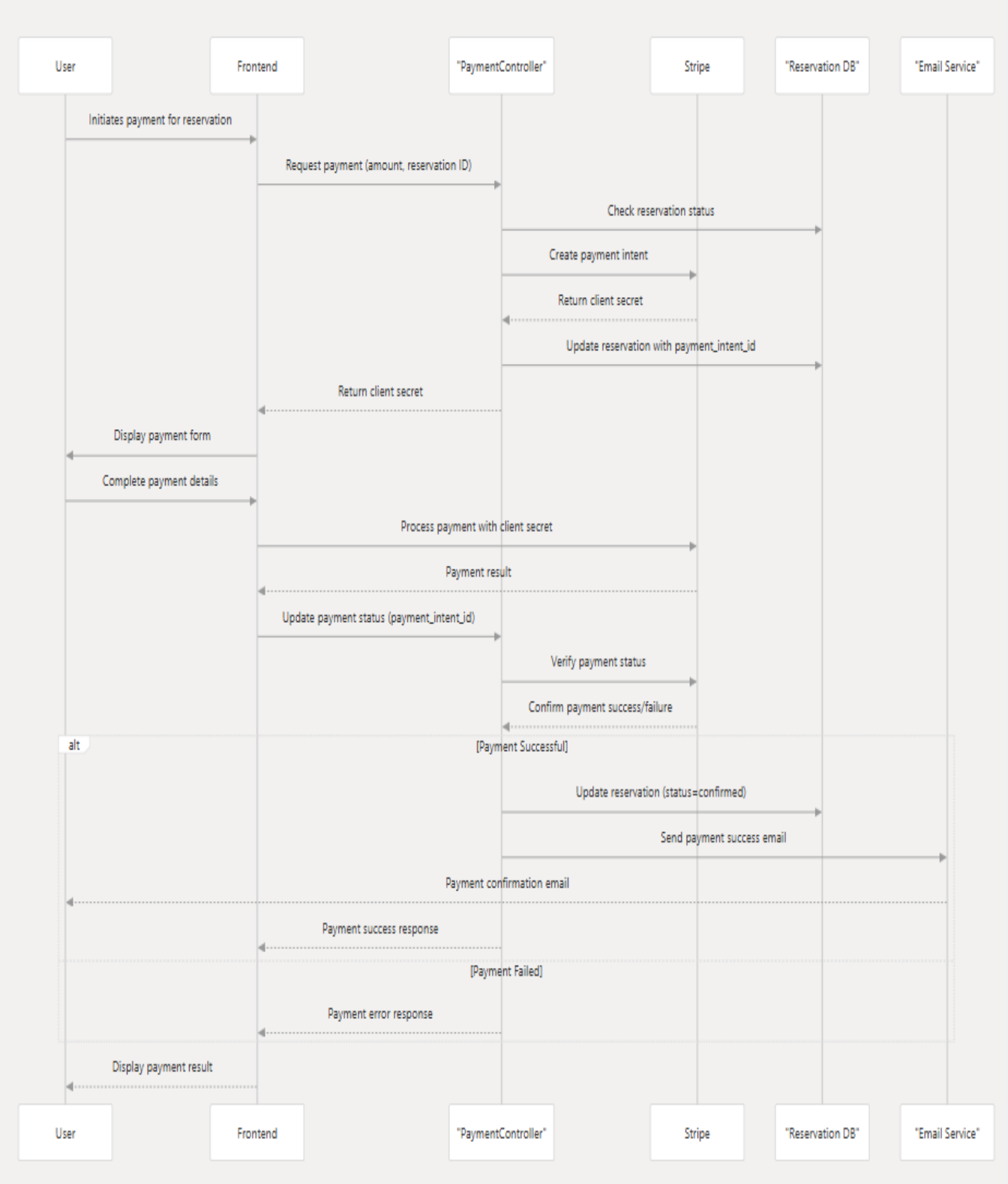


Figure 3.7 Sequence Diagram for payment Process

➢ Sequence Diagram

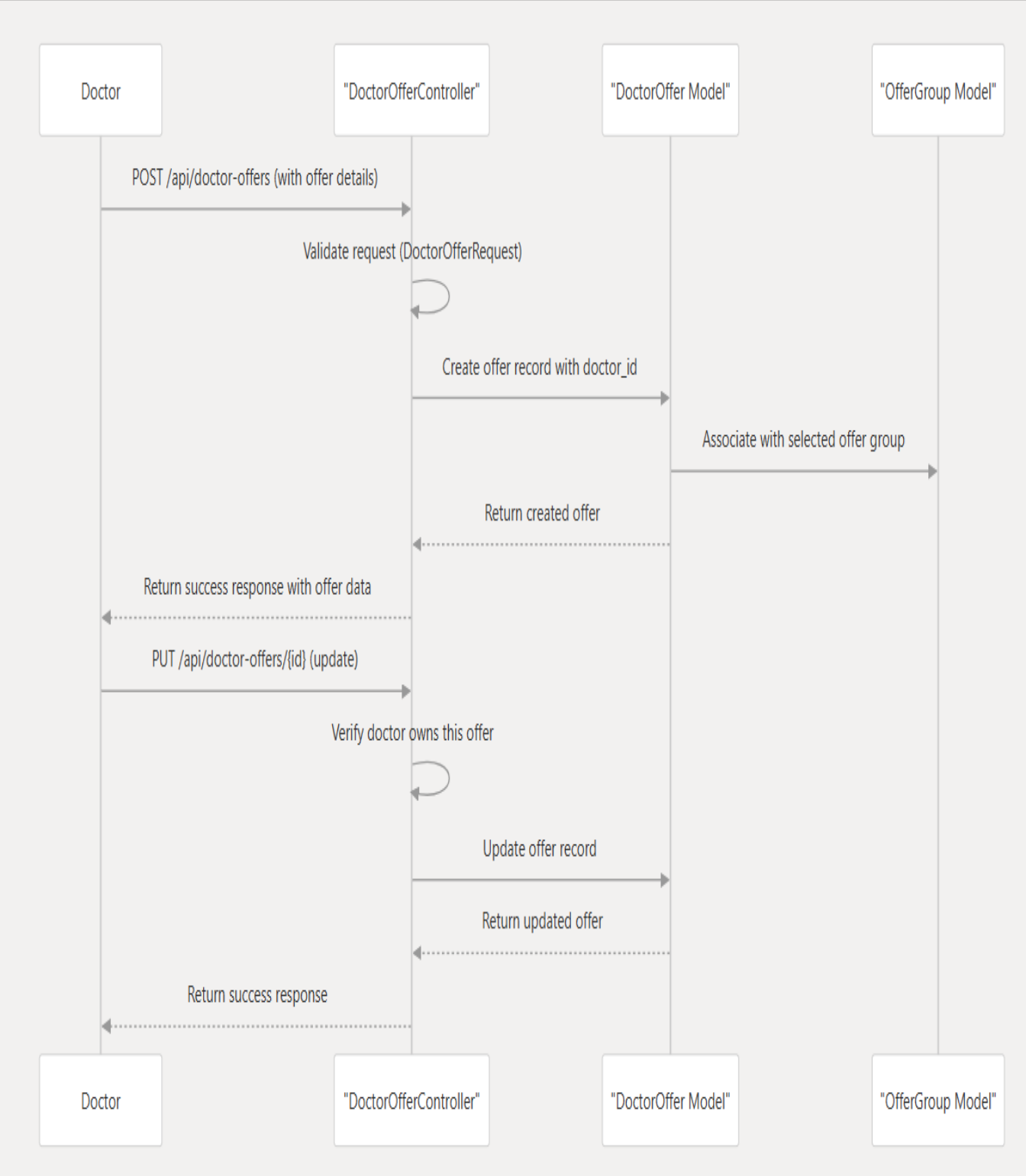


Figure 3.8 Sequence Diagram for doctor Offers

➢ Sequence Diagram

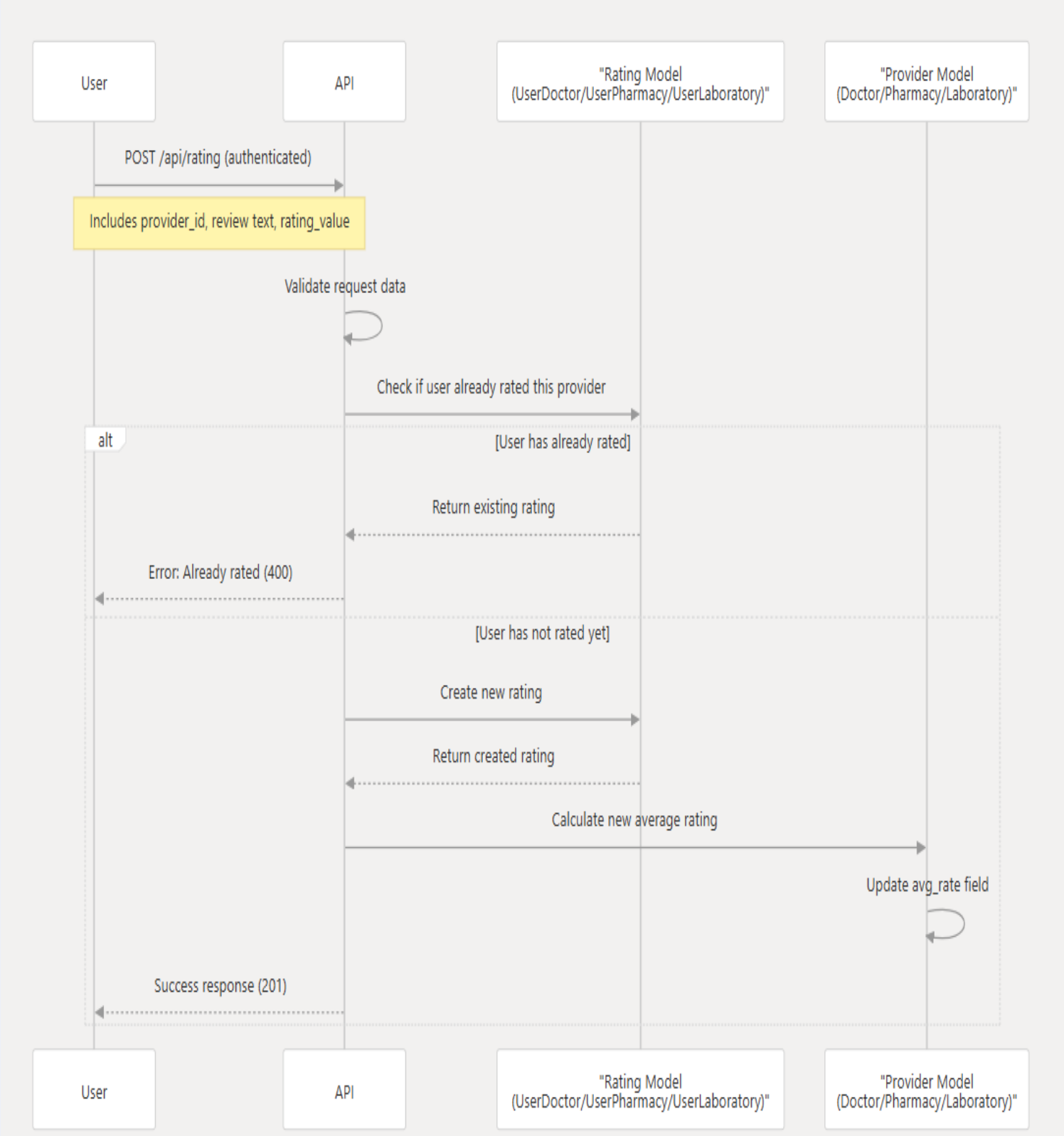


Figure 3.9 Sequence Diagram for feedback process

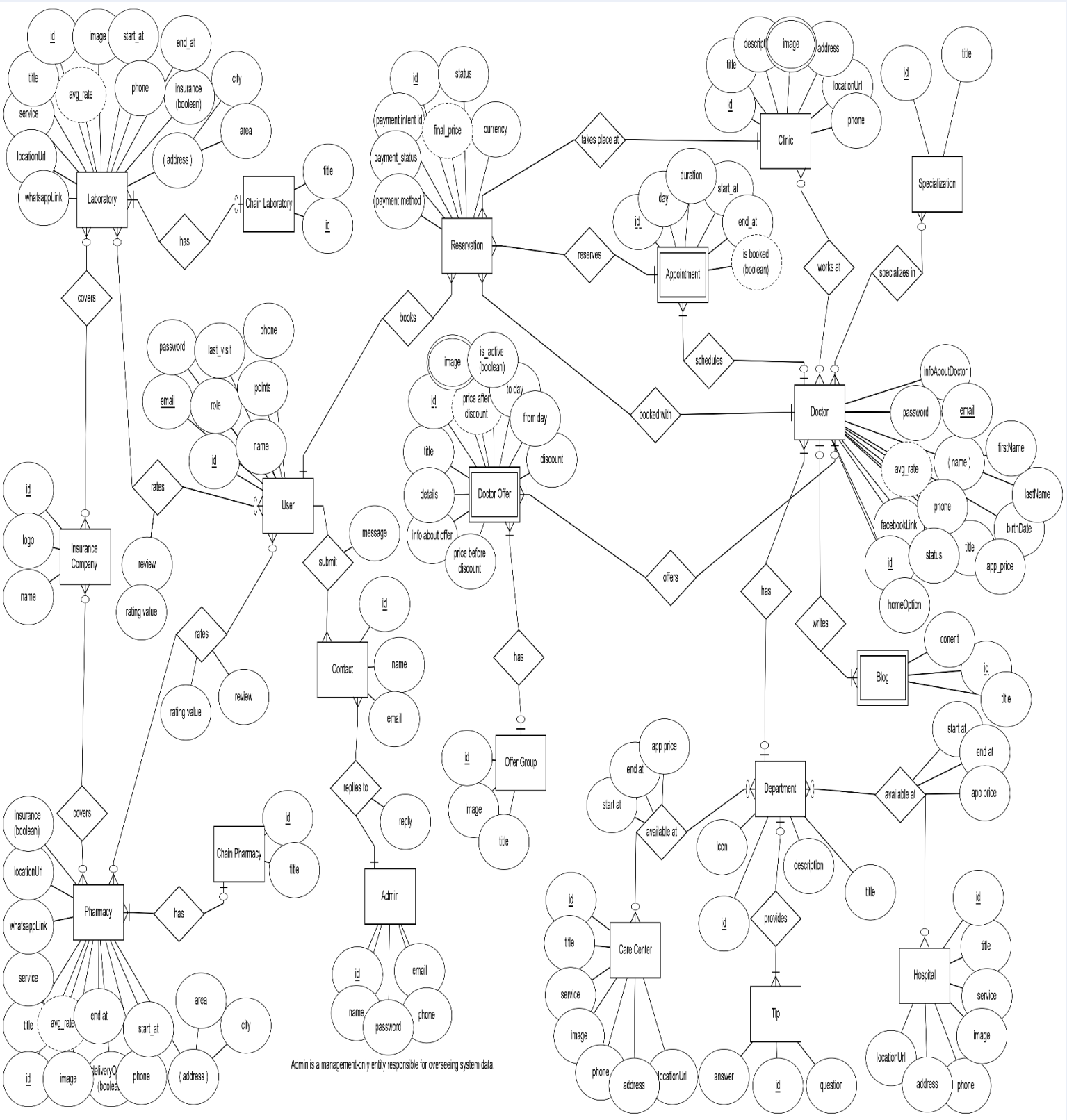


Figure 3.10 Entity Relations Diagram

➢ Mapping

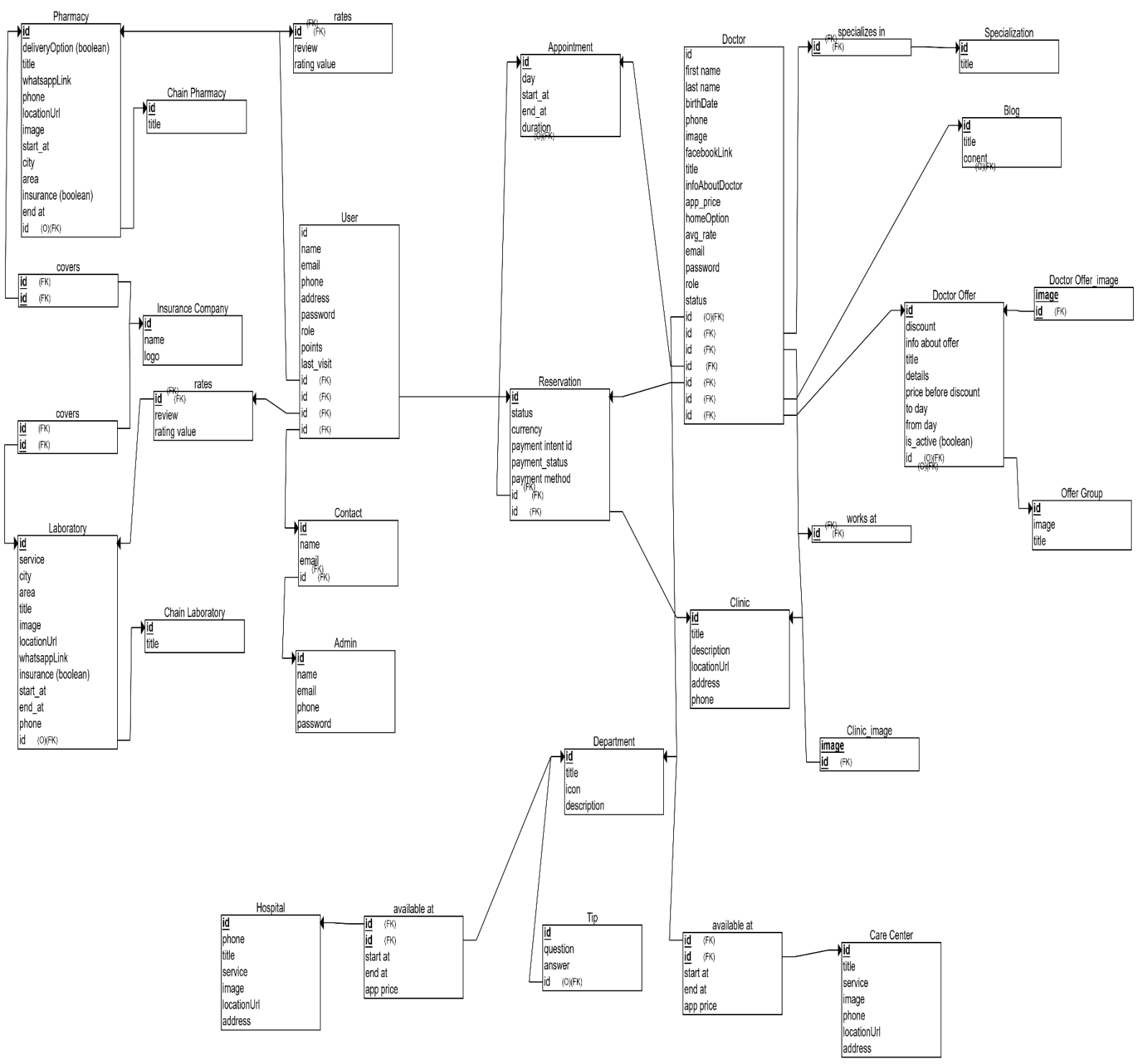


Figure 3.11 Mapping Schema

Chapter 4: Project Tools

In this section we will talk about the project tools.

Tools

➢ Tools which we used in Back-end:

Category	Technology
language	PHP
Framework	Laravel 10
Database	MySQL
Authentication	JWT(JSON Web Tokens)
Payment Processing	Stripe API
AI Analysis	Gemini API
OAuth Integration	Google API Client
Email service	Gmail App Password
Monitoring	Laravel Telescope
Testing	Postman

➢Tools which we used in Mobile Application:

Category	Technology
language	Dart
Framework	Flutter
State Management	Bloc / Cubit (Flutter Bloc)
Networking	Dio / http
Authentication	JWT Integration
UI Design	Material Design + Custom Widgets
Routing	GoRouter / Navigator 2.0
Local Storage	SharedPreferences
Image Caching	CachedNetworkImage
Animations	flutter_animate / Lottie
Maps Integration	Google Maps Flutter
Form Validation	Flutter Form + Validators
Responsive Design	flutter_screenutil / LayoutBuilder

➢Tools which we used in front-end:

Category	Technology
Core Technologies	HTML,CSS,Typescript
Framework	Angular 18 (standalone components)
UI Libraries	Bootstrap 5.3, PrimeNG, Font Awesome, Animate.css
State Management	RxJS Observables
Internationalization	ngx-translate
Authentication	JWT Authentication, Google Auth
Payment Processing	Stripe.js

➢Tools which we used in UI/UX:

- **Figma**

Chapter 5

System Design and Implementation (Design & Code)

In this section ,we will focus on detailing the design, and how the system is implemented.

This includes the design of the user interface (UI), and how the features of the system are coded.

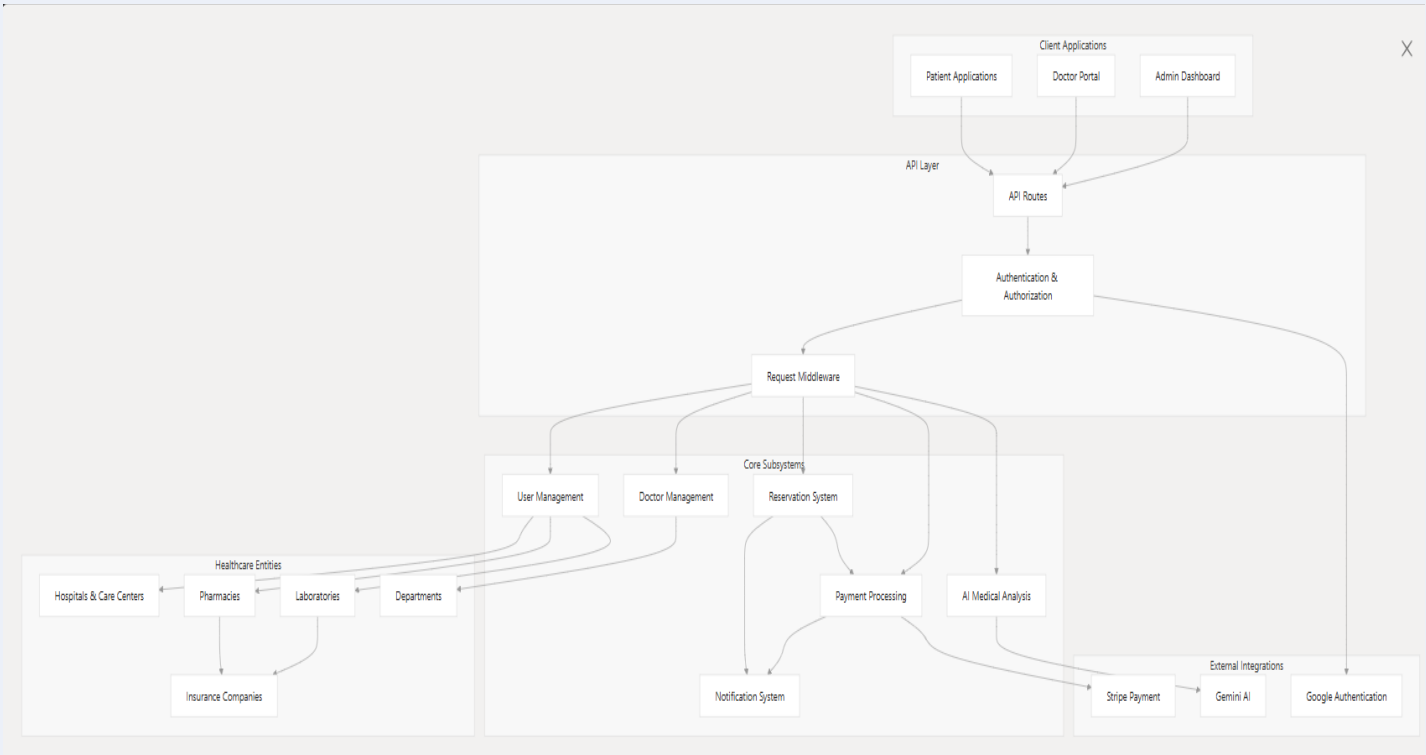
Backend system design and implementation

1. Overview

The MediCareAPI is a comprehensive healthcare platform developed using Laravel 10. It facilitates connections between patients and healthcare providers, offering services like appointment scheduling, pharmacy access, laboratory testing, and AI-powered medical analysis. The system employs a modular architecture with RESTful APIs and implements role-based access control for patients, doctors, and administrators.

2. High-Level Architecture

The MediCareAPI architecture follows a modular design with clearly separated concerns. The system is built on Laravel framework and uses a RESTful API approach for all client-server communications.



3. API Routes Structure

➢ The API routes in MediCareAPI are organized into five main categories, each contained in its own file:

- ✓ **Admin Routes** - Endpoints accessible to administrators
- ✓ **Doctor Routes** - Endpoints accessible to medical professionals
- ✓ **User Routes** - Endpoints accessible to regular users/patients
- ✓ **Auth Routes** - Authentication-related endpoints for all user types
- ✓ **Common** Routes - Shared endpoints accessible to multiple user roles

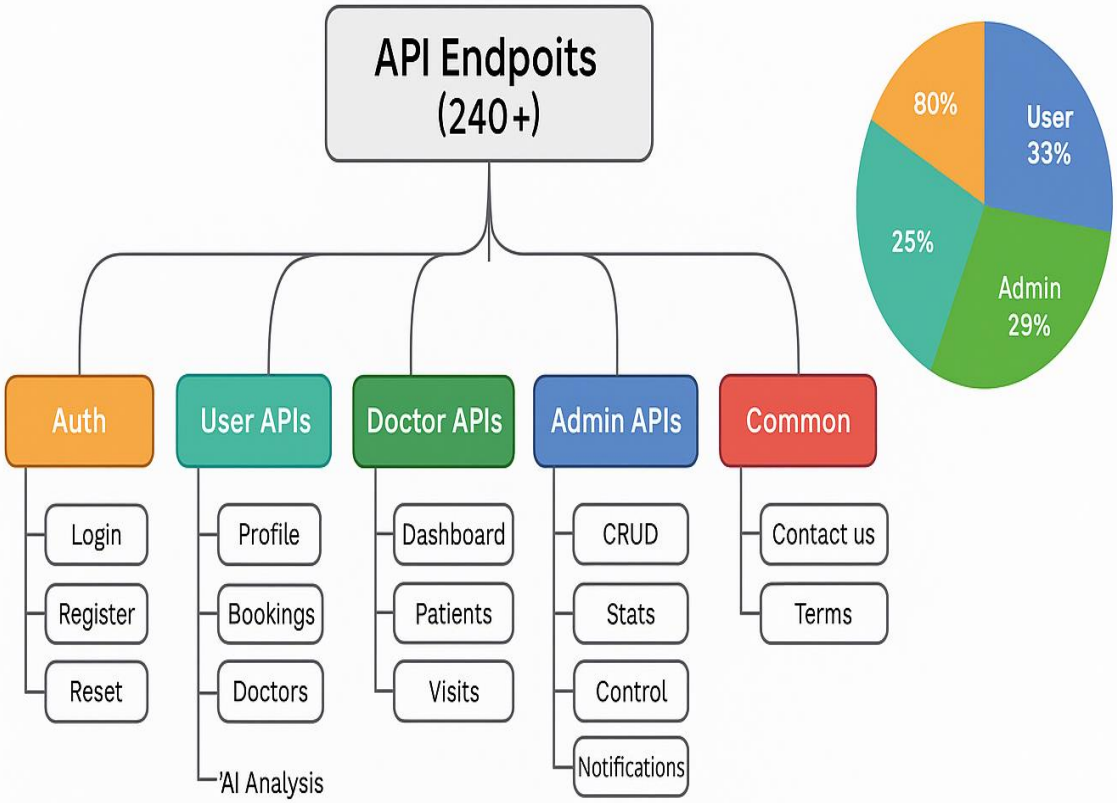
This organization provides clear separation of concerns and makes it easier to manage permissions and maintain the codebase.

➢ Route Categories and Authentication Requirements

This table summarizes the route categories, their authentication requirements, and primary purposes:

Route Category	Authentication Required	Primary Purpose
Admin Routes	Yes (Admin role)	System administration, entity management, statistics
Doctor Routes	Yes (Doctor role)	Appointment management, offer creation, profile management
User Routes	Yes (User role)	Reservations, ratings, profile management, points system
Auth Routes	Varies by endpoint	User registration, login, password reset, OAuth
Common Routes	Mostly no	Browsing departments, hospitals, pharmacies, laboratories

➢ Endpoints overview.

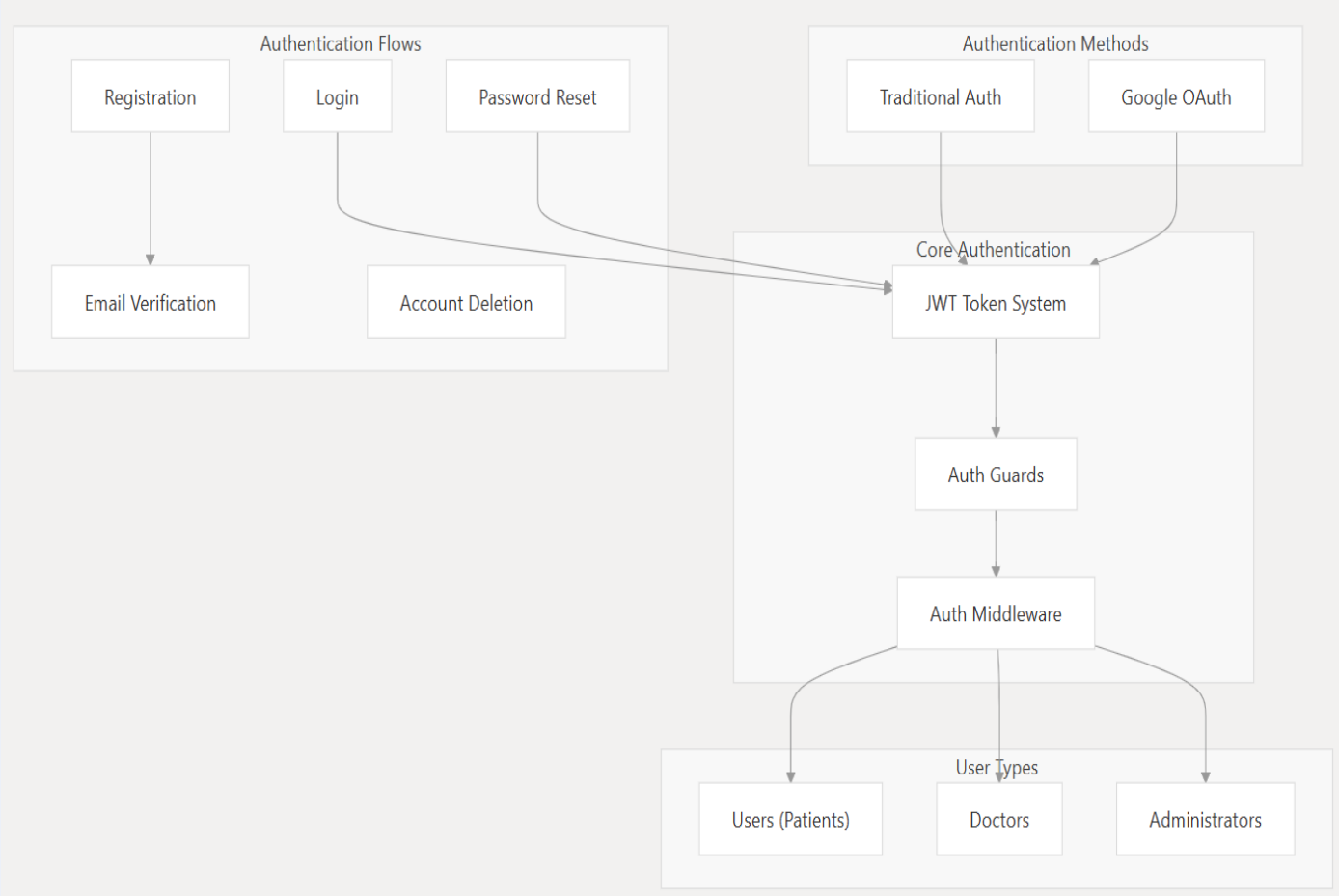


Category	No. of Erodponints	Main Features Covered
Auth	20+	Login, Register, Reset, Google Login
User	50+	Profile, Bookings, Doctors, AI Analysis
Doctor	60+	Dashboard, Patients, Visits, Payments
Admin	70+	Contact us, Terms

4.Authentication System

➢ System overview

The MediCareAPI authentication system implements a token-based authentication strategy using JSON Web Tokens (JWT) for all API requests. It provides multiple authentication methods including traditional email/password authentication and Google OAuth integration. The system also includes complete flows for user registration, email verification, password recovery, and account management.



➢ Authentication Methods

The system supports two primary authentication methods:

1. Traditional Email/Password Authentication

Users can register and authenticate using their email address and password. This method includes:

- Secure password hashing using Laravel's **Hash** façade.
- Email verification via one-time tokens.
- Password reset functionality.
- Account management (profile updates, deletion).

2. Google OAuth Authentication

Users can authenticate using their Google accounts through OAuth 2.0. This provides:

- Simplified one-click authentication.
- Profile information syncing from Google.
- No password management required for users.
- The flow of this process is :
  1. User initiates Google login from the client application
  2. Client obtains a Google ID token.
  3. Client sends the token to the API.
  4. API verifies the token with Google.
  5. If valid, API creates or retrieves the user account.
  6. API issues a JWT token for the authenticated user

➢ Authentication API Endpoints

User Authentication Endpoints

Endpoint	Method	Description	Authentication
`/api/user/register`	POST	Register a new user	None
`/api/user/login`	POST	Authenticate user and get token	None
`/api/user/auth/google`	POST	Login with Google	None
`/api/user/verify-email/{token}`	GET	Verify user's email	None
`/api/user/resend-email`	POST	Resend verification email	None
`/api/user/password/forgot`	POST	Request password reset	None
`/api/user/password/reset`	POST	Reset password	None
`/api/user/getaccount`	GET	Get authenticated user details	JWT
`/api/user/logout`	POST	Invalidate current token	JWT
`/api/user/account`	DELETE	Delete user account	JWT
`/api/user/profile`	POST	Update user profile	JWT

Doctor Authentication Endpoints

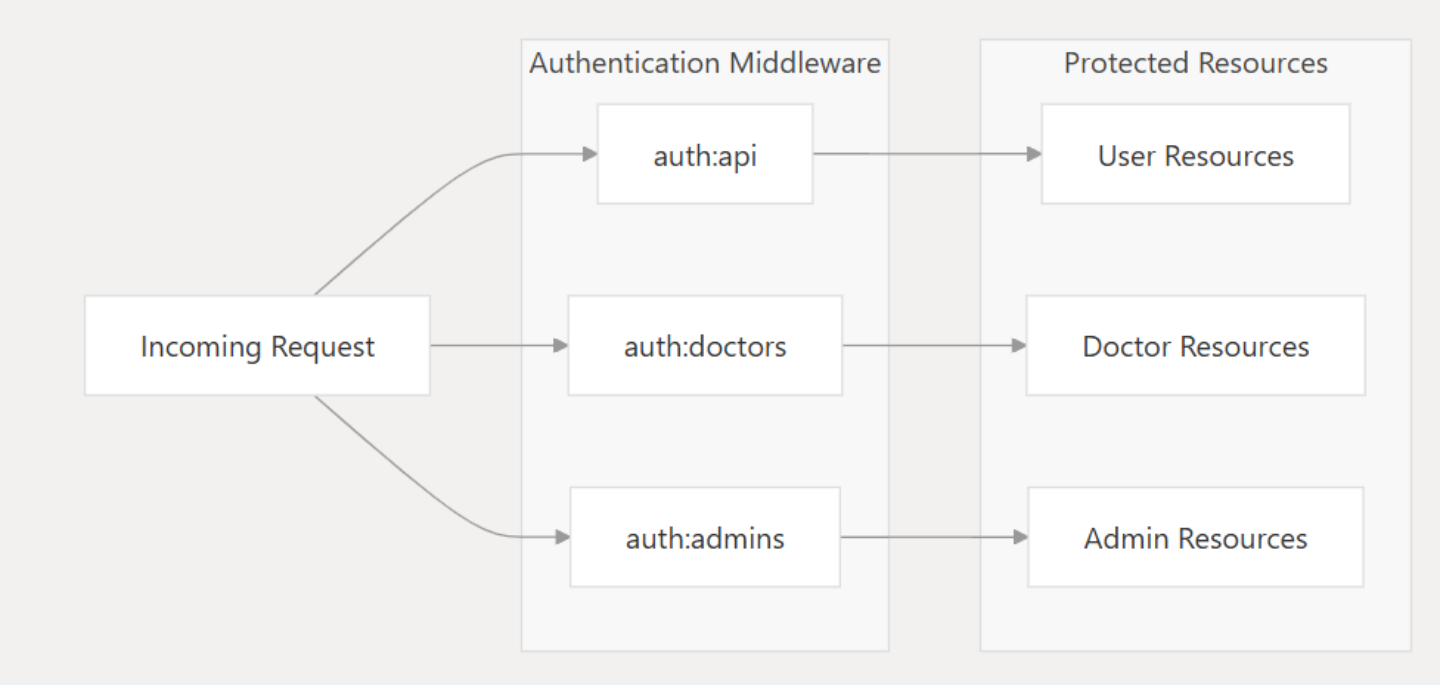
Endpoint	Method	Description	Authentication
`/api/doctor/login`	POST	Authenticate doctor and get token	None
`/api/doctor/logout`	POST	Invalidate current token	JWT
`/api/doctor/getaccount`	GET	Get doctor details	JWT

Admin Authentication Endpoints

Endpoint	Method	Description	Authentication
`/api/admin/login`	POST	Authenticate admin and get token	None
`/api/admin/register`	POST	Register a new admin	None
`/api/admin/logout`	POST	Invalidate current token	JWT
`/api/admin/getaccount`	GET	Get admin details	JWT

➢ Authentication Middleware

The system uses middleware to protect routes that require authentication. Different middleware is used for each user type:



➢ Security Features

The authentication system implements several security best practices:

1. **Password Hashing**: All passwords are hashed using Laravel's Hash facade (bcrypt)
2. **Timebound Tokens**: Email verification and password reset tokens expire after a set period
3. **Account Protection**: Password verification required for sensitive operations like account deletion
4. **Token Expiration**: JWT tokens have configurable expiration times
5. **CORS Protection**: API routes are protected against cross-origin request forgery

5.user management

➢ Profile Management

Users can view and update their profile information. The system keeps track of essential user information required for healthcare services.

➢ User Controller Operations

The UserController provides the following functionality:

1. Admin-only operations:
   - List all users
   - View specific user details
   - Delete users
2. User-specific operations:
   - Get user progress (points and last visit)
   - Redeem points for benefits

➢ Points Reward System

Points are stored in the User model and can be:

- Earned through system interactions like logging.
- Redeemed for benefits within the platform.

Points Redemption

Users can redeem their points when they accumulate at least 200 points:

- The system checks if the user has sufficient points.
- If successful, points are deducted from the user's balance.
- A success message is returned.
- If insufficient points, an appropriate error message is returned.

Email Notifications

The User Management system sends various email notifications including:

- Email verification upon registration.
- Password reset emails.

Email Verification

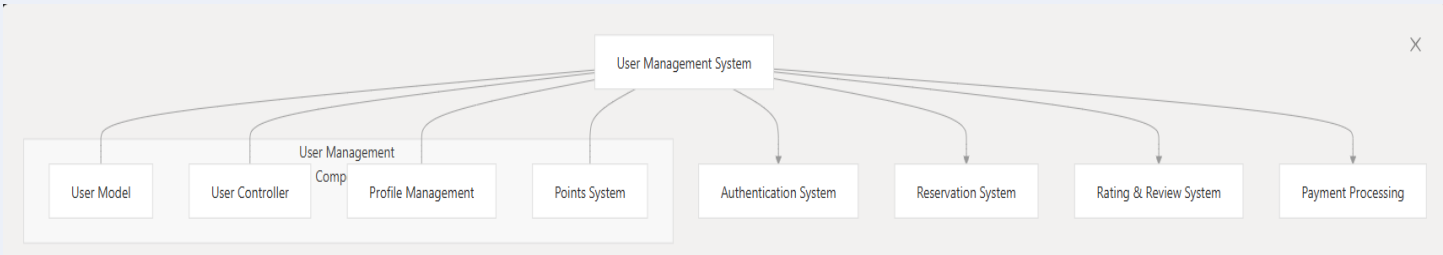
When a user registers, a verification email is sent with a token that expires after a set period. This ensures the email address belongs to the registering user.

Password Reset

Users can request a password reset, which generates a token and sends an email with a link to reset their password. The token expires after one hour for security.

Integration with Other Systems

The User Management system serves as a foundation for many other components of the MediCareAPI:



5. Doctor Appointments

➢ Appointment Overview

Doctor appointments in MediCareAPI represent time slots when doctors are available for consultations at specific clinics. These slots become available for patients to book through the reservation system. The appointment system allows doctors to define their availability by creating multiple time slots of fixed durations.

➢ Validation Rules

Appointments are subject to the following validation rules:

- The day must be a valid date
- Start and end times must be in HH:MM format
- End time must be after start time
- The doctor must be associated with the clinic where the appointment is being created
- Appointments cannot overlap with the doctor's existing appointments at the same clinic on the same day

➢ Endpoints

Method	Endpoint	Description
GET	`/appointments`	List all appointments for the authenticated doctor
POST	`/appointments`	Create new appointment slots
GET	`/appointments/{id}`	Get details of a specific appointment
PUT	`/appointments/{id}`	Update an appointment
DELETE	`/appointments/{id}`	Delete an appointment

5. Doctor Offers

Doctor Offer Endpoints

Endpoint	Method	Authentication	Description
`/api/doctor-offers`	GET	Doctor	Lists offers belonging to the authenticated doctor
`/api/doctor-offers`	POST	Doctor	Creates a new doctor offer
`/api/doctor-offers/{id}`	GET	Public	Views details of a specific offer with related data
`/api/doctor-offers/{id}`	PUT	Doctor	Updates an existing offer (doctor can only modify their own offers)
`/api/doctor-offers/{id}`	DELETE	Doctor	Deletes an offer (doctor can only delete their own offers)

Offer Group Endpoints

Endpoint	Method	Authentication	Description
`/api/offer-groups`	GET	Public	Lists all offer groups
`/api/offer-groups`	POST	Admin	Creates a new offer group
`/api/offer-groups/{id}`	GET	Public	Views details of a specific offer group with associated offers
`/api/offer-groups/{id}`	PUT	Admin	Updates an existing offer group
`/api/offer-groups/{id}`	DELETE	Admin	Deletes an offer group

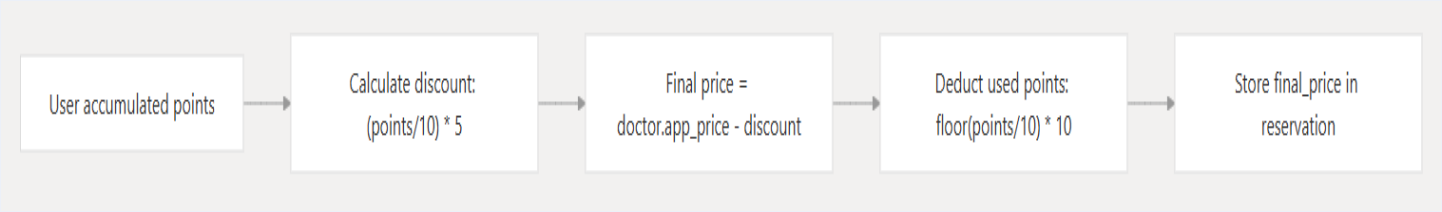
6- Reservation System

➢ Endpoints

Endpoint	Method	Description	Authentication
`getAvailableAppointments/{doctorId}/{day}`	GET	Retrieves available appointments for a doctor on a specific day	None
`store`	POST	Creates a new reservation	User
`confirmReservation/{reservationId}`	PUT	Confirms a reservation	User
`cancelReservation/{reservationId}`	DELETE	Cancels a reservation	User
`getUserReservations`	GET	Retrieves all reservations for authenticated user	User
`getDoctorReservations`	GET	Retrieves all reservations for authenticated doctor	Doctor
`getReservationById/{id}`	GET	Retrieves a specific reservation	Doctor
`markNotificationAsRead/{notificationId}`	PUT	Marks a notification as read	User/Doctor
`doctorConfirmReservation/{reservationId}`	PUT	Doctor confirms a reservation	Doctor
`doctorCancelReservation/{reservationId}`	PUT	Doctor cancels a reservation	Doctor
`markAsVisited/{reservationId}`	PUT	Doctor marks a reservation as visited	Doctor

➢ Points-Based Discount System

The Reservation System incorporates a points-based discount system that rewards user loyalty:



The discount calculation logic:

- For every 10 points, the user gets a 5 unit discount
- Points are deducted in multiples of 10
- The final price is stored with the reservation

➢ Race Condition Prevention

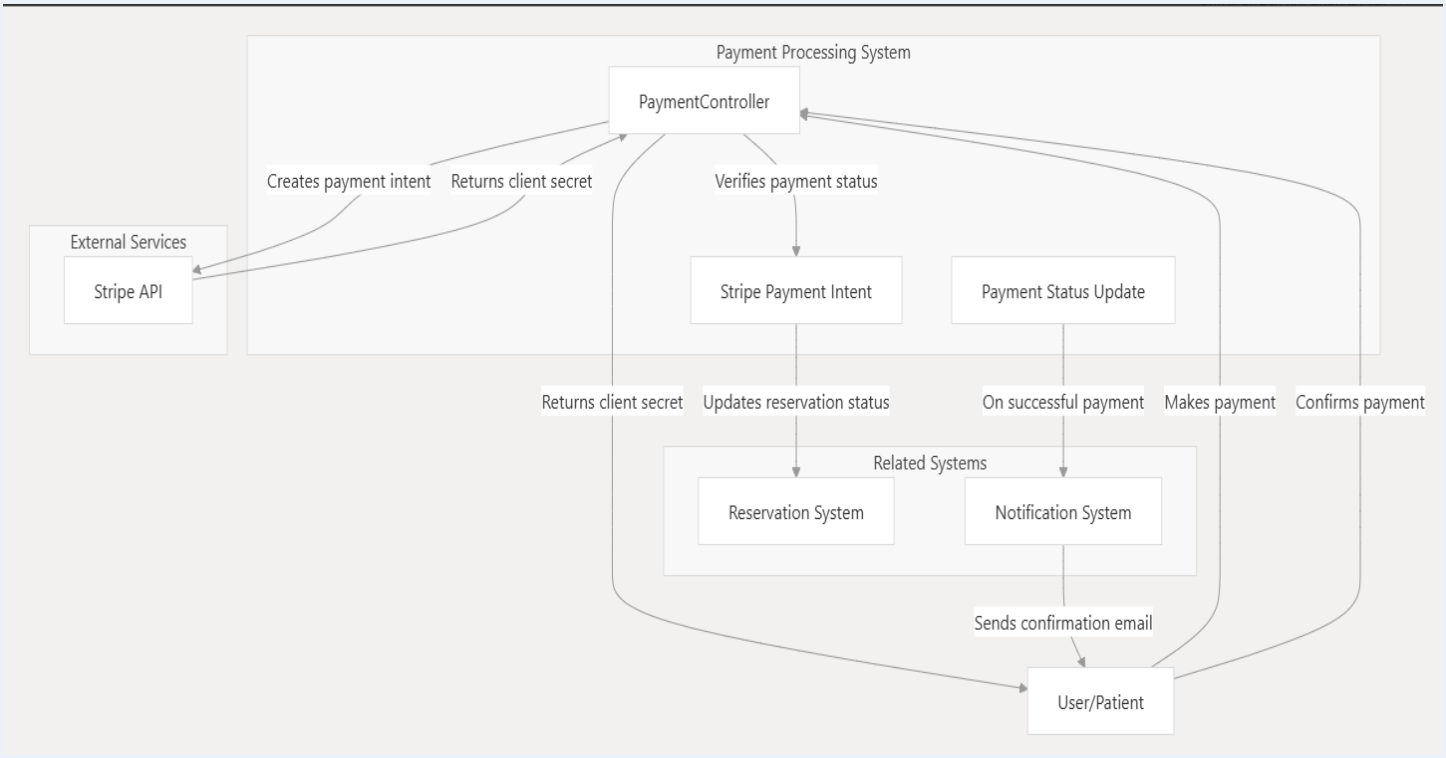
The system uses database transactions and row locking to prevent double-booking of appointments:

1. Begins a database transaction
2. Locks the appointment row for update
3. Verifies appointment availability within the locked context
4. Creates reservation and marks appointment as booked
5. Commits the transaction

This ensures that even with concurrent reservation attempts, only one will succeed.

7- Payment System

MediCareAPI implements a secure payment processing system using Stripe as the payment service provider. The system allows users to make payments for their medical reservations using credit/debit cards, with the payment flow integrated directly into the reservation process.



➢ Endpoints

Method	Endpoint	Description	Authentication
`POST`	`/payment`	Creates a new Stripe payment intent for a reservation or updates an existing one	Yes
`POST`	`/payment/update`	Verifies and updates the payment status after processing	Yes

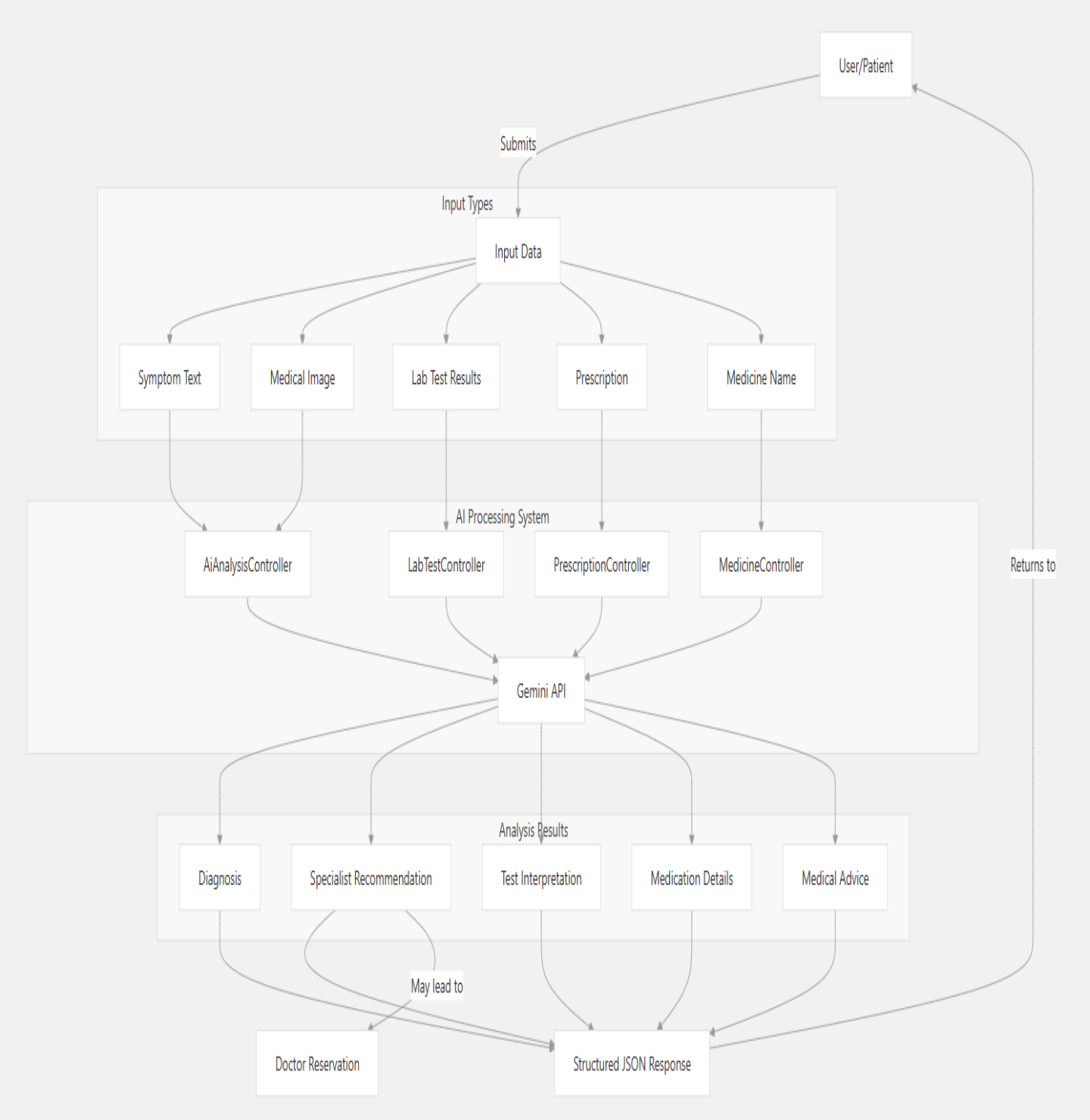
8- AI Medical Analysis

The AI Medical Analysis system serves as an intelligent assistant that can analyze health-related data and provide preliminary medical insights. It is not designed to replace professional medical advice but to provide users with initial guidance and better understanding of their medical information.

Endpoint	Method	Controller	Purpose
`/lab-tests/analyze`	POST	LabTestController	Analyze lab test image
`/prescriptions/analyze`	POST	PrescriptionController	Analyze prescription image
`/prescriptions/medicine/{name}`	GET	PrescriptionController	Get medicine details
`/medicines/{name}/details`	GET	MedicineController	Get detailed medicine information

➢ System Core Features

- Symptom analysis from text descriptions.
- Medical image analysis.
- Laboratory test result interpretation.
- Prescription analysis and medication information.
- Recommendations for medical specialists may be from the database.
- Confidence scores for analysis results.


User/Patient
Submits
Input Types
Input Data
Symptom Text
Medical Image
Lab Test Results
Prescription
Medicine Name
AI Processing System
AiAnalysisController
LabTestController
PrescriptionController
MedicineController
Returns to
Gemini API
Analysis Results
Diagnosis
Specialist Recommendation
Test Interpretation
Medication Details
Medical Advice
May lead to
Doctor Reservation
Structured JSON Response

9- Healthcare Providers

Pharmacy Endpoints

Method	Endpoint	Description	Authentication
GET	/api/pharmacies	List all pharmacies with optional filtering and pagination (6 items per page)	None
GET	/api/pharmacies/{id}	Get details of a specific pharmacy with its user ratings and insurance companies	None
POST	/api/pharmacies	Create a new pharmacy	Admin only
PUT	/api/pharmacies/{id}	Update an existing pharmacy	Admin only
DELETE	/api/pharmacies/{id}	Delete a pharmacy	Admin only

Laboratory Endpoints

Method	Endpoint	Description	Authentication
GET	/api/laboratories	List all laboratories with optional filtering and pagination (6 items per page)	None
GET	/api/laboratories/{id}	Get details of a specific laboratory with its user ratings and insurance companies	None
POST	/api/laboratories	Create a new laboratory	Admin only
PUT	/api/laboratories/{id}	Update an existing laboratory	Admin only
DELETE	/api/laboratories/{id}	Delete a laboratory	Admin only

Available Filter Parameters

Parameter	Description	Example
search	General search term for title, service, city, area, phone	?search=pediatrics
deliveryOption	Filter by delivery option (pharmacies only)	?deliveryOption=1
insurence	Filter by insurance acceptance	?insurence=1
min_rate	Filter by minimum average rating	?min_rate=4
chain_pharmacy_id / chain_laboratory_id	Filter by chain	?chain_pharmacy_id=123
city	Filter by city	?city=Cairo
area	Filter by area	?area=Maadi

➢ Integration with Other Systems

- *Rating System Integration*

Pharmacies and laboratories are integrated with the user rating system through many-to-many relationships. This allows:

1. Users to rate and review pharmacies and laboratories.
2. Calculation of average ratings displayed on listing pages.
3. Filtering entities by minimum rating score.

- *Insurance Company Integration*

Both pharmacies and laboratories can be associated with multiple insurance companies through pivot tables. This integration:

1. Indicates which insurance providers are accepted.
2. Allows filtering pharmacies and laboratories by insurance acceptance.
3. Provides details about insurance coverage when viewing a specific entity.

- *Chain Management Integration*

Pharmacies and laboratories can optionally belong to a chain (e.g., a branded network of pharmacies or laboratories). This relationship:

1. Allows grouping related entities together.
2. Enables filtering by chain affiliation.
3. Provides consistent branding and service information.

➢ Chain Pharmacies Endpoints

Method	Endpoint	Description	Access
GET	`/api/chain-pharmacies`	Retrieve all pharmacy chains	Public
POST	`/api/chain-pharmacies`	Create a new pharmacy chain	Admin Only
GET	`/api/chain-pharmacies/{id}`	Retrieve a specific pharmacy chain with its pharmacies	Public
PUT/PATCH	`/api/chain-pharmacies/{id}`	Update a pharmacy chain	Admin Only
DELETE	`/api/chain-pharmacies/{id}`	Delete a pharmacy chain	Admin Only

➢ Chain Laboratories Endpoints

Method	Endpoint	Description	Access
GET	`/api/chain-laboratories`	Retrieve all laboratory chains	Public
POST	`/api/chain-laboratories`	Create a new laboratory chain	Admin Only
GET	`/api/chain-laboratories/{id}`	Retrieve a specific laboratory chain with its laboratories	Public
PUT/PATCH	`/api/chain-laboratories/{id}`	Update a laboratory chain	Admin Only
DELETE	`/api/chain-laboratories/{id}`	Delete a laboratory chain	Admin Only

➢ Departments , Hospitals and Care centers Endpoints

Endpoint	Method	Access	Description
`/api/hospitals`	GET	Public	List all hospitals
`/api/hospitals`	POST	Admin	Create a new hospital
`/api/hospitals/{id}`	GET	Public	View hospital with its departments
`/api/hospitals/{id}`	PUT	Admin	Update hospital information
`/api/hospitals/{id}`	DELETE	Admin	Delete a hospital
`/api/department-hospital`	POST	Admin	Link department to hospital
`/api/department-hospital/{departmentId}/{hospitalId}`	PUT	Admin	Update relationship details
`/api/care-centers`	GET	Public	List all care centers
`/api/care-centers`	POST	Admin	Create a new care center
`/api/care-centers/{id}`	GET	Public	View care center with its departments
`/api/care-centers/{id}`	PUT	Admin	Update care center information
`/api/care-centers/{id}`	DELETE	Admin	Delete a care center
`/api/care-center-department`	POST	Admin	Link department to care center
`/api/care-center-department/{departmentId}/{careCenterId}`	PUT	Admin	Update relationship details

Front-end design and implementation

1.Application Architecture

The application is divided into three main modules:

1. **Public Site Module**: Accessible to all users, contains home page, authentication, and browsing features.
2. **Doctor Panel Module**: Restricted to authenticated doctors, provides tools for managing appointments and content.
3. **Admin Panel Module**: Restricted to administrators, provides system management capabilities.
   Each module is ***lazy-loaded*** to optimize performance and only loads when needed.

2.Single Page Application (SPA)

MediCare is built as an SPA using Angular, meaning all content is dynamically loaded without full page reloads. This results in faster navigation, reduced server load, and a smoother user experience, especially important in healthcare where users expect efficiency.

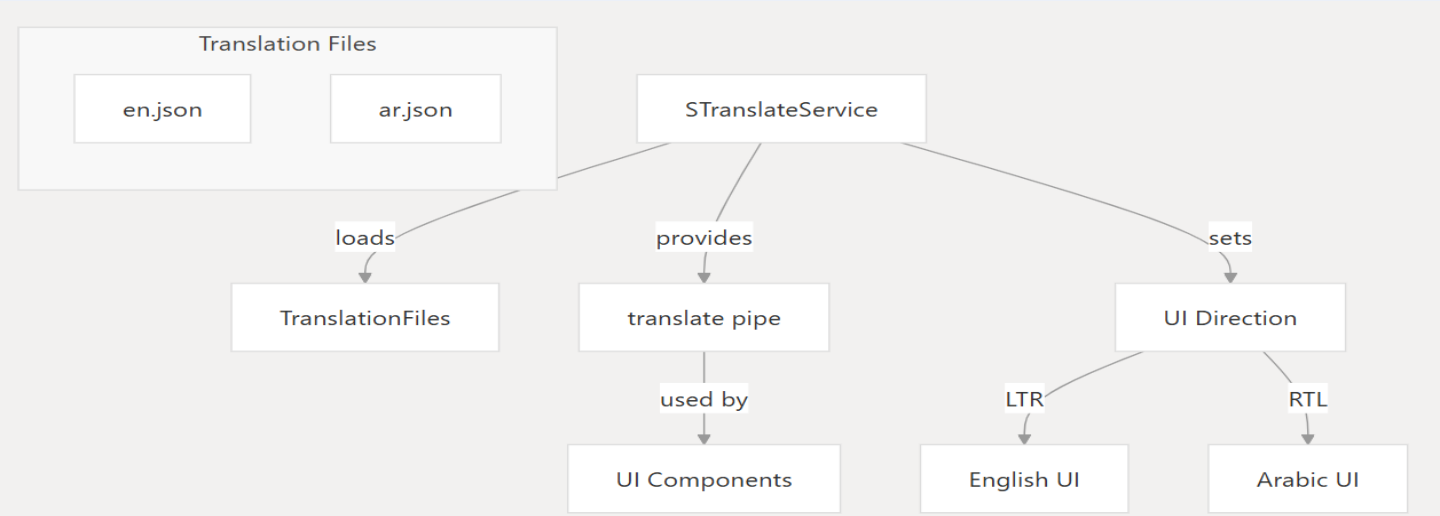
3.Server-Side Rendering (SSR) Support

MediCare uses Angular Universal for SSR to improve performance and SEO.
Pages are rendered on the server before being sent to the browser, leading to:

- Faster initial load.
- Better SEO (especially for dynamic routes).
- Enhanced sharing previews.

4. Internationalization (i18n)

MediCare supports multiple languages (English and Arabic) with a comprehensive internationalization system:



The i18n system includes:

1. **Translation Service**: Core service for managing language settings
2. **Translation Files**: JSON files with key-value pairs for each language
3. **Translation Pipe**: Used in templates for translating text
4. **Bidirectional Support**: Handles both LTR (English) and RTL (Arabic) text directions

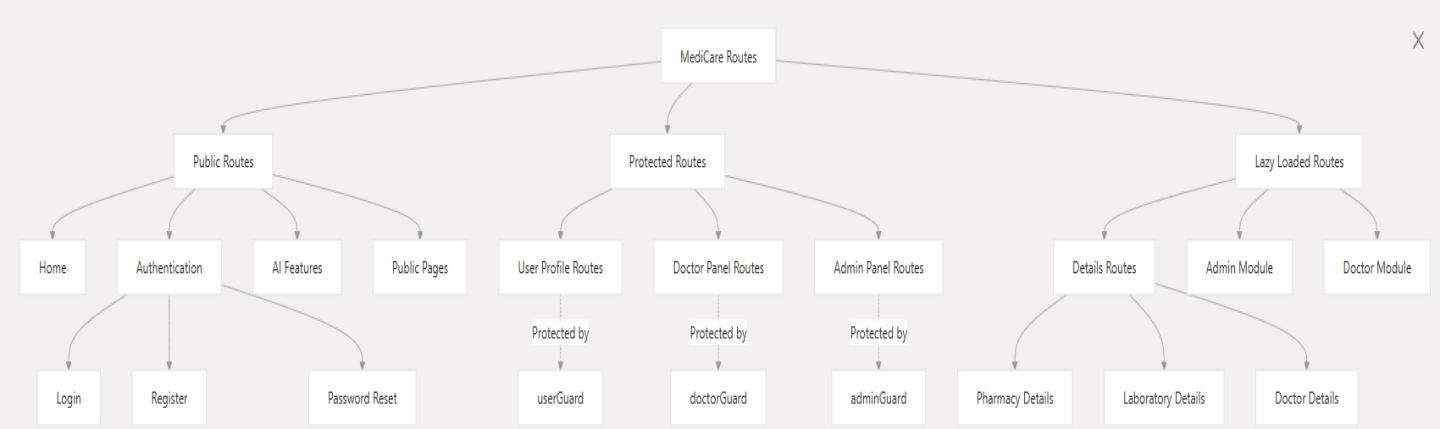
The application uses **@ngx-translate** for translation management, configured in the application's module.

5-Progressive Web App (PWA)

- *Installable Like a Native App*
  Users can install MediCare directly from their browser onto their ***mobile*** or ***desktop devices***, creating a native-like experience without needing to visit an app store.
- *Offline Access*
  Using Angular Service Workers, MediCare caches essential assets and data, allowing users to access previously visited pages and key functionality even when offline—critical for areas with limited connectivity.
- *Fast Load Times*
  PWAs load instantly thanks to pre-cached resources, offering a seamless experience that rivals native apps.

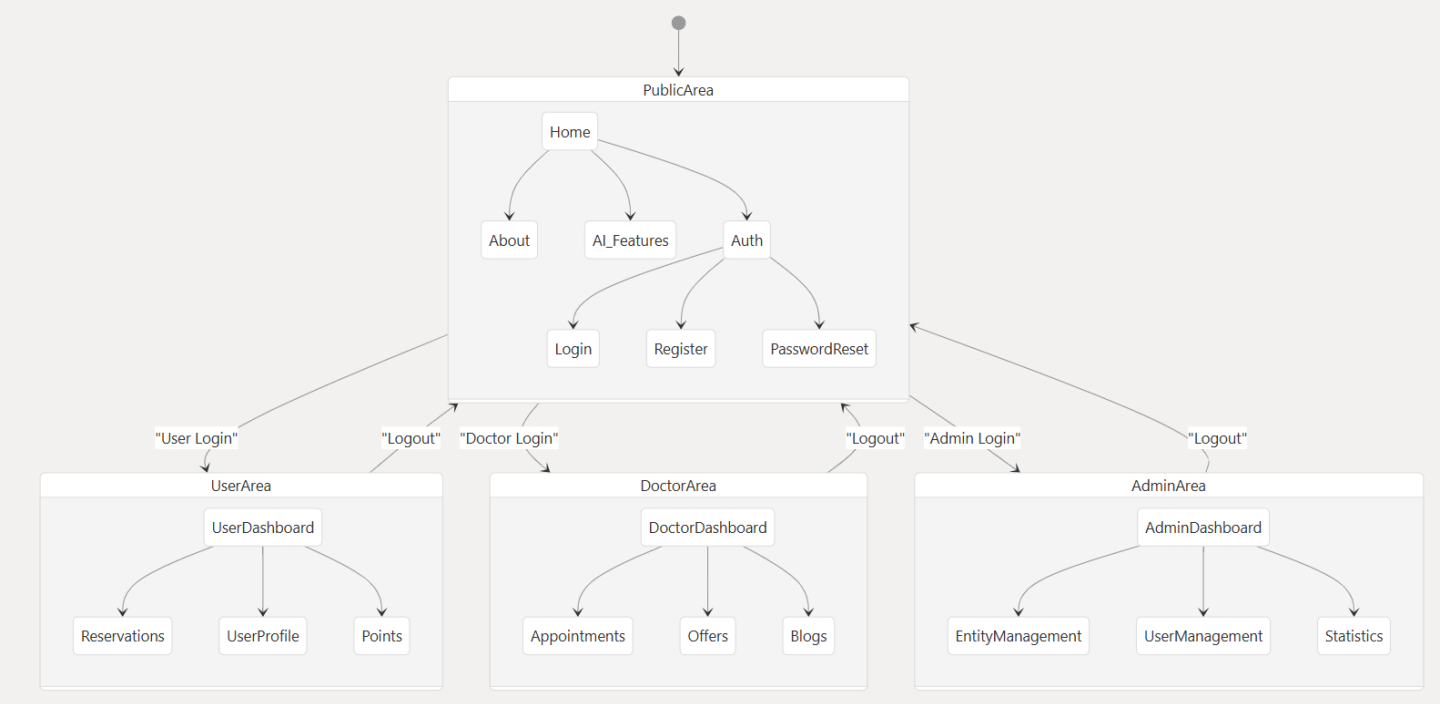
6.Routing and Navigation

The Medicare application uses Angular's routing system to organize its navigation into distinct categories based on user roles and functionality. The application's routes are hierarchically structured to support clear separation of concerns and role-based access control.



Role-Based Navigation Flow

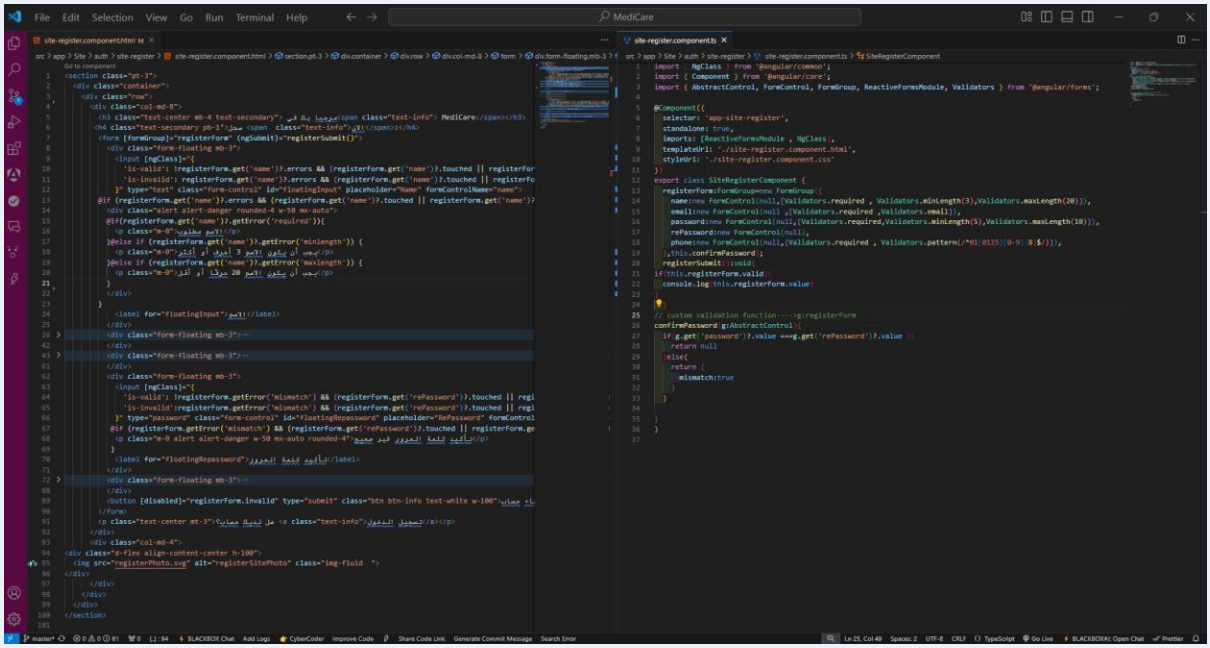
The application implements distinct navigation flows based on user roles, ensuring that users can only access appropriate functionality.



Web Site Design

1-Register Page

Used Code



Used Code

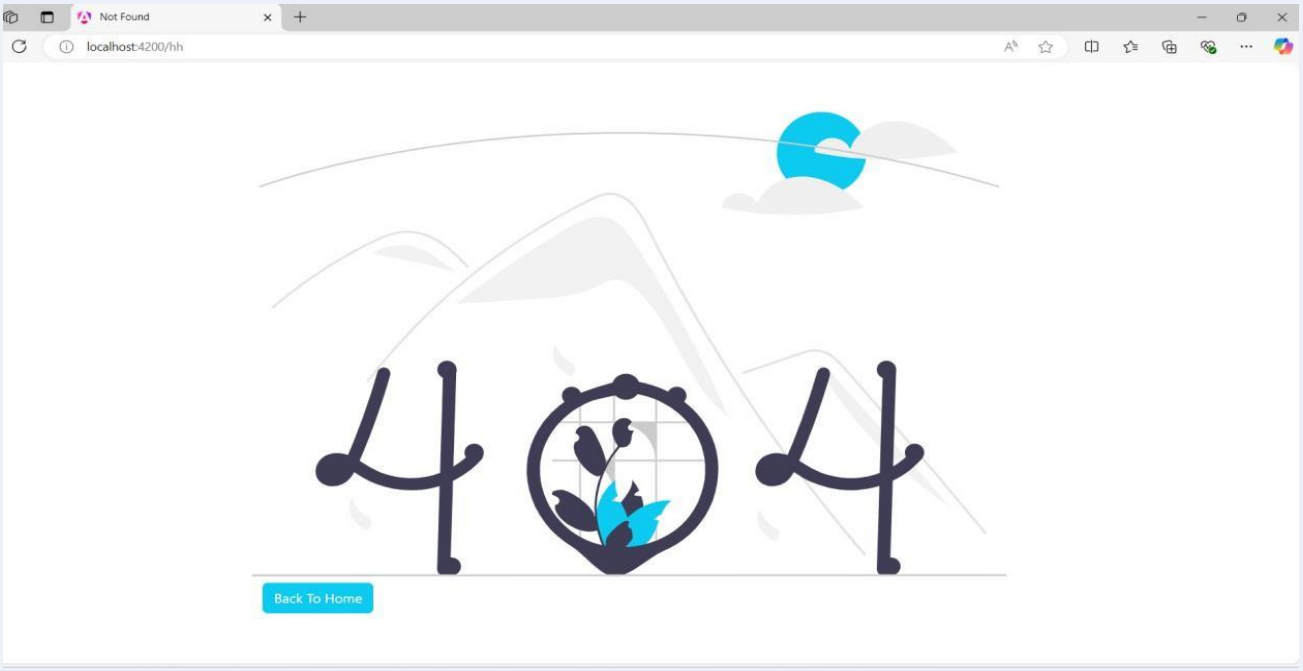
```
<section id="departments">
  <div class="container text-center">
    <h3 class="head-title">الأقسام الطبية</h3>
  </div>

  <div class="container mt-4">
    <p-carousel id="department-carousel" class="py-5" [value]="Departments" [numVisible]="4"
[circular]="true"
      [responsiveOptions]="responsiveOptions">
      <ng-template let-department pTemplate="item">
        <div class="border shadow-sm border-0 rounded-border m-2">
          <div class="relative mx-auto">
            <img [src]="department.imageUrl" class=" d-block mx-auto rounded-border" />
          </div>
          <h5 class="title">{{ department.title }}</h5>
        </div>
      </ng-template>
    </p-carousel>
  </div>
</section>
```

Used Code

```
<!-- Start Pharmacy Section -->
<section id="Pharmacy">
  <div class="container text-center" data-aos="fade-up" data-aos-duration="1000">
    <h2 class="head-title dark-text">{{"Pharmacies.Pharmacies"|translate}}</h2>
  </div>
  <div class="container my-5">
    <div class="row">
      <div class="col-md-6 my-2" data-aos="fade-right" data-aos-duration="1000">
        <a routerLink="/all/pharmacies">
          <img src="./images/pharmacy2.png" alt="صورة صيدلية" class="img-fluid" loading="lazy">
        </a>
      </div>
      <div class="col-md-6 d-flex flex-column justify-content-center" data-aos="fade-left" data-aos-duration="1000">
        <h3 class="dark-text"> <span>MediCare</span>{{"Pharmacies.contentPharm"|translate}}
        </h3>
        <a routerLink="/all/pharmacies"
          class="btn btn-outline-info d-block fs-5 mx-auto py-2 px-4 d-flex align-items-center rounded-4">
          <i class="bi bi-arrow-left px-2"></i>
          {{"Pharmacies.explorePharmacies"|translate}}
        </a>
      </div>
    </div>
  </div>
</section>
<!-- End Pharmacy Section -->
```

Used Code

```
<section id="DocOffers">
  <div class="container text-center">
    <h3 class="head-title">عروض الأطباء</h3>
  </div>

  <div class="container my-4">
    <p-carousel id="docOffers-carousel" class="py-5" [value]="DocOffers" [numVisible]="4"
[circular]="true"
      [responsiveOptions]="responsiveOptions">
      <ng-template let-docOffer pTemplate="item">
        <div class="docOffer-item shadow-sm p-6 border border-0 rounded-border m-2">
          <div class="info">
            <h6>25%</h6>
            <h6>OFF</h6>
          </div>
          <div class="relative mx-auto">
            <img [src]="docOffer.imageUrl" class="w-100 rounded-border" />
          </div>
          <h5 class="title">{{ docOffer.title }}</h5>
        </div>
      </ng-template>
    </p-carousel>
  </div>
</section>

```



Used Code

```
<div>
  <div class=
"vh-75 d-flex justify-content-center align-items-center">
    <img src="not-found.svg"  class="w-100"  alt=
"not-found">
  </div>
  <div class="container">
    <div class="row">
      <div class="col-md-12 mt-2">
        <button routerLink="/home" class=
"btn btn-info text-white">Back To Home</button>
      </div>
    </div>
  </div>
</div>
```


MediCare
خدمات الذكاء الاصطناعي
تحليل الفحوصات المختبرية
تحليل الوصفات الطبية
تحليل الأعراض
SECTION OF AI

```
<div class="container-fluid py-5">
  <!-- AI Features Section -->
  <section class="ai-features-section py-5" [attr.dir]="direction">
    <div class="container" data-aos="fade-up" data-aos-duration="1000">
      <h2 class="text-center mb-5 head-title dark-text">{{ 'ai-features.title' | translate }}</h2>
      <div class="row g-4">

        <!-- Feature Cards -->
        <div class="col-md-4" *ngFor="let feature of features" >
          <div class="card ai-feature-card">
            <div class="card-img-top position-relative">
              <img [src]="feature.image" class="img-fluid feature-image" [alt]="feature.alt">
              <div class="icon-overlay"><i class="{{ feature.icon }}"></i></div>
            </div>
            <div class="card-body text-center">
              <h5 class="card-title dark-text">
                <i class="{{ feature.iconClass }} me-2"></i>
                {{ feature.title | translate }}
              </h5>
              <p class="card-text dark-text">
                {{ feature.description | translate }}
              </p>
              <a [routerLink]="feature.link" class="btn btn-info">
                <i class="fas fa-arrow-circle-right me-2"></i> {{ 'ai-features.startNow' | translate }}
              </a>
            </div>
          </div>
        </div>

      </div>
    </div>
  </section>
</div>

```

```dart
import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';

class RobotCard extends StatelessWidget {
  final String title;
  final String description;
  final String image;
  final VoidCallback onTap;

  const RobotCard({
    required this.title,
    required this.description,
    required this.image,
    required this.onTap,
    super.key,
  });

  @override
  Widget build(BuildContext context) {
    return Card(
      elevation: 5,
      color: Theme.of(context).colorScheme.surface, // خلفية الكارد من الثيم
      surfaceTintColor: Theme.of(context).colorScheme.primary, // لون التمييز
      child: InkWell(
        onTap: onTap,
        child: Column(
          mainAxisSize: MainAxisSize.min,
          children: [
            SizedBox(
              height: 150.h,
              width: double.infinity,
              child: Image.asset(
                image,
                fit: BoxFit.cover,
                errorBuilder: (context, error, stackTrace) {
                  return Icon(
                    Icons.image,
                    size: 50.sp,
                    color:
                        Theme.of(context).colorScheme.primary, // لون من
الثيم
                  );
                },
              ),
            ),
            SizedBox(height: 10.h),
            Text(
              title,
              style: Theme.of(context).textTheme.titleLarge?.copyWith(
                    fontSize: 20.sp,
                  ), // نمط النص من الثيم مع تعديل الحجم
              textAlign: TextAlign.center,
            ),
            SizedBox(height: 10.h),
            Padding(
              padding: const EdgeInsets.symmetric(horizontal: 8.0),
              child: Text(
                description,
                textAlign: TextAlign.center,
                style: Theme.of(context).textTheme.titleMedium?.copyWith(
                      fontSize: 16.sp,
                    ), // نمط النص من الثيم مع تعديل الحجم
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

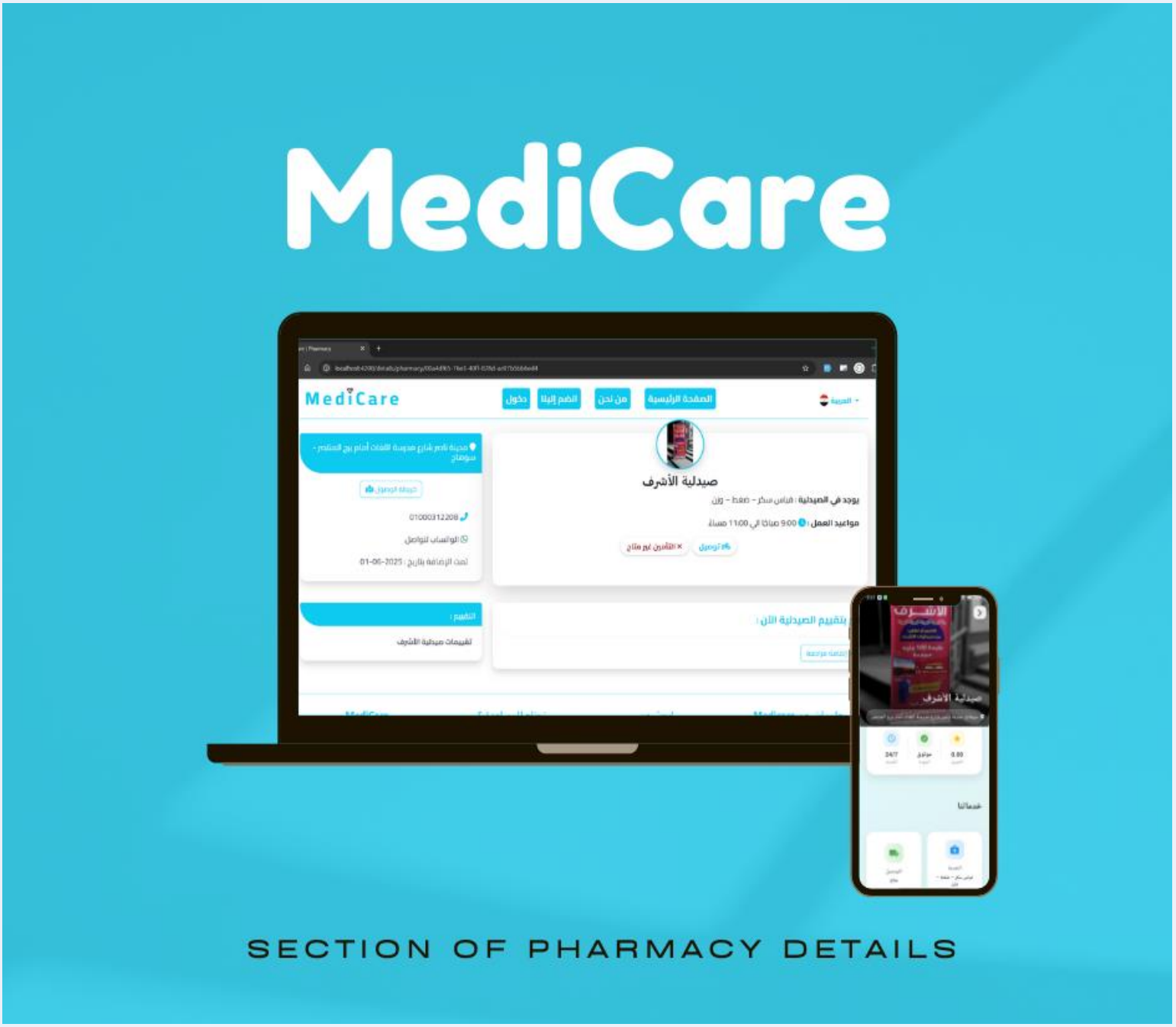

MediCare
SECTION OF PHARMACIES

```
<div class="container mt-2" [ngStyle]="{'direction': isRtl ? 'rtl' : 'ltr'}">
  <form (ngSubmit)="handleSubmit($event)" class="card bg-dark-el border border-info border-1">
    <div class="form-group p-3">
      <div class="row">
        <div class="col-sm-6 my-1">
          <div class="dropdown ">
            <button class="btn btn-info text-light dropdown-toggle w-100" type="button" id="dropdownMenuButton1"
              data-bs-toggle="dropdown" aria-expanded="false">
              {{"Pharmacies.pharmaciesChain"|translate}}
            </button>
            <ul class="dropdown-menu" aria-labelledby="dropdownMenuButton1">
              <li>
                <button class="dropdown-item" (click)="loadPharmaciesByChain('all')">
                  {{"Pharmacies.all"|translate }}
                </button>
              </li>
              @for (item of ChainsPharmacies; track $index) {
              <li>
                <button class="dropdown-item" (click)="loadPharmaciesByChain(item.id)">
                  {{ item.title }}
                </button>
              </li>
              }
            </ul>
          </div>
        </div>
        <div class="col-sm-6 my-1">
          <input type="text" class="form-control w-100" placeholder="ابحث عن صيدلية..." [(ngModel)]="searchQuery"
            name="search" />
        </div>

      </div>
      <div class="row">
        <div class="col-6 my-1">
          <label class="form-label dark-text">{{"Pharmacies.city"|translate }}</label>
          <input type="text" class="form-control" name="city" (input)="handleCityChange($event)" />
        </div>
        <div class="col-6 my-1">
          <label class="form-label dark-text">{{"Pharmacies.area"|translate }}</label>
          <input type="text" class="form-control" name="area" (input)="handleAreaChange($event)" />
        </div>
      </div>
      <div class="row">
        <div class="col-4 my-1">
          <label class="form-label dark-text">{{"Pharmacies.delivery"|translate }}</label>
          <select class="form-select" name="deliveryOption" (change)="handleDeliveryOptionChange($event)">
            <option value="all">{{"Pharmacies.all"|translate }}</option>
            <option value="1">{{"Pharmacies.available"|translate }}</option>
            <option value="0">{{"Pharmacies.notAvailable"|translate }}</option>
          </select>
```

```
</div>
        <div class="col-4 my-1">
          <label class="form-label dark-text">{{"Pharmacies.insurance"| translate}}</label>
          <select class="form-select" (change)="handleInsurenceChange($event)" name="insurence">
            <option value="all">{{"Pharmacies.all"|translate }}</option>
            <option value="1">{{"Pharmacies.available"|translate }}</option>
            <option value="0">{{"Pharmacies.notAvailable"|translate }}</option>
          </select>
        </div>
        <div class="col-4 my-1">
          <label class="form-label dark-text">{{"Pharmacies.rating"|translate }}</label>
          <input type="number" min="0" max="5" class="form-control" name="minRate"
            (change)="handleMinRateChange($event)" />
        </div>
        <div class="col-sm-2 my-1 mx-auto">
          <button class="btn btn-info text-light w-100" type="submit">
            {{"Pharmacies.search"|translate}}
          </button>
        </div>
      </div>
    </div>
  </form>
  @if (isFetching()) {
  <div class="row justify-content-center mt-4">
    <div class="col-12 text-center">
      <div class="spinner-border text-info" role="status">
        <span class="visually-hidden">Loading...</span>
      </div>
    </div>
  </div>
  } @else {
  <div class="container my-5">
    <div class="row" dir="rtl">
      @for (item of Pharmacies; track $index) {
      <div class="col-md-6 col-xl-4 my-2" data-aos="fade-up" data-aos-duration="1000">
        <div class="card-top text-center bg-dark-el shadow-lg border-0 overflow-hidden">

          <!-- الجزء العلوي -->
          <div class="card-header bg-info text-white py-4" style="border-bottom-left-radius: 50px;">
            <i class="fas fa-prescription-bottle-medical fa-3x"></i>
            <h4 class="fw-bold mt-2">{{item.title}}</h4>
            <p class="mb-2">{{ item.service }}</p>
          </div>

          <!-- الصورة -->
          <div class="position-relative bg-dark-el p-3 text-center">
            <img [src]="item.image ? item.image : './images/pharmacy.gif'"
              class="rounded-circle border border-4 border-light shadow"
              style="width: 100px; height: 100px; margin-top: -50px;" alt="pharmacy Image" />

```

```
  @if (item.insurence ===1) {
              <p class="insurence pb-2 m-0">
                {{"Pharmacies.insurance"| translate}}
                <i class="bi bi-shield-check"></i>
              </p>
              }
              @if (item.deliveryOption ===1) {
              <p class="delivery m-0 pb-2">
                {{"Pharmacies.delivery"| translate}}
                <i class="fa-solid fa-motorcycle fs-4"></i>
              </p>
              }

            </div>

            <!-- التفاصيل -->
            <div class="card-body px-2">
              <div class="d-flex flex-column justify-content-between align-content-center">
                <p class="fw-bold dark-text">
                  <i class="fa-solid fa-map-marker-alt text-info px-2">
                  </i> {{ item.area }}
                  <!-- - {{item.city}} -->
                </p>
                <p class="fw-bold dark-text"> {{ item.phone }} <i class=
"fa-solid fa-phone text-info px-2"></i></p>
              </div>
              <div class="d-flex justify-content-between align-content-center">
                <a routerLink="/details/pharmacy/{{ item.id }}"
                  class="btn btn-info text-white fw-bold w-50 rounded-pill shadow-sm mt-1 mx-2">
                  {{ "depDetails.BtnInfo" | translate }}
                </a>
                <a (click)="showInMap(item.locationUrl)"
                  class="btn btn-info text-white fw-bold w-50 rounded-pill shadow-sm mt-1 " title
="Location">
                  <i class="bi bi-geo-alt text-light"></i>
                </a>
              </div>
            </div>
          </div>
        </div>
        }@empty {
        <div class="not-found">
          <img src="./images/empty.png" alt="" class="img-not-found" />
          <h3 class="head-not-found">{{"notFound.notFoundPage"|translate }}</h3>
          <h6 class="head-not-found2">
            {{"notFound.notFoundPage2"|translate }}
          </h6>
        </div>
        }
      </div>
      <div class="d-flex justify-content-center mt-4">
        <button class="btn btn-outline-info mx-1" (click)="handlePageChange(currentPage - 1)"
          [disabled]="currentPage === 1">
          {{"depDetails.lastPage"|translate}}
        </button>
        @for (page of [].constructor(totalPages); track $index) {
        <button class="btn btn-outline-info mx-1" (click)="handlePageChange($index + 1)"
          [class.active]="$index + 1 === currentPage">
          {{$index + 1}}
        </button>
        }
        <button class="btn btn-outline-info mx-1" (click)="handlePageChange(currentPage + 1)"
          [disabled]="currentPage === totalPages || totalPages === 0">
          {{"depDetails.nextPage"|translate}}
        </button>
      </div>
    </div>
    }
</div>
```

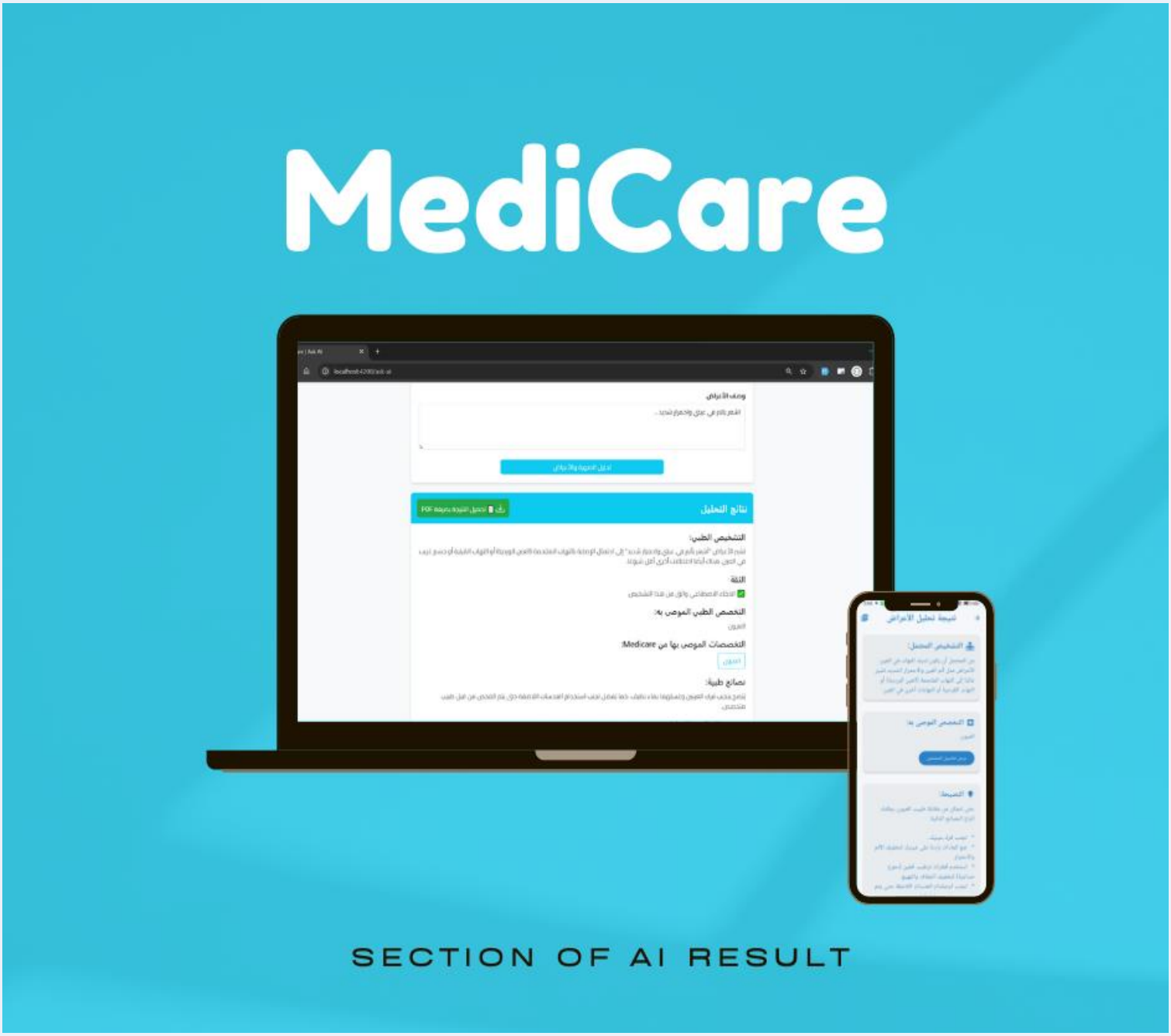
```dart
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import '../../pharmacies/manager/cubit/pharmacy_cubit.dart';
import 'package:media_care/core/utils/widgets/custom_circular_indicator.dart';

class PharmacyListView extends StatelessWidget {
  PharmacyListView({super.key});

  @override
  Widget build(BuildContext context) {
    return Container(
      height: 210.h,
      child: BlocBuilder<PharmacyCubit, PharmacyState>(
        builder: (context, state) {
          if (state is PharmacyLoadingState) {
            return CustomProgressIndicator();
          } else if (state is PharmacySuccessState) {
            final pharmacy = state.pharmacies;
            return ListView.builder(
              scrollDirection: Axis.horizontal,
              itemCount: pharmacy.length,
              itemBuilder: (context, index) {
                return Padding(
                  padding:
                      const EdgeInsets.symmetric(horizontal: 10.0, vertical: 8),
                );
              },
            );
          } else {
            return Center(
              child: Text(
                'لا توجد بيانات متاحة للعرض',
                style: TextStyle(
                  color: Theme.of(context).colorScheme.primary,
                  fontSize: 18.sp,
                ),
              ),
            );
          }
        },
      ),
    );
  }
}
```


MediCare
MediCare
الصفحة الرئيسية
من نحن
انضم إلينا
دخول
صيدلية الأشرف
SECTION OF PHARMACY DETAILS

```
<div class="container py-4">
  <p-toast />
  @if (isFetching()) {
  <div class="row justify-content-center">
    <div class="col-12 text-center">
      <div class="spinner-border text-info" role="status">
        <span class="visually-hidden">Loading...</span>
      </div>
    </div>
  </div>
  } @else {
  <div class="row my-2" data-aos="fade-up" data-aos-duration="1000">
    <div class="col-md-4 order-2 order-md-1">
      <div class="card bg-dark-el card-info border border-0 shadow">
        <div class="card-header bg-info py-3 d-flex justify-content-start align-items-center">
          <h3 class="dark-text">
            <i class="fa-solid fa-location-pin"></i>
            {{ pharmacy.area }} - {{ pharmacy.city }}
          </h3>
        </div>
        <button class="btn btn-outline-info my-3 mx-auto d-block rounded-3"
          (click)="showInMap(pharmacy.locationUrl)">خريطة الوصو
          <i class="fa-solid fa-map-location-dot"></i></button>
        <div class="more-info">
          <p class="pharmacy-phone dark-text">
            <a href="tel:{{ pharmacy.phone }}"><i class="fa-solid fa-phone-flip text-info"></i
></a>
            {{ pharmacy.phone }}
          </p>
          <p class="pharmacy-whatsapp dark-text">
            <a href="{{ pharmacy.whatsappLink }}"><i class="fa-brands fa-whatsapp text-success"
></i>

            </a>
            الواتساب لتواصل
          </p>
          <p class="pharmacy-whatsapp dark-text">
            تمت الإضافة بتاريخ : {{ pharmacy.created_at | date:'yyyy-MM-dd'}}
          </p>
        </div>
      </div>
    </div>
    <div class="col-md-8 order-1 order-md-2">
      <div class="card bg-dark-el pharmacy-card shadow-lg border-0">
        <div class="pharmacy-image">
          <img [src]="pharmacy.image ? pharmacy.image : './images/pharmacy.gif'"
            class="rounded-circle border border-3 border-info shadow" style="
width: 100px; height: 100px;"
            alt="Pharmacy Image">
        </div>
        <div class="card-body">
          <h3 class="text-center fw-bold dark-text">{{ pharmacy.title }}</h3>
          <p class="pharmacy-service dark-text"> <span class="fw-bold">
              يوجد في الصيدلية
          </span> : {{ pharmacy.service }}</p>
          <p class="pharmacy-date dark-text">
            <span class="fw-bold dark-text">مواعيد العمل :  span>
            <i class="fa-solid fa-clock text-info"></i>
            {{ pharmacy.start_at | timeFormat }} {{"details.to" | translate}}
            {{ pharmacy.end_at | timeFormat}}
          </p>
          <div class="d-flex justify-content-center">
            <p class="pharmacy-delivery px-2">
              <!-- <span class="fw-bold">خدمة التوصيل : </span> -->
              @if (pharmacy.deliveryOption === 1) {
              <span class="delivery m-0 pb-1">
                <i class="fa-solid fa-motorcycle fs-6"></i>
                {{"Pharmacies.delivery" | translate}}
              </span>
              }@else{
              <span class="notAvaliabledelivery m-0 pb-1">
                <i class="fa-solid fa-motorcycle fs-6"></i>
                التوصيل غير متاحة
              </span>
              }
            </p>
            <p class="pharmacy-insurence">
              <!-- <span class="fw-bold">خدمة التأمين : </span> -->
              @if (pharmacy.insurence === 1) {
              <span class="insurence m-0 pb-1">
                <i class="bi bi-shield-check fs-6"></i>
                {{"Pharmacies.insurance" | translate}}
              </span>
              }@else{
              <span class="notAvaliableinsurence m-0 pb-1">
                <i class="fa-solid fa-xmark fs-6"></i>
                التامين غير متاح
              </span>
              }
            </p>
          </div>
```

```
<div class="companies-wrapper dark-text mx-auto">
              <div class="companies-grid">
                @for(item of insuranceCompanies; track $index) {
                <div class="company-item text-center">
                  <img src="{{ item.logo }}" width="58" height="58" class="rounded-circle mb-1"
alt="">
                  <span class="d-block small-text">{{ item.name }}</span>
                </div>
                }
              </div>
            </div>

          </div>
        </div>
      </div>
    </div>
    <div class="row" data-aos="fade-up" data-aos-duration="1000">
      <div class="col-md-4 order-2 order-md-1">
        <div class="card bg-dark-el card-rating border border-0 shadow">
          <div class=
"card-header bg-info py-2 d-flex justify-content-start align-items-center rating-container"
            [ngStyle]="{'direction': isRtl ? 'rtl' : 'ltr'}">
            <p class="ratings m-0">{{"details.rating" | translate}} :</p>
            <div class="stars ms-2">
              <ng-container *ngFor="let star of getStarArray()">
                <i class="fa-solid fa-star text-warning"></i>
              </ng-container>
            </div>
          </div>

          <!-- عرض التقييمات -->
          <div class="ratings-container p-3">
            <h3 class="fw-bold dark-text">تقييمات {{ pharmacy.title }} </h3>
            <div *ngFor="let rating of ratesOfPharmacy"
              class="rating-card bg-dark-el dark-text p-2 shadow-sm rounded-3 my-2">
              <div class="d-flex align-items-center justify-content-center mb-2">
                <i *ngFor="let star of [].constructor(rating.pivot.rating_value); let i = index"
                  class="fa-solid fa-star text-warning me-1"></i>
                <span *ngIf="rating.pivot.rating_value < 5">
                  <i *ngFor="let star of [].constructor(5 - rating.pivot.rating_value)"
                    class="fa-solid fa-star text-muted"></i>
                </span>
              </div>
              <p class="my-1"><strong>مراجعة :</strong> {{ rating.pivot.review }}</p>

              <p class="text-muted dark-text small p-0 my-1" style="font-size: 12px;">
                <i class="fa-solid fa-user"></i> {{ rating.name }}
              </p>
              <p class="text-muted small my-1 dark-text"><i class="fa-solid fa-clock"></i> {{
 rating.pivot.created_at |
                date :
                "short" }}
              </p>
            </div>
          </div>
        </div>
      </div>
      <div class="col-md-8 order-1 order-md-2">
        <div class="card bg-dark-el card-rating border border-0 shadow">
          <div class=
"card-header bg-dark-el text-info py-3 d-flex justify-content-start align-items-center">
            <h3 class="dark-text">
              قم بتقييم الصيدلية الآن :
            </h3>
          </div>
          <!-- إضافة مراجعة -->
          <div class="d-flex justify-content-start">
            <div class="review-section p-3">
              <button class="btn btn-outline-info w-100" (click)="toggleReviewInput()">
                <i class="fas fa-pencil-alt"></i> إضافة مراجعة
              </button>

              <div *ngIf="showReviewInput" class="mt-3">
                <form [formGroup]="reviewForm" (ngSubmit)="submitReview(reviewForm)">
                  <div class="d-flex justify-content-center mb-2">
                    <ng-container *ngFor="let star of stars; let i = index">
                      <i class="fa-solid fa-star rating-star"
                        [class.filled]="i < (reviewForm.get('rating_value')?.value || 0)" (click)=
"setRating(i + 1)">
                      </i>
                    </ng-container>
                  </div>

                  <textarea formControlName="review" class="form-control mb-2" rows="3"
                    placeholder="اكتب مراجعتك هنا..."></textarea>
                  <button type="submit" [disabled]="!isAuth" class="btn btn-info text-white w-100"
>إرسال المراجعة</button>
                </form>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  }
</div>
```

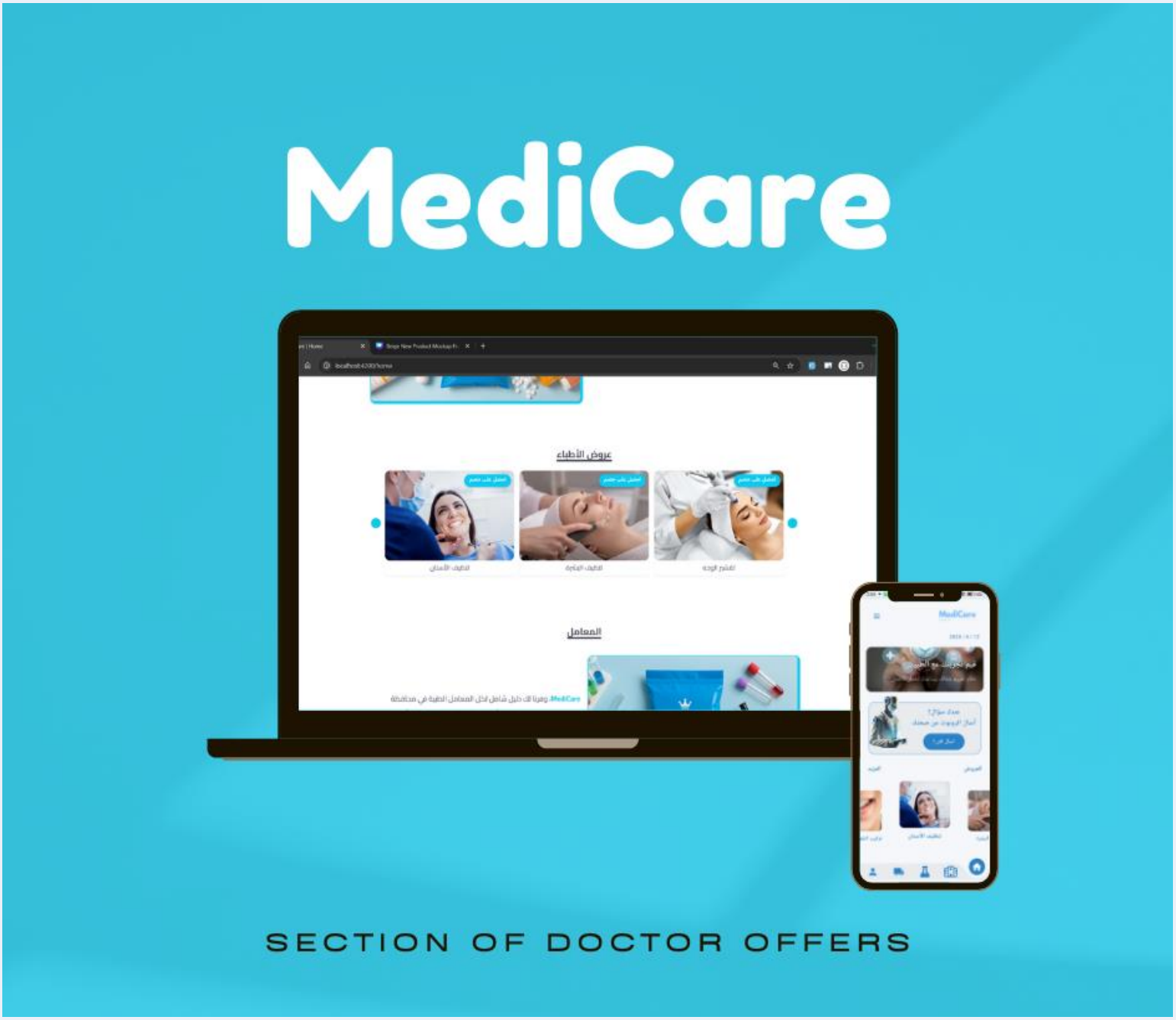
```dart
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:media_care/core/utils/widgets/custom_shimmer_list_view.dart';
import
import 'package:media_care/features/pharmacies/widgets/pharmacy_list_view.dart';

import '../manager/cubit/pharmacy_cubit.dart';

class pharmacyViewBody extends StatelessWidget {
  const pharmacyViewBody({
    super.key,
    required ScrollController scrollController,
  }) : _scrollController = scrollController;
  final ScrollController _scrollController;
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Theme.of(context).scaffoldBackgroundColor,
      appBar: AppBar(
        forceMaterialTransparency: true,
        title:
            Text('الصيدليات ', style: Theme.of(context).textTheme.titleLarge),
        centerTitle: true,
        iconTheme: Theme.of(context).appBarTheme.iconTheme,
      ),
      body: BlocBuilder<PharmacyCubit, PharmacyState>(
        builder: (context, state) {
          if (state is PharmacyLoadingState) {
            return Shimmer.fromColors(
              baseColor: Theme.of(context).colorScheme.primary,
              highlightColor: Theme.of(context).colorScheme.surface,
              child: const CustomShimmerListView(),
            );
          } else if (state is PharmacySuccessState) {
            return CustomPharmacyListView(
                pharmacies: state.pharmacies,
                scrollController: _scrollController);
          } else if (state is PharmacyErrorState) {
            return ErrorText(message: state.message);
          }
          return Shimmer.fromColors(
            baseColor: Theme.of(context).colorScheme.primary,
            highlightColor: Theme.of(context).colorScheme.surface,
            child: const CustomShimmerListView(),
          );
        },
      ),
    );
  }
}

class ErrorText extends StatelessWidget {
  final String message;

  const ErrorText({super.key, required this.message});

  @override
  Widget build(BuildContext context) {
    return Center(child: Text(message));
  }
}
```


MediCare
SECTION OF AI RESULT

```
<!-- Start Navbar -->
<app-site-navbar></app-site-navbar>

<!-- Main Container -->
<div class="details-container bg-dark-el" [attr.dir]="isRtl ? 'rtl' : 'ltr'" data-aos="fade-up" data-aos-duration="1000">
  <div class="content-wrapper">
    <!-- Card: Upload and Analyze -->
    <div class="details-card bg-dark-el">
      <div class="details-header">
        <h2>{{ 'ai.titleOfSection' | translate }}</h2>
      </div>
      <div class="details-body">
        <!-- File Upload Section -->
        <div class="mb-4">
          <label for="fileInput" class="form-label dark-text">{{ 'ai.chooseFile' | translate }}</label>
          <input type="file" class="form-control" id="fileInput" accept="image/*" (change)="onFileSelected($event)" />

          <!-- File Preview -->
          <div *ngIf="previewUrl" class="mt-3 text-center">
            <img [src]="previewUrl" class="img-thumbnail" style="max-height: 200px;" [alt]="'ai.imageAnalysisResult' |
translate" />
            <button class="btn btn-danger btn-sm mt-2" (click)="clearFile()">
              {{ 'ai.removeFile' | translate }}
            </button>
          </div>

          <!-- Upload Progress -->
          <div *ngIf="isUploading" class="mt-3">
            <div class="progress">
              <div class="progress-bar progress-bar-striped progress-bar-animated bg-info" role="progressbar"
                [style.width]="uploadProgress + '%'" [attr.aria-valuenow]="uploadProgress" aria-valuemin="0" aria-valuemax=
"100">
                {{ uploadProgress }}%
              </div>
            </div>
          </div>
        </div>

        <!-- Symptoms Input -->
        <div class="mb-4">
          <label for="symptomsInput" class="form-label dark-text">{{ 'ai.describeSymptoms' | translate }}</label>
          <textarea id="symptomsInput" class="form-control" rows="4" [(ngModel)]="symptomsText"
            [placeholder]="'ai.enterSymptoms' | translate"></textarea>
        </div>

        <!-- Analyze Button -->
        <div class="text-center">
          <button class="btn btn-info w-50 text-light" (click)="analyzeContent()" [disabled]="loading || (!symptomsText && !
selectedFile)">
            <span *ngIf="!loading">{{ 'ai.imageAnalysis' | translate }}</span>
            <span *ngIf="loading" class="spinner-border spinner-border-sm" role="status" aria-hidden="true"></span>
            <span *ngIf="loading"> {{ 'ai.analyzing' | translate }}</span>
          </button>
        </div>
      </div>
    </div>

```

```
<!-- Card: Analysis Result -->
    <div class="details-card bg-dark-el mt-4" *ngIf="showResult && (selectedFile || symptomsText) && apiResponse" id=
"analysisResult">
      <div class="details-header">
        <h2>{{ 'ai.analysisResult' | translate }}</h2>
        <button class="btn btn-success" (click)="generatePDF()">
          <i class="bi bi-download"></i> {{ 'ai.downloadPDF' | translate }}
        </button>
      </div>
      <div class="details-body">
        <!-- Image Analysis Result -->
        <div class="section" *ngIf="selectedFile">
          <h4 class="dark-text">{{ 'ai.imageAnalysisResult' | translate }}</h4>
          <p class="dark-text">{{ apiResponse.imageAnalysis || ('ai.noImageAnalysis' | translate) }}</p>
        </div>

        <!-- Diagnosis -->
        <div class="section" *ngIf="apiResponse.diagnosis">
          <h4 class="info-text">{{ 'ai.diagnosis' | translate }}</h4>
          <p class="dark-text">{{ apiResponse.diagnosis }}</p>
        </div>

        <!-- Confidence Message -->
        <div class="section" *ngIf="apiResponse.message">
          <h4 class="info-text">{{ (isRtl ? 'الثقة' : 'Confidence') }}</h4>
          <p class="dark-text">{{ apiResponse.message }}</p>
        </div>

        <!-- Recommended Specialization -->
        <div class="section">
          <h4 class="info-text">{{ 'ai.recommendedSpecialization' | translate }}</h4>
          <p class="dark-text">{{ apiResponse.recommendedSpecialization || ('ai.notSpecified' | translate) }}</p>
        </div>

        <!-- Medicare Recommendations -->
        <div class="section" *ngIf="recommendedDepartments.length > 0">
          <h4 class="info-text">{{ 'ai.medicareRecommendations' | translate }}</h4>
          <div class="d-flex flex-wrap gap-2">
            <button *ngFor="let department of recommendedDepartments" class="btn btn-outline-info"
              (click)="handleDepartmentSelection(department.id)">
              {{ department.title }}
            </button>
          </div>
        </div>

        <!-- Advice -->
        <div class="section">
          <h4 class="info-text">{{ 'ai.advice' | translate }}</h4>
          <p class="dark-text">{{ apiResponse.advice || ('ai.noAdvice' | translate) }}</p>
        </div>

        <!-- Confidence Level -->
        <div class="section" *ngIf="apiResponse.confidence_score">
          <h4 class="info-text">{{ 'ai.confidenceLevel' | translate }}</h4>
          <p class="dark-text">{{ apiResponse.confidence_score }}%</p>
        </div>

        <!-- Suggested Medications -->
        <div class="section" *ngIf="apiResponse.suggested_medications?.length">
          <h4 class="info-text">{{ 'ai.suggestedMedications' | translate }}</h4>
            <ul class="list-group">
              <li *ngFor="let medication of apiResponse.suggested_medications" class="list-group-item bg-dark-el dark-text">
                <strong>{{ 'ai.medicationName' | translate }}</strong> {{ medication.name }}<br>
                <strong>{{ 'ai.medicationDosage' | translate }}</strong> {{ medication.dosage }}<br>
                <strong>{{ 'ai.medicationNotes' | translate }}</strong> {{ medication.notes }}
              </li>
            </ul>
        </div>

        <!-- Medication Warning -->
        <div class="section text-danger" *ngIf="apiResponse.medication_warning">
          <h4>{{ 'ai.medicationWarning' | translate }}</h4>
          <p>{{ apiResponse.medication_warning }}</p>
        </div>
      </div>
    </div>

    <!-- Error Message -->
    <div class="error-message" *ngIf="errorMessage && !loading">
      <span class="material-icons">error</span>
      {{ errorMessage }}
    </div>
  </div>
</div>

<!-- Start Footer -->
<app-site-footer></app-site-footer>
```

```dart
import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';

class RobotCard extends StatelessWidget {
  final String title;
  final String description;
  final String image;
  final VoidCallback onTap;

  const RobotCard({
    required this.title,
    required this.description,
    required this.image,
    required this.onTap,
    super.key,
  });

  @override
  Widget build(BuildContext context) {
    return Card(
      elevation: 5,
      color: Theme.of(context).colorScheme.surface, // خلفية الكارد من الثيم
      surfaceTintColor: Theme.of(context).colorScheme.primary, // لون التمييز
      child: InkWell(
        onTap: onTap,
        child: Column(
          mainAxisSize: MainAxisSize.min,
          children: [
            SizedBox(
              height: 150.h,
              width: double.infinity,
              child: Image.asset(
                image,
                fit: BoxFit.cover,
                errorBuilder: (context, error, stackTrace) {
                  return Icon(
                    Icons.image,
                    size: 50.sp,
                    color:
                        Theme.of(context).colorScheme.primary, // لون من
الثيم
                  );
                },
              ),
            ),
            SizedBox(height: 10.h),
            Text(
              title,
              style: Theme.of(context).textTheme.titleLarge?.copyWith(
                    fontSize: 20.sp,
                  ), // نمط النص من الثيم مع تعديل الحجم
              textAlign: TextAlign.center,
            ),
            SizedBox(height: 10.h),
            Padding(
              padding: const EdgeInsets.symmetric(horizontal: 8.0),
              child: Text(
                description,
                textAlign: TextAlign.center,
                style: Theme.of(context).textTheme.titleMedium?.copyWith(
                      fontSize: 16.sp,
                    ), // نمط النص من الثيم مع تعديل الحجم
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```


MediCare
عروض الأطباء
المعامل
SECTION OF DOCTOR OFFERS

```
<section id="DocOffers">
  <div class="container text-center" data-aos="fade-up" data-aos-duration="1000">
    <h3 class="head-title dark-text">{{"docOffers.DoctorsOffers"|translate}}</h3>
  </div>
  <div class="container my-4">
    @defer(when isBrowser) {
    <p-carousel id="docOffers-carousel" class="py-5" [value]="DocOffers" [numVisible]="
3" [circular]="true"
      [responsiveOptions]="responsiveOptions">
      <ng-template let-docOffer pTemplate="item">
        <a routerLink="/all/doctor-offers/{{docOffer.id}}">
          <div class="docOffer-item shadow-sm p-6 border border-0 rounded-border m-2"
data-aos="fade-up"
            data-aos-duration="1000">
            <div class="info">
              <h6>{{"docOffer.secDiscount"|translate}}</h6>
            </div>
            <div class="relative mx-auto">
              <img [src]="docOffer.image" class="w-100 rounded-border" lazy="loading"
 />
            </div>
            <h5 class="title dark-text">{{ docOffer.title }}</h5>
          </div>
        </a>
      </ng-template>
    </p-carousel>
    }
    @placeholder {
    <div class="docOffers-carousel-placeholder">
      <div class="relative mx-auto">
        <div class="w-100 h-800px rounded-border d-block mx-auto bg-light"></div>
      </div>
    </div>
    }
  </div>
</section>

```

```dart
class DoctorOfferDetailsView extends StatefulWidget {
  final DoctorOffer doctorOffer;

  const DoctorOfferDetailsView({super.key, required this.doctorOffer});

  @override
  State<DoctorOfferDetailsView> createState() => _DoctorOfferDetailsViewState();
}

///-----------------------------------------------------------------------------

class _DoctorOfferDetailsViewState extends State<DoctorOfferDetailsView> {
  final PageController _pageController = PageController();

  @override
  Widget build(BuildContext context) {
    final hasMultipleImages = widget.doctorOffer.images.length > 1;

    return Scaffold(
      body: CustomScrollView(
        slivers: [
          SliverAppBar(
            expandedHeight: 300.h,
            floating: false,
            pinned: true,
            backgroundColor: Theme.of(context).colorScheme.surface,
            flexibleSpace: FlexibleSpaceBar(
              background: Stack(
                alignment: Alignment.bottomCenter,
                children: [
                  hasMultipleImages
                      ? PageView.builder(
                          controller: _pageController,
                          itemCount: widget.doctorOffer.images.length,
                          itemBuilder: (context, index) {
                            return CachedNetworkImage(
                              imageUrl: widget.doctorOffer.images[index].image,
                              fit: BoxFit.cover,
                              placeholder: (context, url) => const Center(
                                  child: CircularProgressIndicator()),
                              errorWidget: (context, url, error) => Icon(
                                  Icons.error,
                                  size: 50.sp,
                                  color: Colors.white),
                            );
                          },
                        )
                      : CachedNetworkImage(
                          imageUrl: widget.doctorOffer.images.isNotEmpty
                              ? widget.doctorOffer.images[0].image
                              : '',
                          fit: BoxFit.cover,
                          placeholder: (context, url) =>
                              const Center(child: CircularProgressIndicator()),
                          errorWidget: (context, url, error) => Icon(
                              Icons.error,
                              size: 50.sp,
                              color: Colors.white),
                        ),

                  /// Dot Indicator
                  if (hasMultipleImages)
                    Positioned(
                      bottom: 16.h,
                      child: SmoothPageIndicator(
                        controller: _pageController,
                        count: widget.doctorOffer.images.length,
                        effect: ExpandingDotsEffect(
                          activeDotColor: Colors.white,
                          dotColor: Colors.grey.shade400,
                          dotHeight: 8.h,
                          dotWidth: 8.h,
                          expansionFactor: 2,
                          spacing: 6.w,
                        ),
                      ),
                    ),
                ],
              ),
            ),
          ),
```


MediCare
ADMIN DASHBOARD

```typescript
import { SAdminService } from './../../Core/services/s-admin.service';
import { Component, OnDestroy, OnInit, AfterViewInit, ViewChild, ElementRef } from '@angular/core';
import { RouterModule } from '@angular/router';
import { TranslateModule, TranslateService } from '@ngx-translate/core';
import { Subject, takeUntil } from 'rxjs';
import { Chart } from 'chart.js/auto';

@Component({
  selector: 'app-admin-statistics',
  standalone: true,
  imports: [RouterModule, TranslateModule],
  templateUrl: './admin-statistics.component.html',
  styleUrl: './admin-statistics.component.css',
})
export class AdminStatisticsComponent implements OnInit, OnDestroy, AfterViewInit {
  @ViewChild('statisticsChart') chartRef!: ElementRef;
  chart!: Chart;
  departmentsCount = 0;
  doctorsCount = 0;
  contactsCount = 0;
  usersCount = 0;
  doctorBlogsCount = 0;
  chainPharmaciesCount = 0;
  InsuranceCompaniesCount = 0;
  chainLaboratoriesCount = 0;
  hospitalsCount = 0;
  pharmaciesCount = 0;
  laboratoriesCount = 0;
  careCentersCount = 0;
  clinicsCount = 0;
  offersCount = 0;
  private destroy$ = new Subject<void>();
  private isViewInitialized = false;
  constructor(private _SAdminService: SAdminService, private translate: TranslateService) {
    this.translate.onLangChange.subscribe(() => {
      this.renderChart();
    });
  }
  ngOnInit() {
    this.getCounts();
  }
  ngAfterViewInit() {
    this.isViewInitialized = true;
  }
  getCounts() {
    this._SAdminService
      .getStatisticsInfo()
      .pipe(takeUntil(this.destroy$))
      .subscribe({
        next: (data) => {
          this.departmentsCount = data.data.departmentsCount;
          this.doctorsCount = data.data.doctorsCount;
          this.usersCount = data.data.usersCount;
          this.doctorBlogsCount = data.data.doctorBlogsCount;
          this.chainPharmaciesCount = data.data.chainPharmaciesCount;
          this.chainLaboratoriesCount = data.data.chainLaboratoriesCount;
          this.hospitalsCount = data.data.hospitalsCount;
          this.pharmaciesCount = data.data.pharmaciesCount;
          this.laboratoriesCount = data.data.laboratoriesCount;
          this.careCentersCount = data.data.careCentersCount;
          this.clinicsCount = data.data.clinicsCount;
          this.contactsCount = data.data.contactsCount;
          this.offersCount = data.data.offersCount;
          this.InsuranceCompaniesCount = data.data.InsuranceCompaniesCount;
          if (this.isViewInitialized) {
            this.renderChart();
          }
        },
      });
  }
```

```
renderChart() {
    if (!this.chartRef) return;
    if (this.chart) {
      this.chart.destroy();
    }
    this.translate.get([
      'STATISTICS.DEPARTMENTS', 'STATISTICS.DOCTORS', 'STATISTICS.USERS',
      'STATISTICS.CLINICS', 'STATISTICS.OFFERS', 'STATISTICS.DOCTOR_BLOGS',
      'STATISTICS.HOSPITALS', 'STATISTICS.CARE_CENTERS', 'STATISTICS.CHAIN_PHARMACIES',
      'STATISTICS.CHAIN_LABORATORIES', 'STATISTICS.PHARMACIES',
'STATISTICS.LABORATORIES', 'STATISTICS.CONTACT',
      'STATISTICS.COUNT', 'STATISTICS.INSURANCE_COMPANIES'
    ]).subscribe(translations => {
      this.chart = new Chart(this.chartRef.nativeElement, {
        type: 'bar',
        data: {
          labels: [
            translations['STATISTICS.DEPARTMENTS'], translations['STATISTICS.DOCTORS'],
            translations['STATISTICS.USERS'], translations['STATISTICS.CLINICS'],
            translations['STATISTICS.OFFERS'], translations['STATISTICS.DOCTOR_BLOGS'],
            translations['STATISTICS.HOSPITALS'], translations[
'STATISTICS.CARE_CENTERS'],
            translations['STATISTICS.CHAIN_PHARMACIES'], translations[
'STATISTICS.CHAIN_LABORATORIES'],
            translations['STATISTICS.PHARMACIES'], translations[
'STATISTICS.LABORATORIES'], translations['STATISTICS.CONTACT'], translations[
'STATISTICS.INSURANCE_COMPANIES']
          ],
          datasets: [{
            label: translations['STATISTICS.COUNT'],
            data: [
              this.departmentsCount, this.doctorsCount, this.usersCount,
              this.clinicsCount, this.offersCount, this.doctorBlogsCount,
              this.hospitalsCount, this.careCentersCount,
              this.chainPharmaciesCount, this.chainLaboratoriesCount,
              this.pharmaciesCount, this.laboratoriesCount, this.contactsCount, this.
InsuranceCompaniesCount
            ],
            backgroundColor: [
              '#354f23', // أحمر فاقع
              '#3cb44b', // أخضر زاهي
              '#ffe119', // أصفر
              '#4363d8', // أزرق قوي
              '#f58231', // برتقالي واضح
              '#911eb4', // بنفسجي غامق
              '#46f0f0', // سماوي فاقع
              '#f032e6', // وردي
              '#bcf60c', // أخضر فسفوري
              '#fabebe', // وردي فاتح
              '#008080', // أخضر مزرق
              '#e6beff', // بنفسجي فاتح
              '#9a6324', // بني
              '#fffac8'  // أصفر باهت
            ],
            borderColor: '#000',
            borderWidth: 1,
          }]
        },
        options: {
          responsive: true,
          maintainAspectRatio: false,
          scales: {
            y: {
              beginAtZero: true,
              ticks: {
                color: '#0dcaf0' // مثلاً أحمر Y لون النص على المحور
              }
            },
            x: {
              ticks: {
                color: '#0dcaf0' // مثلاً أزرق X لون النص على المحور
              }
            }
          }
        }
      });
    });
  }
  ngOnDestroy() {
    this.destroy$.next();
    this.destroy$.complete();
  }
}
```

Mobile Design

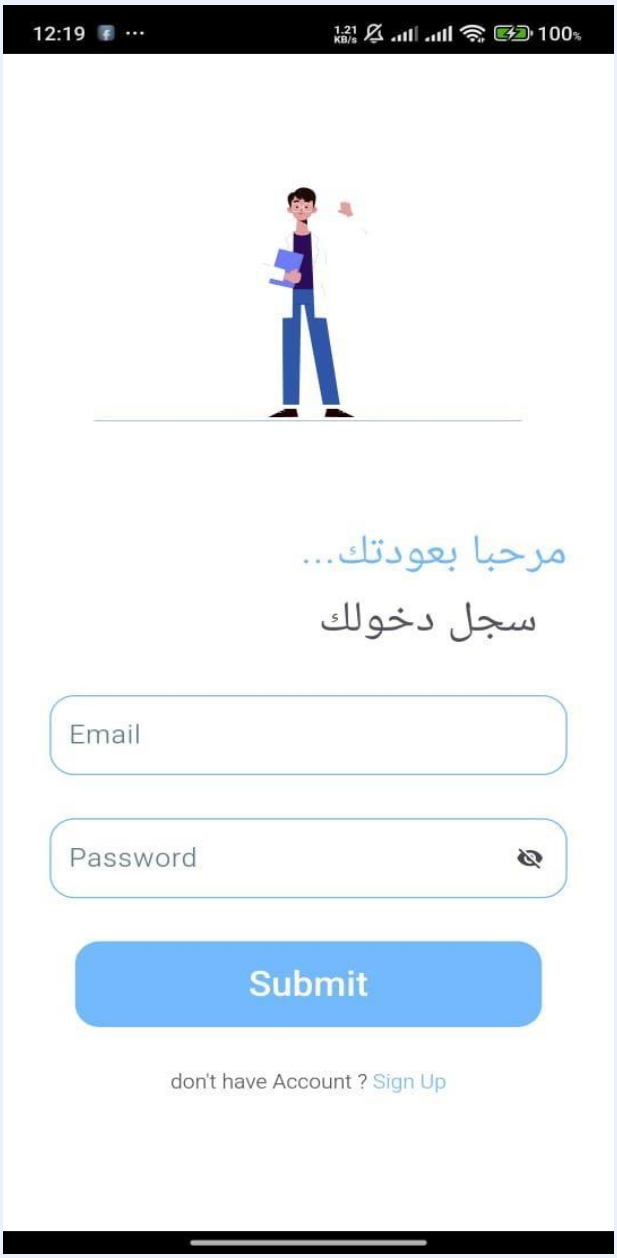
Splash Screen

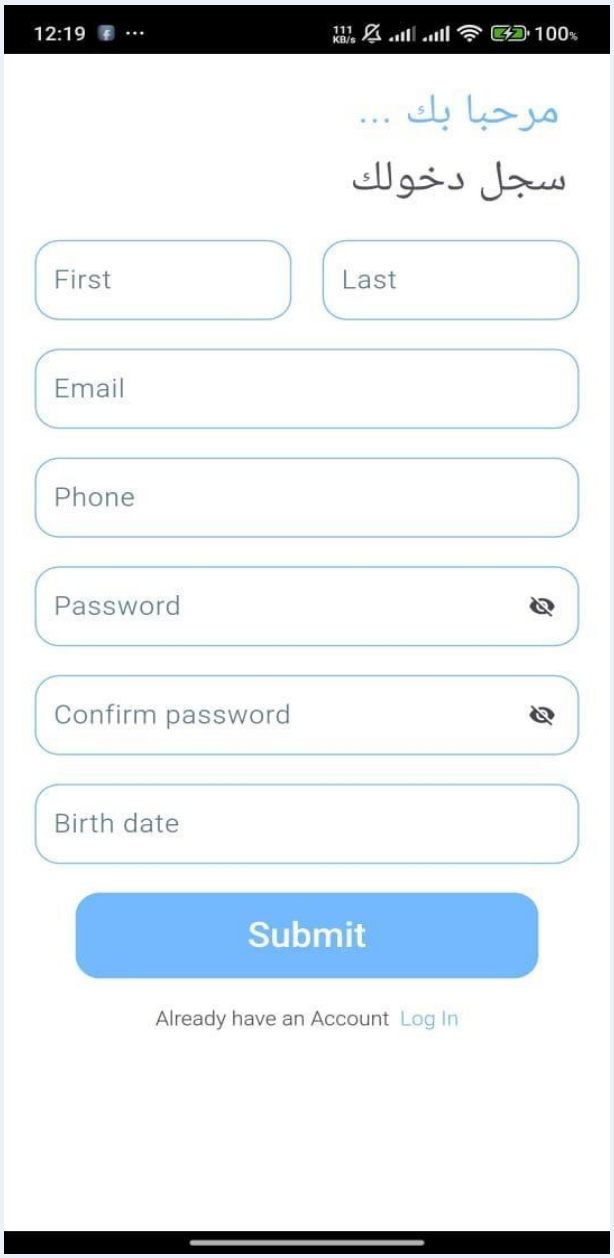
```dart
import 'package:animated_splash_screen/animated_splash_screen.dart';
import 'package:flutter/material.dart';
import 'package:lottie/lottie.dart';
import 'package:media_care/presentation/views/intro/introduction_page_view.dart';

class SplashViewBody extends StatelessWidget {
  const SplashViewBody({
    super.key,
  });

  @override
  Widget build(BuildContext context) {
    return AnimatedSplashScreen(
      splash: Column(
        mainAxisAlignment: MainAxisAlignment.spaceBetween,
        children: [
          Flexible(
            child: Lottie.asset(
              'assets/animation/doctorAnimation1.json',
            ),
          ),
          Flexible(
            child: Image.asset(
              'assets/animation/Medicare.png',
            ),
          )
        ],
      ),
      splashIconSize: 500,
      nextScreen: const IntroView(),
      // duration: 3500,
      // backgroundColor: Colors.white,
    );
  }
}
```

OnBoarding Screen

```dart
import 'package:flutter/material.dart';
import 'package:media_care/presentation/views/intro/widgets/custom_intro_page.dart';

import '../../../../core/utlis/app_images.dart';
import 'custom_bottom_sheet.dart';
import 'custom_lets_go_button.dart';

class IntroViewBody extends StatefulWidget {
  const IntroViewBody({super.key});

  @override
  State<IntroViewBody> createState() => _IntroViewBodyState();
}

final controller = PageController();
bool isLastPage = false;

List<CustomIntroPage> intros = [
  CustomIntroPage(
      image: Assets.imagesDoctors,
      title: 'صحتك تبدأ هنا !',
      subTitle: 'استكشف مجموعة متكاملة من الخدمات الطبية بسهولة وسرعة .'),
  CustomIntroPage(
      image: Assets.imagesApointment,
      title: 'احجز موعدك بضغطة زر !',
      subTitle: 'ابحث عن الاطباء المتخصصين واحجز موعدك في ثوانٍ .'),
  CustomIntroPage(
      image: Assets.imagesArticals,
      title: 'معلومات طبية في أي وقت !',
      subTitle: 'احصل على استشارات طبية موثوقة مباشرة عبر التطبيق .'),
  CustomIntroPage(
      image: Assets.imagesLabs,
      title: 'الرعاية الصحية أصبحت أسهل !',
      subTitle: 'استمتع بتجربة طبية مريحة ومناسبة لجميع احتياجاتك .'),
];

class _IntroViewBodyState extends State<IntroViewBody> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Container(
        padding: const EdgeInsets.only(bottom: 50),
        child: PageView(
          onPageChanged: (index) => setState(() {
            isLastPage = index == intros.length - 1;
          }),
          controller: controller,
          children: intros,
        ),
      ),
      bottomSheet: isLastPage
          ? CustomLetsGoButton()
          : CustomButomSheet(controller: controller, intros: intros),
    );
  }
}
```

Auth Screen


12:19
100%
مرحبا بعودتك...
سجل دخولك
Email
Password
Submit
don't have Account ? Sign Up

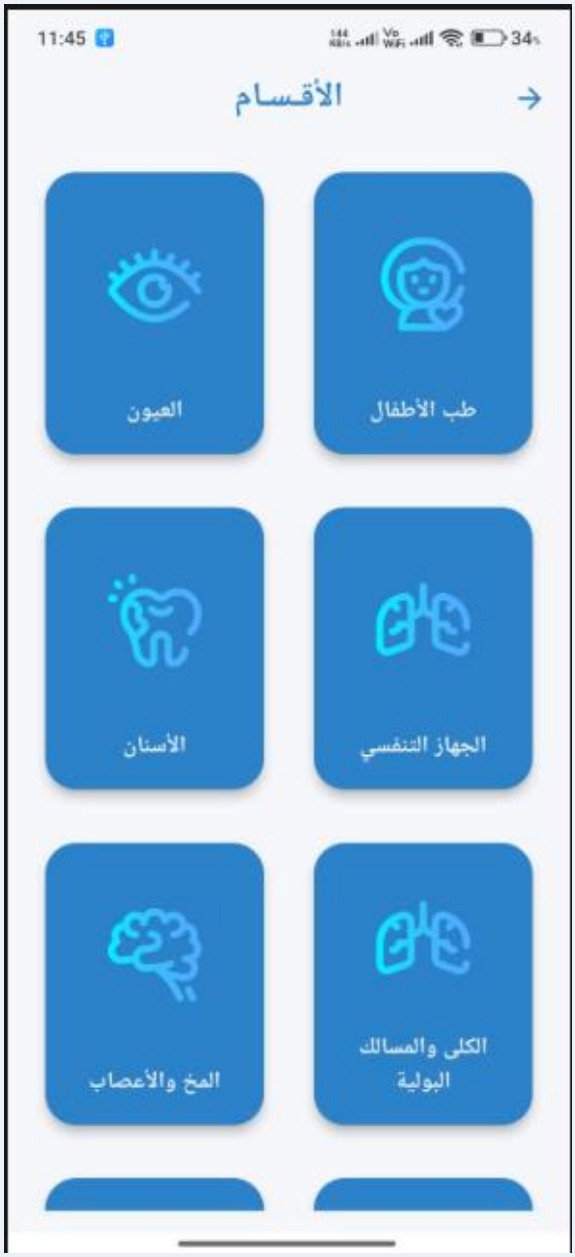

12:19
100%
مرحبا بك ...
سجل دخولك
First
Last
Email
Phone
Password
Confirm password
Birth date
Submit
Already have an Account Log In

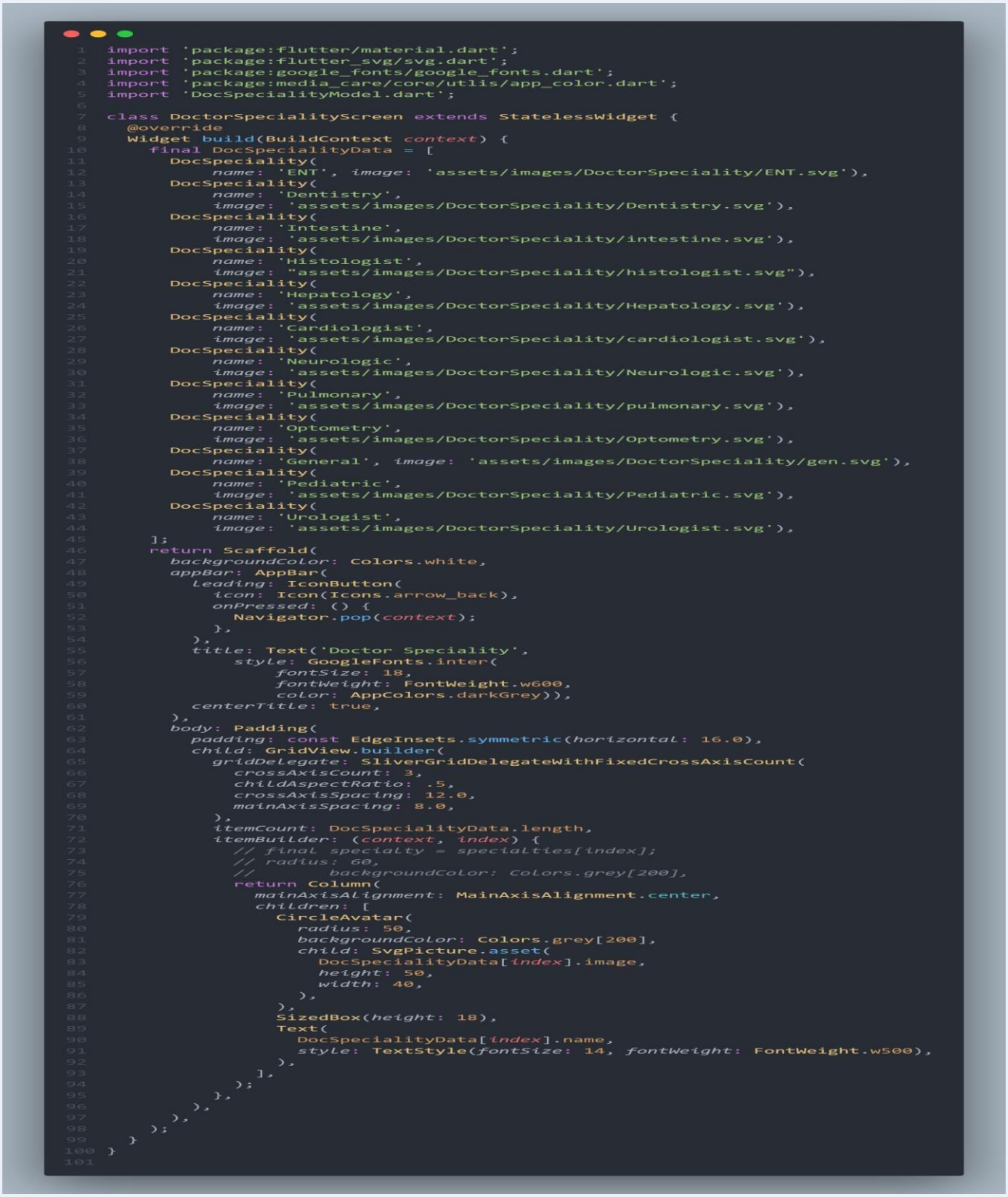
```dart
import 'package:flutter/material.dart';
import 'package:lottie/lottie.dart';
import 'package:media_care/presentation/views/Auth/register/regester_view.dart';

import '../../../../../core/utlis/app_color.dart';
import 'dont_have_email_password.dart';
import 'email_and_passowrd_form.dart';

class LoginViewBody extends StatelessWidget {
  const LoginViewBody({
    super.key,
  });

  @override
  Widget build(BuildContext context) {
    return SafeArea(
      child: Scaffold(
        backgroundColor: Colors.white,
        body: SingleChildScrollView(
          child: Padding(
            padding: const EdgeInsets.symmetric(horizontal: 30, vertical: 24),
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.center,
              children: [
                SizedBox(
                    height: 300,
                    width: 300,
                    child: Lottie.asset(
                      'assets/animation/doctorWelcomed.json',
                    )),
                Align(
                  alignment: AlignmentDirectional.centerEnd,
                  child: Column(
                    children: [
                      Text(
                        'مرحبا بعودتك...',
                        style: TextStyle(
                          fontSize: 30,
                          color: AppColors.primary,
                        ),
                      ),
                      Text(
                        'سجل دخولك ',
                        style: TextStyle(
                          fontSize: 30,
                          color: AppColors.darkGrey,
                        ),
                      ),
                    ],
                  ),
                ),
                SizedBox(
                  height: 30,
                ),
                EmailAndPasswordForm(),
                SizedBox(
                  height: 30,
                ),
                DontHaveAccountText(
                  router: RegisterView(),
                  text: "don't have Account ?",
                  boldText: " Sign Up",
                ),
              ],
            ),
          ),
        ),
      );
    }
}

```

```dart
import 'package:flutter/material.dart';
import 'package:media_care/core/utlis/app_regex.dart';
import 'package:media_care/presentation/views/Auth/login/login_view.dart';
import 'package:media_care/presentation/views/Auth/login/widgets/custom_login_button.dart';

import '../../login/widgets/custom_text_form_field.dart';
import 'first_last_names_form.dart';

class RegisterForm extends StatefulWidget {
  const RegisterForm({
    super.key,
  });

  @override
  State<RegisterForm> createState() => _RegisterFormState();
}

class _RegisterFormState extends State<RegisterForm> {
  GlobalKey<FormState> formKey = GlobalKey();
  bool isSecure = true;

  @override
  Widget build(BuildContext context) {
    return Form(
      key: formKey,
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.center,
        children: [
          FirstAndLastNameForm(),
          SizedBox(
            height: 20,
          ),
          CustomTextField(
            label: 'Email',
            validator: (value) {
              if (!AppRegex.isEmailValid(value!)) {
                return 'Enter a Valid Email';
              }
              return null;
            },
          ),
          SizedBox(
            height: 20,
          ),
          CustomTextField(
            inputType: TextInputType.phone,
            label: 'Phone',
            validator: (value) {
              if (value == null || value.isEmpty) {
                return 'field is required';
              } else if (!AppRegex.isPhoneNumberValid(value)) {
                return 'Enter correct form of password';
              }
              return null;
            },
          ),
          SizedBox(
            height: 20,
          ),
          CustomTextField(
            suffixIcon: GestureDetector(
                onTap: () {
                  setState(() {
                    isSecure = !isSecure;
                  });
                },
                child: isSecure
                    ? Icon(Icons.visibility_off)
                    : Icon(Icons.visibility)),
            label: 'Password',
            isObscureText: isSecure,
            validator: (value) {
              if (value == null || value.isEmpty) {
                return 'field is required';
              } else if (!AppRegex.isPasswordValid(value)) {
                return 'Enter correct form of password';
              }
              return null;
            },
          ),
```

```
    SizedBox(
                height: 20,
              ),
              CustomTextField(
                isObscureText: isSecure,
                suffixIcon: GestureDetector(
                    onTap: () {
                      setState(() {
                        isSecure = !isSecure;
                      });
                    },
                    child: isSecure
                        ? Icon(Icons.visibility_off)
                        : Icon(Icons.visibility)),
                label: 'Confirm password',
                validator: (value) {
                  if (value == null || value.isEmpty) {
                    return 'field is required';
                  } else if (!AppRegex.isPasswordValid(value)) {
                    return 'Enter correct form of password';
                  }
                  return null;
                },
              ),
              SizedBox(
                height: 20,
              ),
              CustomTextField(
                inputType: TextInputType.datetime,
                label: 'Birth date',
                validator: (value) {
                  if (value == null || value.isEmpty) {
                    return 'field is required';
                  } else if (!AppRegex.isDateOfBirthValid(value)) {
                    return 'Enter Invalid Date';
                  }
                  return null;
                },
              ),
              SizedBox(
                height: 20,
              ),
              CustomLoginButton(
                  text: 'Submit',
                  onPresed: () {
                    if (formKey.currentState!.validate()) {
                      Navigator.push(context, MaterialPageRoute(
                        builder: (context) {
                          return LoginView();
                        },
                      ));
                    }
                  }),
              SizedBox(
                height: 20,
              ),
            ],
          ),
        );
    }
}
```

Department Screen





```dart
import 'package:flutter/material.dart';
import 'package:flutter_svg/svg.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:media_care/core/utlis/app_color.dart';
import 'DocSpecialityModel.dart';

class DoctorSpecialityScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final DocSpecialityData = [
      DocSpeciality(
          name: 'ENT', image: 'assets/images/DoctorSpeciality/ENT.svg'),
      DocSpeciality(
          name: 'Dentistry',
          image: 'assets/images/DoctorSpeciality/Dentistry.svg'),
      DocSpeciality(
          name: 'Intestine',
          image: 'assets/images/DoctorSpeciality/intestine.svg'),
      DocSpeciality(
          name: 'Histologist',
          image: "assets/images/DoctorSpeciality/histologist.svg"),
      DocSpeciality(
          name: 'Hepatology',
          image: 'assets/images/DoctorSpeciality/Hepatology.svg'),
      DocSpeciality(
          name: 'Cardiologist',
          image: 'assets/images/DoctorSpeciality/cardiologist.svg'),
      DocSpeciality(
          name: 'Neurologic',
          image: 'assets/images/DoctorSpeciality/Neurologic.svg'),
      DocSpeciality(
          name: 'Pulmonary',
          image: 'assets/images/DoctorSpeciality/pulmonary.svg'),
      DocSpeciality(
          name: 'Optometry',
          image: 'assets/images/DoctorSpeciality/Optometry.svg'),
      DocSpeciality(
          name: 'General', image: 'assets/images/DoctorSpeciality/gen.svg'),
      DocSpeciality(
          name: 'Pediatric',
          image: 'assets/images/DoctorSpeciality/Pediatric.svg'),
      DocSpeciality(
          name: 'Urologist',
          image: 'assets/images/DoctorSpeciality/Urologist.svg'),
    ];
    return Scaffold(
      backgroundColor: Colors.white,
      appBar: AppBar(
        leading: IconButton(
          icon: Icon(Icons.arrow_back),
          onPressed: () {
            Navigator.pop(context);
          },
        ),
        title: Text('Doctor Speciality',
            style: GoogleFonts.inter(
                fontSize: 18,
                fontWeight: FontWeight.w600,
                color: AppColors.darkGrey)),
        centerTitle: true,
      ),
      body: Padding(
        padding: const EdgeInsets.symmetric(horizontal: 16.0),
        child: GridView.builder(
          gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
            crossAxisCount: 3,
            childAspectRatio: .5,
            crossAxisSpacing: 12.0,
            mainAxisSpacing: 8.0,
          ),
          itemCount: DocSpecialityData.length,
          itemBuilder: (context, index) {
            // final specialty = specialties[index];
            // radius: 60,
            //         backgroundColor: Colors.grey[200],
            return Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                CircleAvatar(
                  radius: 50,
                  backgroundColor: Colors.grey[200],
                  child: SvgPicture.asset(
                    DocSpecialityData[index].image,
                    height: 50,
                    width: 40,
                  ),
                ),
                SizedBox(height: 18),
                Text(
                  DocSpecialityData[index].name,
                  style: TextStyle(fontSize: 14, fontWeight: FontWeight.w500),
                ),
              ],
            );
          },
        ),
      ),
    );
  }
}
```

Summary

MediCare is a comprehensive medical application and website designed to simplify healthcare access. It enables users to book doctor appointments, view doctor specializations, and read ratings for informed decision-making. The platform ensures the credibility and accuracy of doctor information, with each professional having a rating system to evaluate their performance. MediCare aims to provide a seamless and reliable healthcare experience for users.