# Software Requirements Specification

**for**

# OMAS

**Version 1.0**

**Prepared by**


**Group Name: 47**

| | | |
|---|---|---|
| **Daniel Garcia Soni** | **A01796163** | **A01796163@tec.mx** |
| **Carlos Ochoa Gonzalez** | **A01746583** | |
| **Juan José Ponce Estrada** | **A01659811** | **A01659811@tec.mx** |
| **Tonatiuh Salas Ortiz** | **A01080251** | **A01080251@tec.mx** |


**Professor:** ***Perla Angelica García Aguirre***

**Course:** **Análisis, diseño y construcción de software**

**Date:** **28/09/2025**

# Revisions

| Version | Primary Author(s) | Description of Version | Date Completed |
|---|---|---|---|
| 0.1 | Tonatiuh Salas Ortiz | Initial draft: created structure, intro, scope | 15/09/25 |
| 0.2 | Carlos Ochoa González | Added functional requirements F1–F10 | 17/09/25 |
| 0.3 | Daniel Garcia Soni | Expanded modules,requirements F11-F19 use cases U5–U8, integrated constraints, and added security/NFRs | 19/09/25 |
| 1.0 | Daniel Garcia Soni, Tonatiuh Salas Orti, Carlos Ochoa Gonzalez , Juan José Ponce Estrada | Final release: consolidated all sections, polished use cases, completed non-functional reqs, ensured consistency across document | 21/09/25 |

# 1 Introduction

This section introduces the Online Medical Appointment System (OMAS), outlines the scope, goals, and audience for this SRS, and explains how the document is structured. It summarizes the two-sided nature of the platform (patients and providers) and highlights mobile and web delivery.

## 1.1 Document Purpose

This document specifies the requirements for an **Online Medical Appointment System (OMAS)**. The system will enable patients to book, manage, and track medical appointments online while allowing doctors and healthcare providers (e.g., clinics, hospitals, ER rooms) to manage their schedules efficiently.

The purpose of this SRS is to define the functional and non-functional requirements of the system to ensure clarity among developers, clients, healthcare providers, and end-users. It will serve as a baseline for development, testing, and future enhancements.

This system is explicitly designed as a **two-sided platform** that serves both **patients** and **healthcare providers** (clinics, hospitals, independent practitioners). Requirements are organized to reflect patient-facing and provider-facing capabilities, ensuring alignment across both sides.

## 1.2 Product Scope

The system is a ***web and mobile-based platform*** that provides:

- A scheduling interface for patients to book, reschedule, or cancel appointments
- A management dashboard for doctors and clinics to update availability and view appointments
- Notifications and reminders to reduce missed appointments
- Secure storage of patient information and compliance with privacy regulations

**Benefits:**
*For patients:*

- Improved accessibility for patients
- Enhanced patient satisfaction and reduced no-shows

*For healthcare providers:*

- *Reduced administrative burden on clinics*
- *Increased efficiency in managing medical appointments*

## 1.3 Intended Audience and Document Overview

- **Clients/Healthcare Administrators:** to ensure system requirements align with organizational needs
- **Developers & Testers:** to guide the design, coding, and verification of the system
- **End Users (Patients & Doctors):** to validate usability and relevance of features

This document includes:

- Introduction and scope of the system
- Overall description and assumptions
- Specific functional and non-functional requirements
- Use case models and diagrams
- Appendices with data dictionary and group logs

## 1.4 Definitions, Acronyms and Abbreviations

**2FA:** Two-Factor Authentication.
**API:** Application Programming Interface.
**EHR/EMR:** Electronic Health/Medical Record.
**HIPAA-equivalent:** Local regulations providing protections comparable to HIPAA for health data.
**NFR:** Non-Functional Requirement.
**OMAS/OMAP:** Online Medical Appointment System/Platform.
**P0/P1/P2:** Priority levels (highest to lower).
**SMS:** Short Message Service (text messages).
**Two-sided platform:** A product that simultaneously serves two distinct user groups (patients and providers) and creates value by enabling interactions between them.
**UI:** User Interface.
**UML:** Unified Modeling Language.

## 1.5 Document Conventions

- **Formatting:** Arial 11 pt, single-spaced; section headings follow the IEEE template.
- **Labeling:** Functional requirements **F#**; Non-functional requirements **NF#**; Use cases **U#**; Data elements **DD-#**
- **Priorities:** P0 (must), P1 (should), P2 (nice-to-have).
- **Roles:** “Patient”, “Provider” (doctor/clinic), “Admin”.

## 1.6 References and Acknowledgments

---

### *1.6 References and Acknowledgments*

#### *References*


- *IEEE Computer Society. IEEE Std 830-1998: Recommended Practice for Software Requirements Specifications. Institute of Electrical and Electronics Engineers, 1998.*

- *Gomaa, Hassan. Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures. Addison-Wesley, 2004.*

- *Object Management Group (OMG). Unified Modeling Language (UML), Version 2.5.1. 2017. Available at: https://www.omg.org/spec/UML/2.5.1/*

- *World Wide Web Consortium (W3C). Web Content Accessibility Guidelines (WCAG) 2.1. 2018. Available at : https://www.w3.org/TR/WCAG21/*

- *U.S. Department of Health & Human Services. Health Insurance Portability and Accountability Act of 1996 (HIPAA): Privacy Rule. Available at: https://www.hhs.gov/hipaa/*


**Acknowledgments**

The development of this Software Requirements Specification was guided by the course Análisis, diseño y construcción de software at Tecnológico de Monterrey under the supervision of Professor Perla Angélica García Aguirre.

Special thanks to the project team members for their collaborative contributions. Each member was responsible for drafting, expanding, and refining specific sections, ensuring that both patient and provider perspectives were integrated into the OMAS platform requirements.

# 2 Overall Description

## 2.1 Product Overview

The Online Medical Appointment System (OMAS) is a new, self-contained product designed for clinics and hospitals. It connects patients and healthcare providers through a digital platform. Patients access the system via a web or mobile app, while doctors/clinics use an admin portal.

High-Level Workflow:

1. Patients log in and request appointments
2. The system checks availability and confirms bookings
3. Doctors view/manage schedules and patient records
4. Notifications/reminders are sent automatically

The platform operates in a two-sided configuration: a Patient Web/Mobile app for discovery and booking, and a Provider Web/Mobile portal for availability and operations. All clients communicate with a secure backend over HTTPS and a relational database. Mobile is in scope; responsive web and optional native/hybrid mobile apps must remain consistent in features and behavior.

## 2.2 Product Functionality

- Patient registration and profile management
- Doctor/clinic profile management
- Appointment booking, rescheduling, and cancellation
- Notifications via email/SMS
- Secure patient record handling

## 2.3 Design and Implementation Constraints

- **Modeling:** UML is required for architecture and use-case views.
- **Design method:** Apply a component-oriented approach consistent with COMET principles (components, connectors, and explicit interfaces).
- **Tech constraints:** Relational DB (PostgreSQL/MySQL); RESTful APIs over HTTPS; JWT-based sessions; message queue for notifications (e.g., retry/backoff).
- **Security:** Data encryption in transit (TLS 1.2+) and at rest (DB encryption).
- **Mobile consistency:** Responsive web **and** mobile apps must keep feature parity for all P0 requirements.

## 2.4 Assumptions and Dependencies

1. Users will have access to the internet
2. Clinics will provide accurate schedules and availability
3. Third-party services (SMS/email API, payment gateways) will remain available
4. System will integrate with existing EMR/EHR systems only if APIs are provided
5. Mobile push/SMS/email vendors provide stable SLAs and delivery receipts.
6. App stores approval (if native) does not constrain release timelines.
7. Timezone, locales, and daylight-saving rules are provided by the platform/OS.

# 3 Specific Requirements

## 3.1 External Interface Requirements

### 3.1.1 User Interfaces

- **Patient UI (Web/Mobile):** sign-up/login, search by specialty/date, slot selector, book/reschedule/cancel, notifications center.
- **Provider UI (Web/Mobile):** availability editor (weekly grid + exceptions), day/week agenda, appointment detail, mark outcome, reports.
- **Accessibility:** WCAG 2.1 AA.
- **Web/Mobile Consistency.**

### 3.1.2 Hardware Interfaces

System must work on:

- Smartphones (Android, iOS)
- Tablets
- Desktop browsers (Windows, macOS, Linux)
- Servers hosted on a cloud platform (e.g., AWS, Azure, or GCP)

### 3.1.3 Software Interfaces

**Database:** PostgreSQL/MySQL for storing patient, doctor, and appointment data
**APIs:** RESTful APIs for communication between front-end and back-end
**Push notifications:** APNs/FCM (**if enabled**).
**Calendar export:** iCal feed (read only) by provider.

**Third-party Services:**

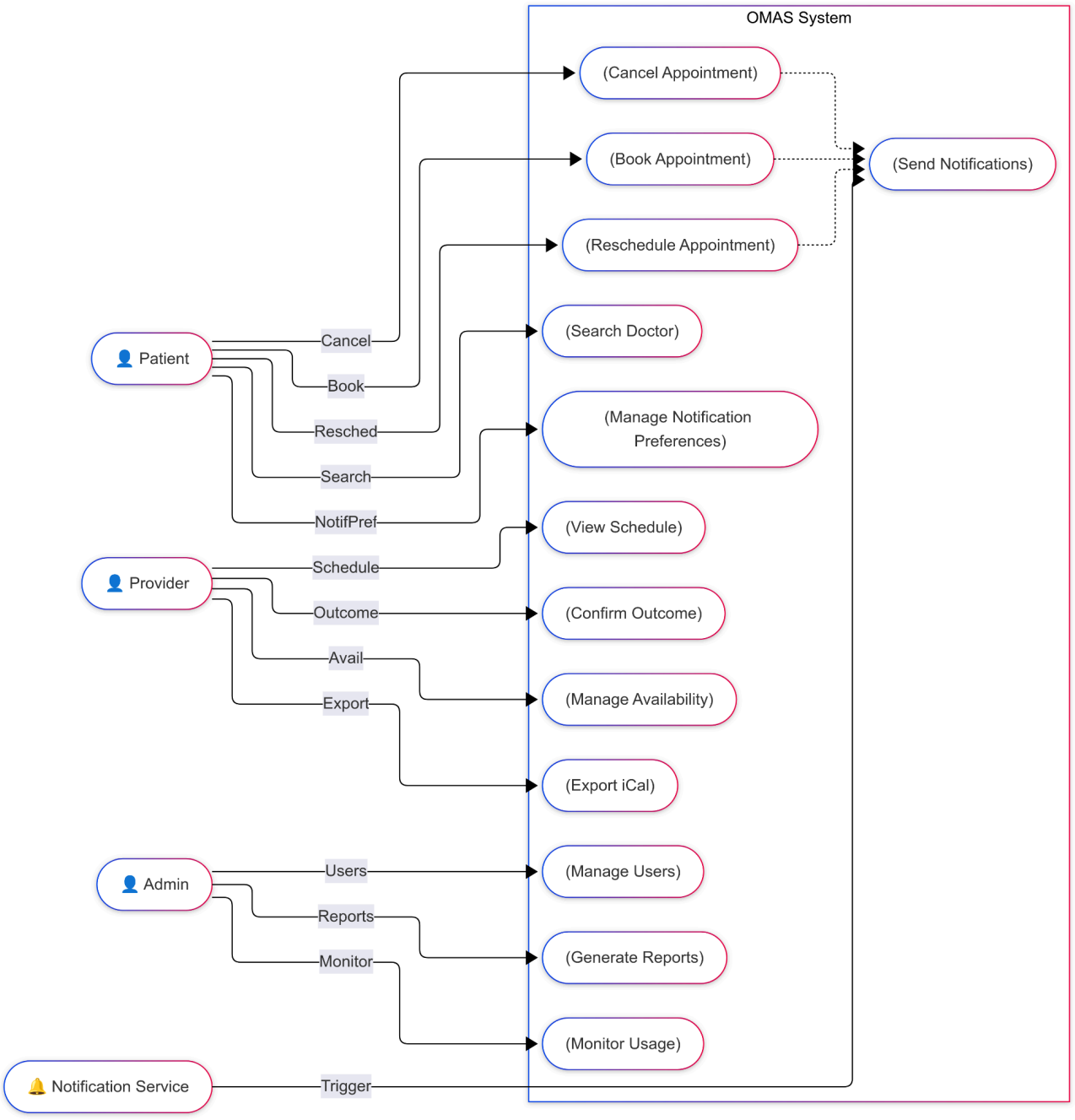
- SMS/email gateway for notifications
- Optional: Payment gateway (Stripe, PayPal)

## 3.2 Functional Requirements

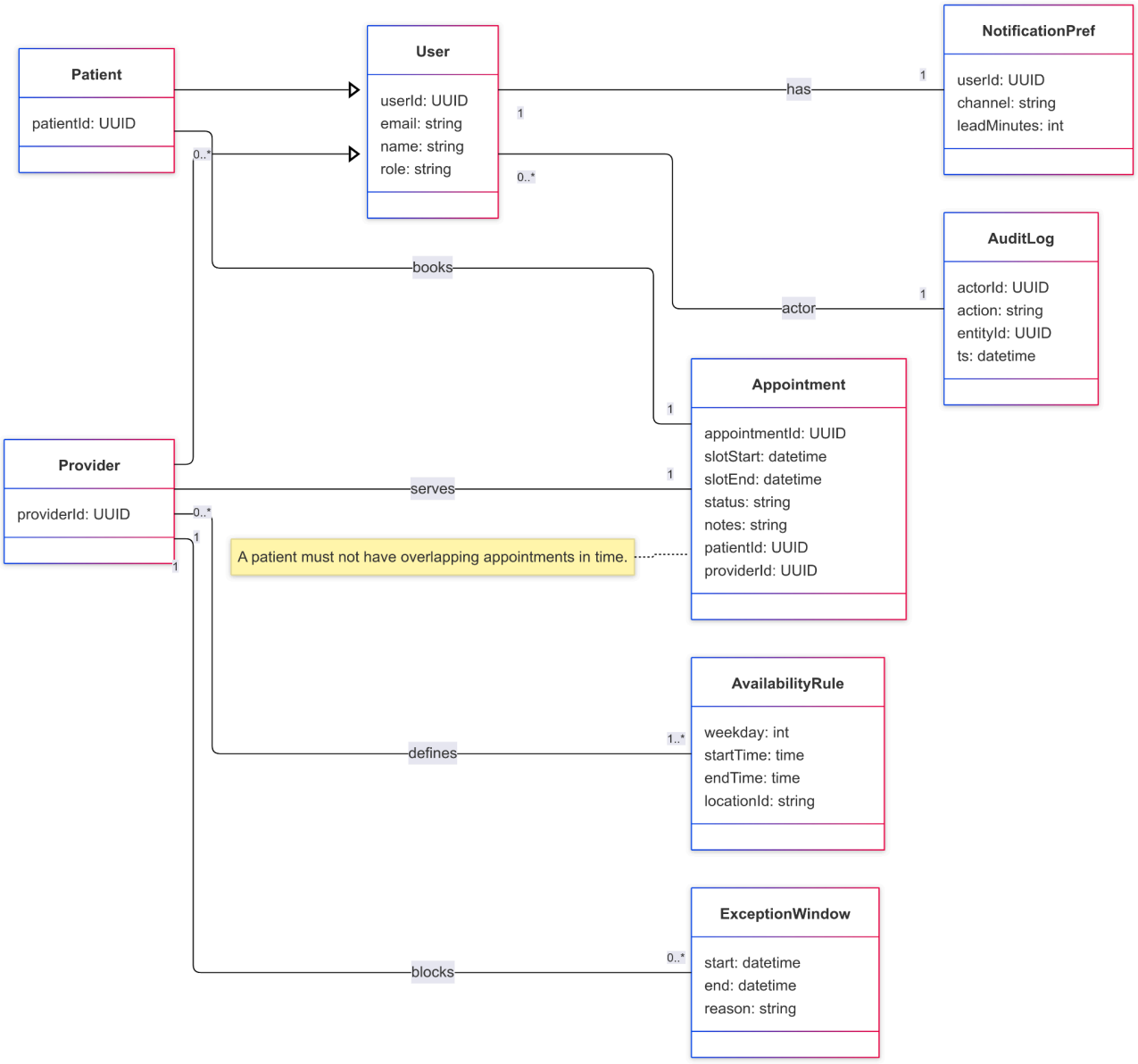
### 3.2.1 F1: The system shall …

F1: The system shall allow patients to register, log in, and manage their profiles.
F2: The system shall allow doctors/clinics to register, log in, and manage their profiles.
F3: The system shall allow patients to search for doctors by specialty, availability, or location.
F4: The system shall allow patients to book appointments online.
F5: The system shall allow patients to reschedule or cancel appointments.
F6: The system shall notify patients and doctors about booked, canceled, or rescheduled appointments via email/SMS.
F7: The system shall allow doctors to update their availability and block time slots.
F8: The system shall maintain an appointment history for patients and doctors.
F9: The system shall allow administrators to monitor system usage and manage users.
F10: The system shall provide basic reporting (e.g., number of appointments per day, doctor utilization).
F11: Provider availability (weekly rules and exceptions).
F12: Provider schedule view (day/week).
F13: Appointment outcomes (completed/no-show + note).
F14: Notification preferences (channel/lead time).
F15: Notification queue and retries with backoff.
F16: Audit logging (auth, PII, schedule changes).
F17: Feature parity P0 on Mobile (consistent functionality).
F18: iCal export (read-only) per provider.
F19: Double-booking and configurable “near-overlap” prevention.

## 3.3 Use Case Model


OMAS System
(Cancel Appointment)
(Book Appointment)
(Reschedule Appointment)
(Send Notifications)
(Search Doctor)
(Manage Notification Preferences)
(View Schedule)
(Confirm Outcome)
(Manage Availability)
(Export iCal)
(Manage Users)
(Generate Reports)
(Monitor Usage)
Patient
Cancel
Book
Resched
Search
NotifPref
Provider
Schedule
Outcome
Avail
Export
Admin
Users
Reports
Monitor
Notification Service
Trigger

## 3.4 Domain Model (Class Diagram)



### 3.4.1 Use Cases

**Use Case U1: Book Appointment**

**Author:** Tonatiuh Salas Ortiz

**Purpose:** To enable a patient to schedule a medical appointment with a doctor.

**Requirements Traceability:** Linked to F3, F4, F6

**Priority:** High

**Preconditions:**

- Patient must be registered and logged in
- Doctor must have availability in the system

**Postconditions:**

- Appointment is created in the system
- Notifications are sent to both patient and doctor

**Actors:**

- Patient (primary)
- Doctor (secondary, receives notification)
- System (validates and stores data)

**Basic Flow:**

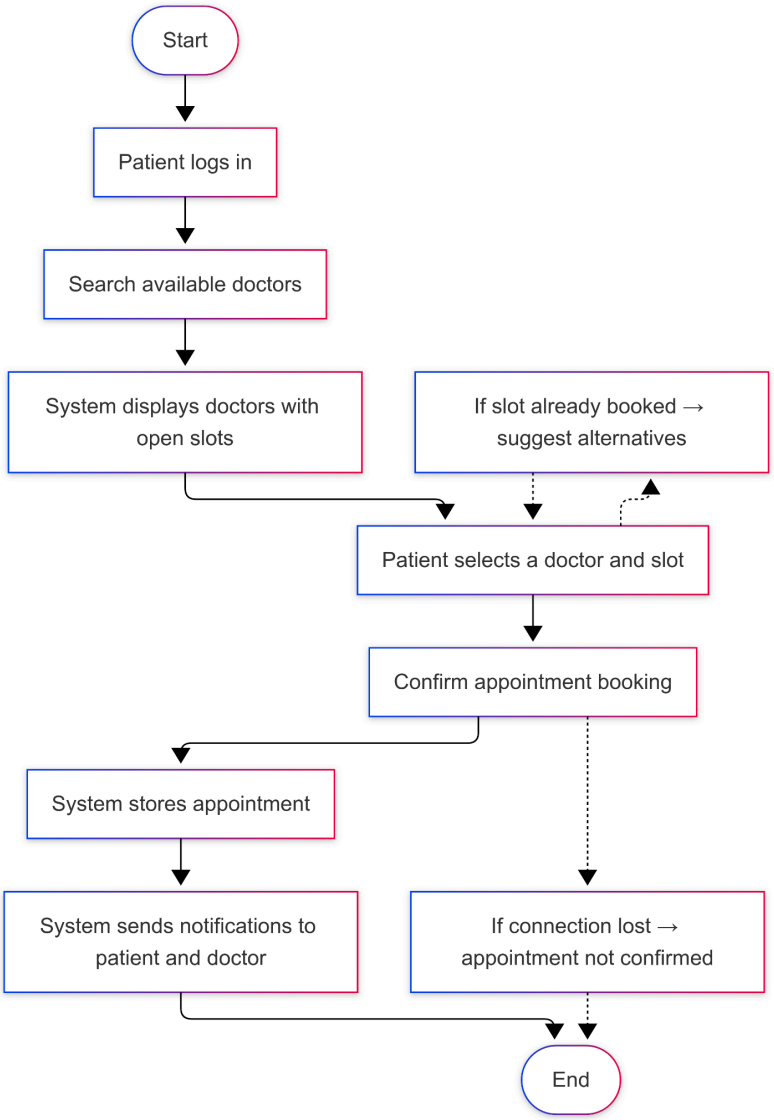
1. Patient logs in
2. Patient searches for available doctors
3. System displays doctors with open time slots
4. Patient selects a time slot and confirms booking
5. System stores appointment and sends notifications

**Alternative Flow:**

- If the chosen slot is already booked, the system suggests other available slots.

**Exceptions:**

- If internet connection is lost during booking, the appointment is not confirmed.




**Use Case U2: Reschedule Appointment**

**Author:** Tonatiuh Salas Ortiz

**Purpose:** Allow patients to change the date/time of an existing appointment.

**Requirements Traceability:** Linked to F5, F6.

**Priority:** High.

**Preconditions:**

- Patient must be logged in
- Patient must have an existing appointment

- Doctor must have other available slots

- **Postconditions:**
  - The old appointment is canceled
  - A new appointment is created in the system
  - Notifications are sent to both patient and doctor

- **Actors:**

  - Patient (primary)
  - Doctor (receives updated schedule notification)
  - System (handles update and notifications)

- **Basic Flow:**

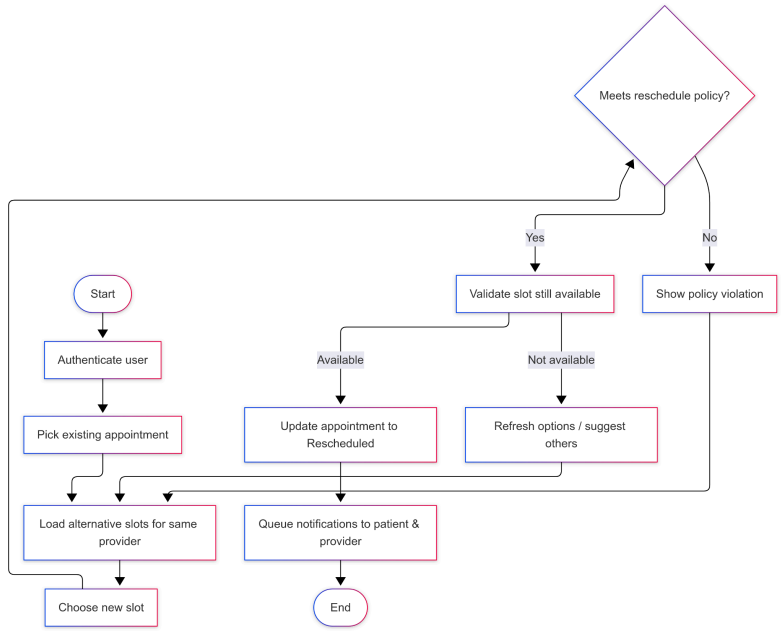
  - Patient logs in and views appointment history
  - Patient selects an appointment to reschedule
  - System shows available time slots for the doctor
  - Patient selects a new slot and confirms
  - System updates appointment and sends notifications

- **Alternative Flow:**

  - If the new slot becomes unavailable during the process, the system suggests alternatives.

- **Exceptions:**

  - If the appointment is within a restricted cancellation/reschedule window (e.g., <24 hrs), the system denies the request.




## Use Case U3: Cancel Appointment

**Author:** Tonatiuh Salas Ortiz

**Purpose:** Allow patients to cancel an appointment.

**Requirements Traceability:** Linked to F5, F6.

**Priority:** Medium.

**Preconditions:**

- Patient must be logged in
- Appointment must exist and be cancellable under clinic policy

**Postconditions:**

- Appointment is removed from the schedule
- Notifications sent to patient and doctor

**Actors:**

- Patient (primary)
- Doctor (receives cancellation notification)
- System (updates records and sends alerts)

**Basic Flow:**

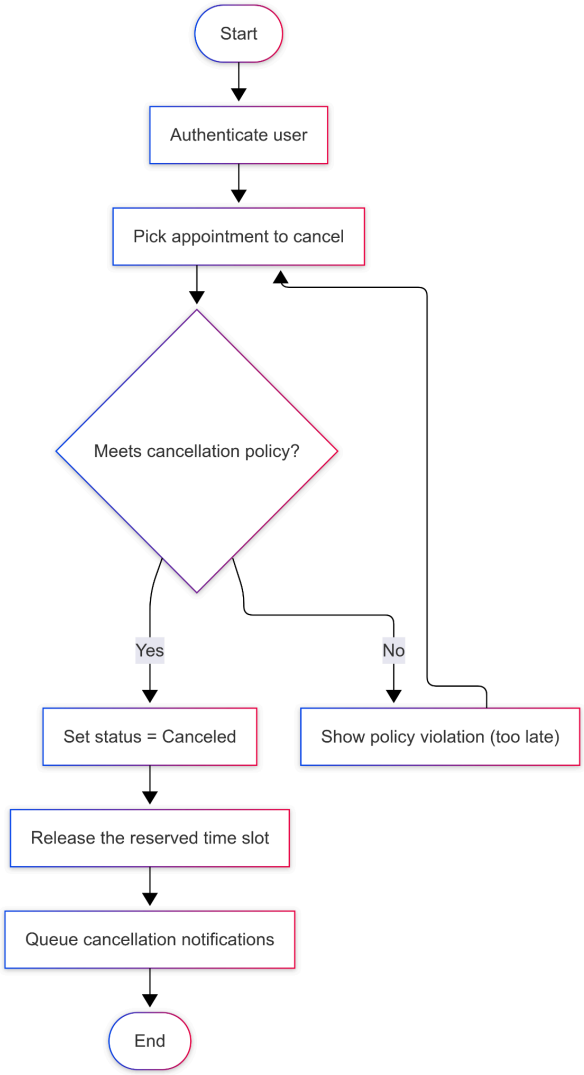
1. Patient logs in.
2. Patient views appointment list
3. Patient selects appointment to cancel
4. System requests confirmation
5. Patient confirms cancellation
6. System updates record and sends notifications

**Alternative Flow:**

- If clinic policy prohibits late cancellations, the system denies the request and displays a message.

**Exceptions:**

- System fails to send notification → retries until successful.



**Use Case U4: Receive Notifications**

**Purpose:** Ensure patients and doctors receive reminders and updates about appointments

**Author:** Tonatiuh Salas Ortiz

**Requirements Traceability:** Linked to F6

**Priority:** High

**Preconditions:**

- Valid appointment exists in the system
- Patient and doctor have valid contact information

**Postconditions:**

- Notifications (SMS/email) are sent for confirmations, reminders, reschedules, and cancellations

**Actors:**

- System (primary, triggers messages)
- Patient (receives messages)
- Doctor (receives messages)
- Notification service provider (third-party API)

**Basic Flow:**

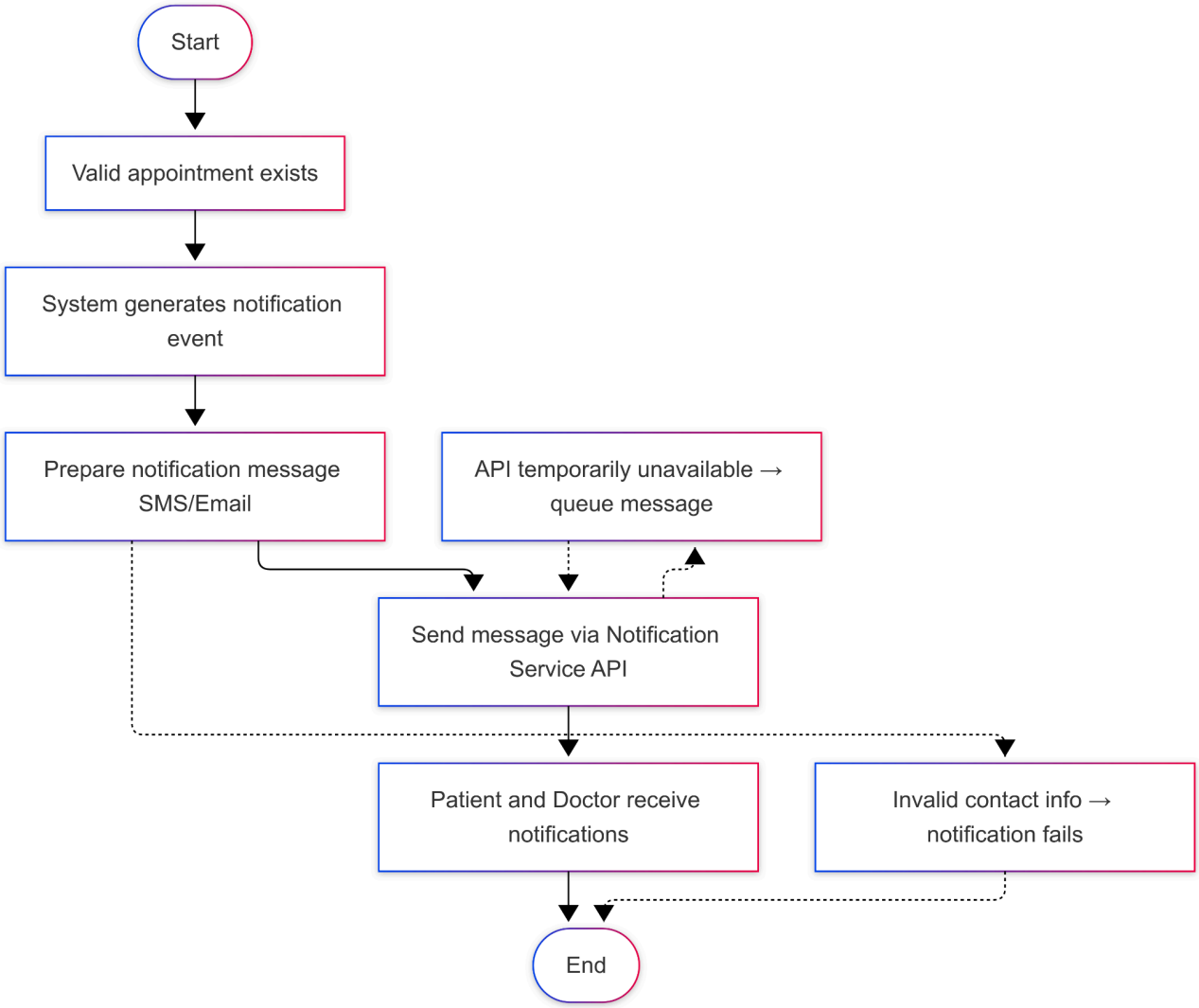
1. Appointment is created, rescheduled, or canceled.
2. System generates a notification message.
3. Message is sent via SMS/email API.
4. Patient and doctor receive confirmation.

**Alternative Flow:**

- If API is temporarily unavailable, system queues notifications and retries.

**Exceptions:**

- Invalid email/phone number → message not delivered → logged as failed notification.




**Use Case U5: Manage doctor availability**

**Purpose:** Allow providers to define weekly availability and exception windows (e.g., vacations).
**Author:** Daniel Garcia Soni
**Requirements Traceability:** Linked to F11, F12

**Priority:** High

**Preconditions:**
Provider must be logged in
Provider profile must exist

**Postconditions:**
Availability rules and exceptions are saved; conflicts revalidated

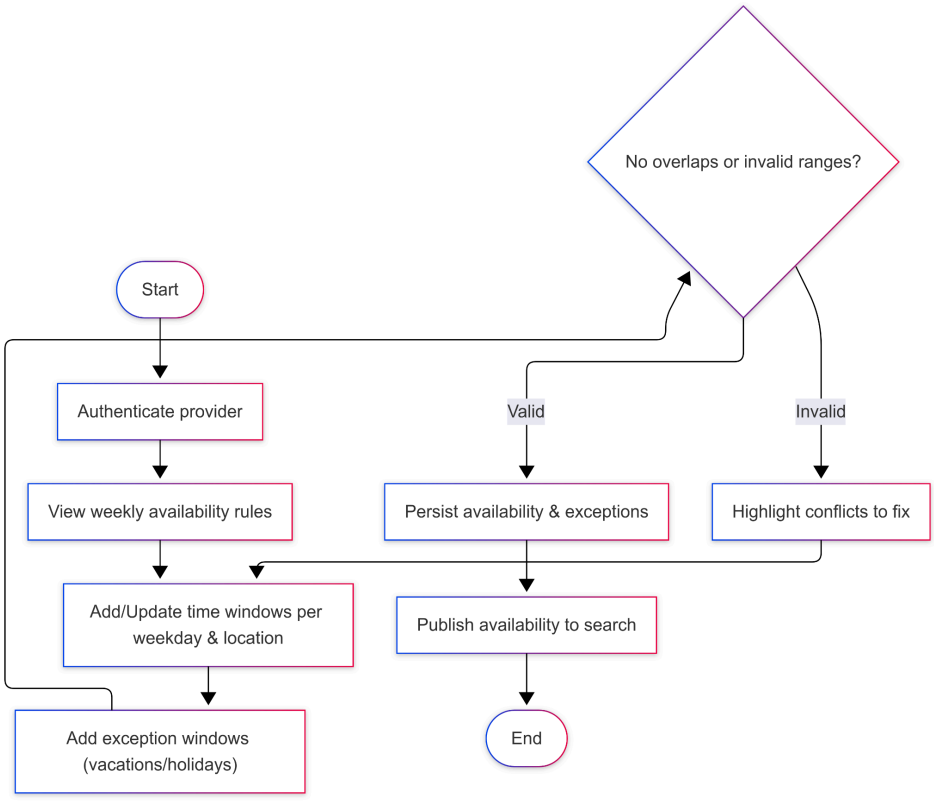
**Actors:**
Provider (primary)
System (validates conflicts and saves data)

**Basic Flow:**

- Provider opens availability module
- Provider edits weekly schedule grid
- Provider adds exception dates (vacation, holiday, etc.)
- System validates and confirms changes
- Provider saves configuration

**Alternative Flow:**
If conflicts are detected (appointments already booked), the system alerts the provider and suggests resolutions.

**Exceptions:**
Database save error → No changes applied → Error displayed and logged




## Use Case U6: View appointment schedule

**Purpose:** Let providers see daily and weekly schedules with appointment details.
**Author:** Mariana López Martínez
**Requirements Traceability:** Linked to F12

**Priority:** High

**Preconditions:**
Provider must be authenticated
Appointments must exist in the system

**Postconditions:**
Schedule displayed with correct details

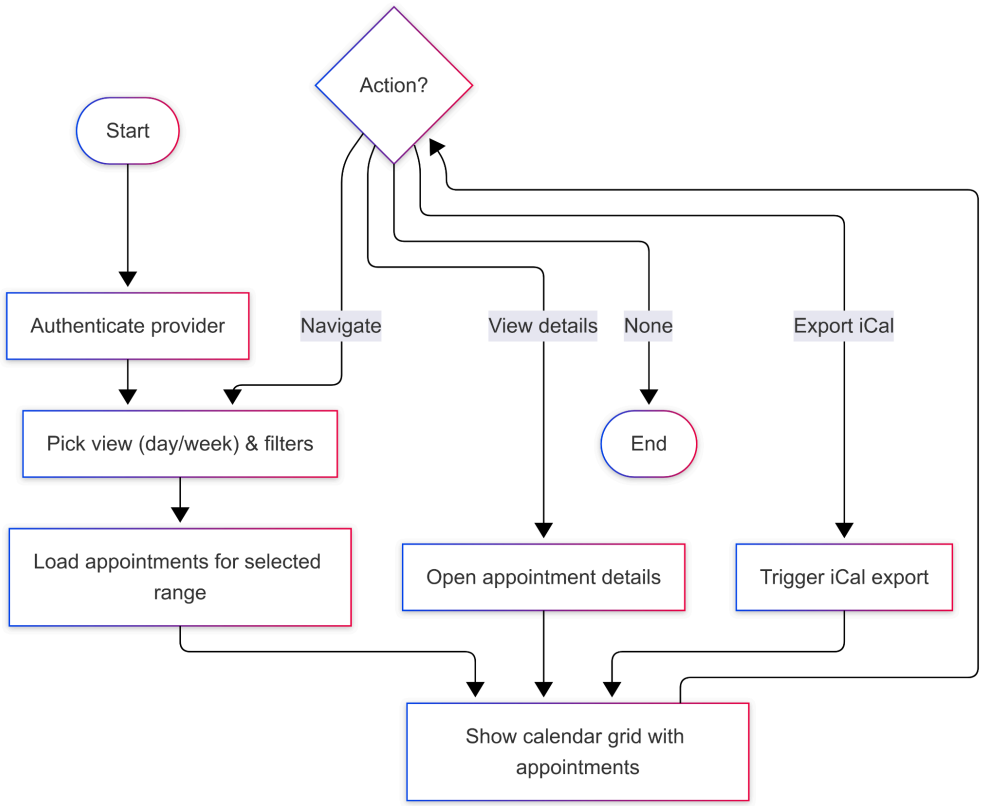
**Actors:**
Provider (primary)
System (renders schedule view)

**Basic Flow:**

- Provider logs in
- Provider opens schedule page
- System displays day/week view of appointments
- Provider clicks an appointment to see details

**Alternative Flow:**
Provider filters appointments by status (booked, completed, canceled).

**Exceptions:**
Data fetch error → System retries and displays cached version if available



## Use Case U7: Confirm appointment completion

**Purpose:** Allow providers to mark appointments as completed or no-show and add notes.
**Author:** Daniel Garcia Soni
**Requirements Traceability:** Linked to F13, F16

**Priority:** Medium

**Preconditions:**
Provider must be logged in
Appointment must exist and be eligible for outcome update

**Postconditions:**
Outcome saved; audit log updated; reports reflect status

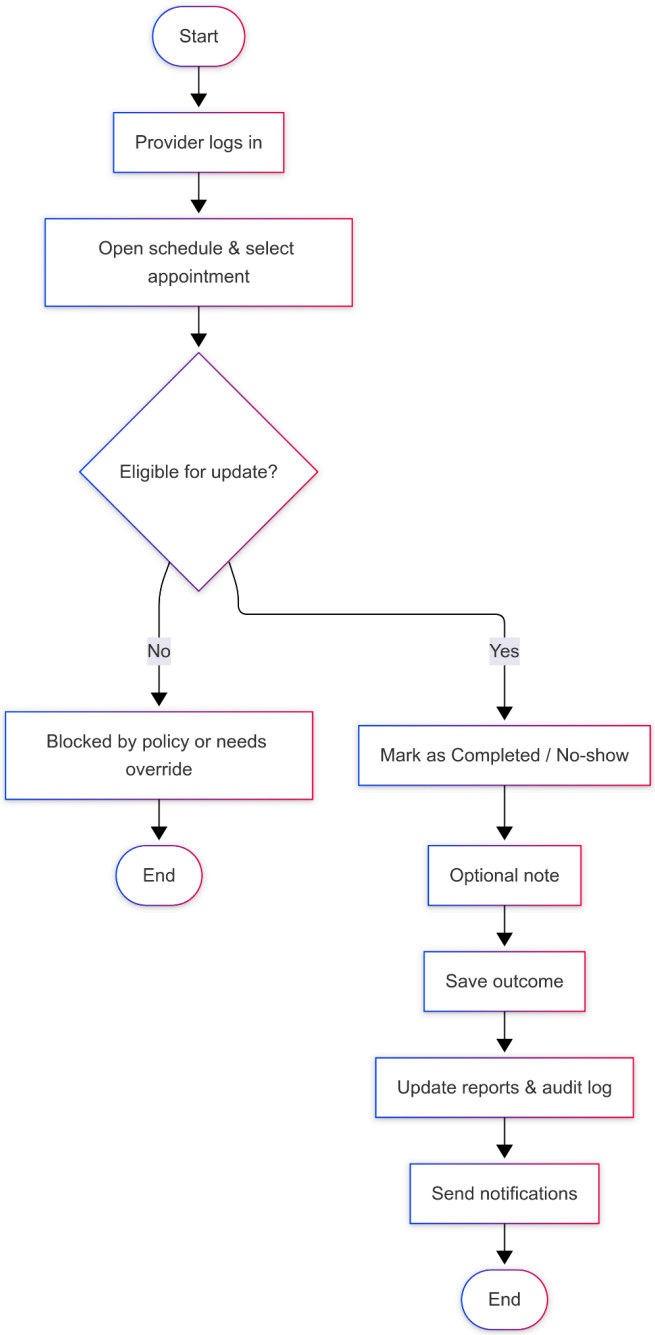
**Actors:**
Provider (primary)
System (saves outcome, updates reports, logs event)

**Basic Flow:**

- Provider selects appointment from schedule
- Provider marks as "Completed" or "No-show"
- Provider optionally adds a short note
- System saves outcome, updates reports, logs action

**Alternative Flow:**
If policy restricts editing outcomes after a certain time, the system blocks the action or requests admin override.

**Exceptions:**
Save failure → Outcome not changed → Error displayed and logged




### *Use Case U8: Configure notifications*

**Purpose:** Let users (patients or providers) configure preferred channels and lead times for notifications.
**Author:** Daniel Garcia Soni
**Requirements Traceability:** Linked to F14, F15

**Priority:** Medium

**Preconditions:**
User must be logged in
Valid contact information must exist

**Postconditions:**
Preferences saved and applied to future notifications

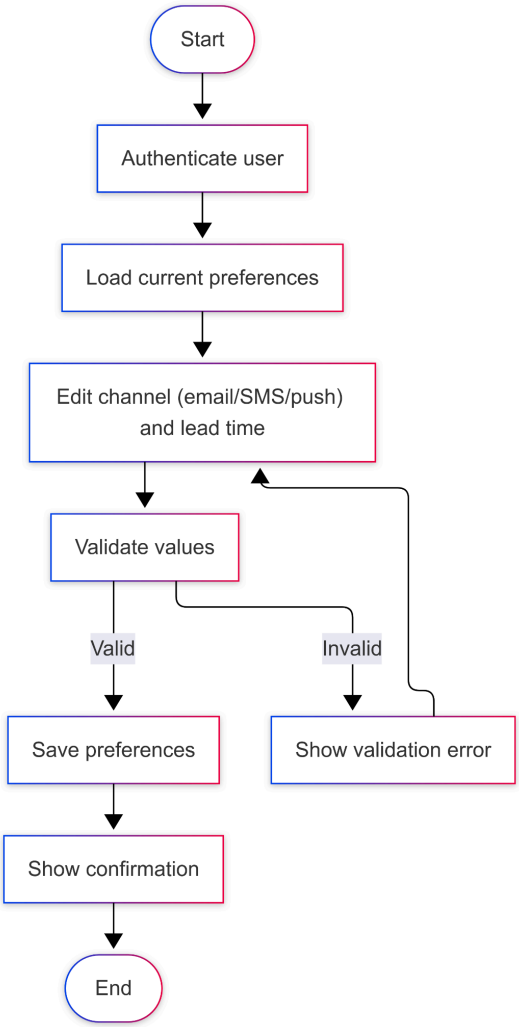
**Actors:**
Patient (primary)
Provider (primary)
System (saves preferences)
Notification service provider (third-party API)

**Basic Flow:**

- User navigates to notification settings
- User selects preferred channels (SMS, email, push) and lead times (24h, 2h, etc.)
- System validates inputs and saves preferences

**Alternative Flow:**
If a selected channel is unavailable, the system suggests alternatives.

**Exceptions:**
Invalid contact info → Preferences not saved → Error displayed
Notification service failure → System retries with exponential backoff




# 4 Other Non-functional Requirements

## 4.1 Performance Requirements

- p95 ≤ 3 s in API with 500 concurrent users.
- Slot search first page ≤ 2 s (p95).
- Notification queuing ≤ 500 ms; async delivery.
- Peak throughput: ≥ 50 appointments/min sustained for 10 min.

## 4.2 Safety and Security Requirements

- **TLS 1.2+, HSTS**; encryption at rest for PII.
- **Mandatory 2FA for Provider/Admin;** optional for Patient.
- **RBAC least privilege;** rate-limiting and anti-bot on login/booking.
- Session management with rotating refresh tokens and expiration.
- **Audit (F16);** retention/deletion in accordance with local regulations.

## 4.3 Software Quality Attributes

- **Reliability:**99.5% uptime; zero loss in confirmed bookings (WAL + backups).
- **Usability:** booking in ≤ 3 steps; WCAG 2.1 AA compliance.
- **Maintainability:** API versioning; ≥ 70% test coverage in core flows.
- **Portability:** latest 2 major versions of browsers and iOS/Android supported.
- **Scalability:** stateless APIs with horizontal scaling, workers for notifications, read replicas for reporting.

---

# 5 Other Requirements

**5.1 Internationalization and Localization**
The system must support at least English and Spanish from launch. All user-facing text should be externalized to resource files so that new languages can be added without code changes. Dates and times will always be stored in UTC and converted to the user's local time.

**5.2 Privacy and Legal Compliance**
The platform must comply with local data protection regulations (HIPAA-equivalent, GDPR principles). Users must provide explicit consent before storing or processing health information. Privacy and Terms of Service pages must be accessible from all main views.

**5.3 Maintenance and Support**
A basic support process must be defined, including a ticketing channel with response times no longer than 24 hours for regular issues and faster responses for critical incidents. Maintenance releases should occur at least once per quarter.

**5.4 Integration and Interoperability**
The system must expose a RESTful API for third-party integration (e.g., EHR, billing systems).

Providers must also have access to an iCal export for their schedules. Integration documentation must be delivered alongside the product.

### 5.5 Accessibility and Usability

The product must follow WCAG 2.1 AA accessibility guidelines. The interface should remain simple to use, ensuring that patients can complete a booking in no more than three steps.

# Appendix A – Data Dictionary

- **patient_id** (UUID) – F1
- **provider_id** (UUID) – F2, F11
- **appointment_id** (UUID) – F4–F5–F8
- **slot_start/slot_end** (datetime TZ) – F4–F5–F19
- **status** (enum: booked, rescheduled, canceled, completed, no_show) – F13
- **notify_channel** (enum: email, sms, push) – F14
- **notify_lead_minutes** (int) – F14
- **audit_event** (actor_id, action, entity_id, ts) – F16
- **availability_rule** (weekday, start, end, location_id) – F11
- **exception_window** (start, end, reason) – F11