# ­­my clinic

# Smart Appointment Booking – CRM Project

## Phase 1: Problem Understanding & Industry Analysis

* Goal: Understand what we are building and why.

### 1. Requirement Gathering

• Studied the manual process of hospitals and clinics where appointments were handled with registers and phone calls.  
• **Identified problems:**  
 – Double bookings.  
 – Missed appointments due to no reminders.  
 – No proper tracking of workload.  
 – No centralized reporting.  
**• Finalized requirements:**  
 ○ Online patient appointment booking.  
 ○ Auto-assign doctors based on specialization and availability.  
 ○ Enforce daily appointment limits for doctors.  
 ○ Reject booking if slots are full.  
 ○ Send confirmation/rejection emails.  
 ○ Send reminder notifications one day before appointment.  
 ○ Doctors update status after completion.  
 ○ Admin dashboards for workload, patient flow, and trends.

### 2. Stakeholder Analysis

• Patients – Book appointments and receive notifications.  
• Doctors – View/manage appointments and update completion status.  
• Admin/Clinic Manager – Oversee bookings, manage resources, generate reports.

### 3. Business Process Mapping

Patient books appointment → System checks doctor availability → Appointment confirmed/rejected → Email sent → Reminder notification sent → Doctor updates status → Admin monitors reports.

### 4. Industry-specific Use Case Analysis

* Healthcare faces high patient load, inefficient manual processes, and communication gaps. The smart appointment solution addresses these by:  
  – Automating scheduling and avoiding overlaps.  
  – Improving patient communication with reminders.  
  – Enforcing doctor workload limits.  
  – Providing insights to management via dashboards.

### 5. AppExchange Exploration

Reviewed Salesforce Health Cloud & Appointment apps. Final choice: build a custom appointment booking solution to demonstrate Salesforce Admin + Developer concepts.



## Phase 2: Org Setup & Configuration

* Goal: Prepare Salesforce environment.

### 1. Salesforce Editions

Used Salesforce Developer Edition Org for implementation.

### 2. Company Profile Setup

Configured clinic information, local time zone, and INR currency for reporting.

### 3. Business Hours & Holidays

* Set hospital working hours (9 AM – 5 PM).
* Added weekends and public holidays to restrict booking.

### 4. Fiscal Year Settings

* Enabled Standard Fiscal Year (Jan–Dec) for reporting.

### 5. User Setup & Licenses

Created users: Admin/Manager, Doctors, Patients (Community/Portal).

### 6. Profiles

* Admin – Full access.
* Doctors – Manage appointments.
* Patients – Limited access for booking.

### 7. Roles

Hierarchy: Admin/Clinic Manager → Doctors → Patients.

### 8. Permission Sets

Created additional permission sets for doctors to access reports/dashboards.

### 9. OWD (Org-Wide Defaults)

* Appointments – Private.
* Doctors – Public Read Only.
* Patients – Private.

### 10. Sharing Rules

* Appointments shared with Admin role automatically.

### 11. Login Access Policies

* Doctors restricted to login 9 AM – 5 PM.
* Admin – full access.
* Patients – 24x7 access.

### 12. Dev Org Setup

* Developer Org used as sandbox for building and testing.

### 13. Sandbox Usage

* Documented process for Sandbox → Production deployment in real-world usage.

### 14. Deployment Basics

* Prepared Change Sets for moving configurations. Documented SFDX/VS Code deployment methods.

# Phase 3: Data Modeling & Relationships

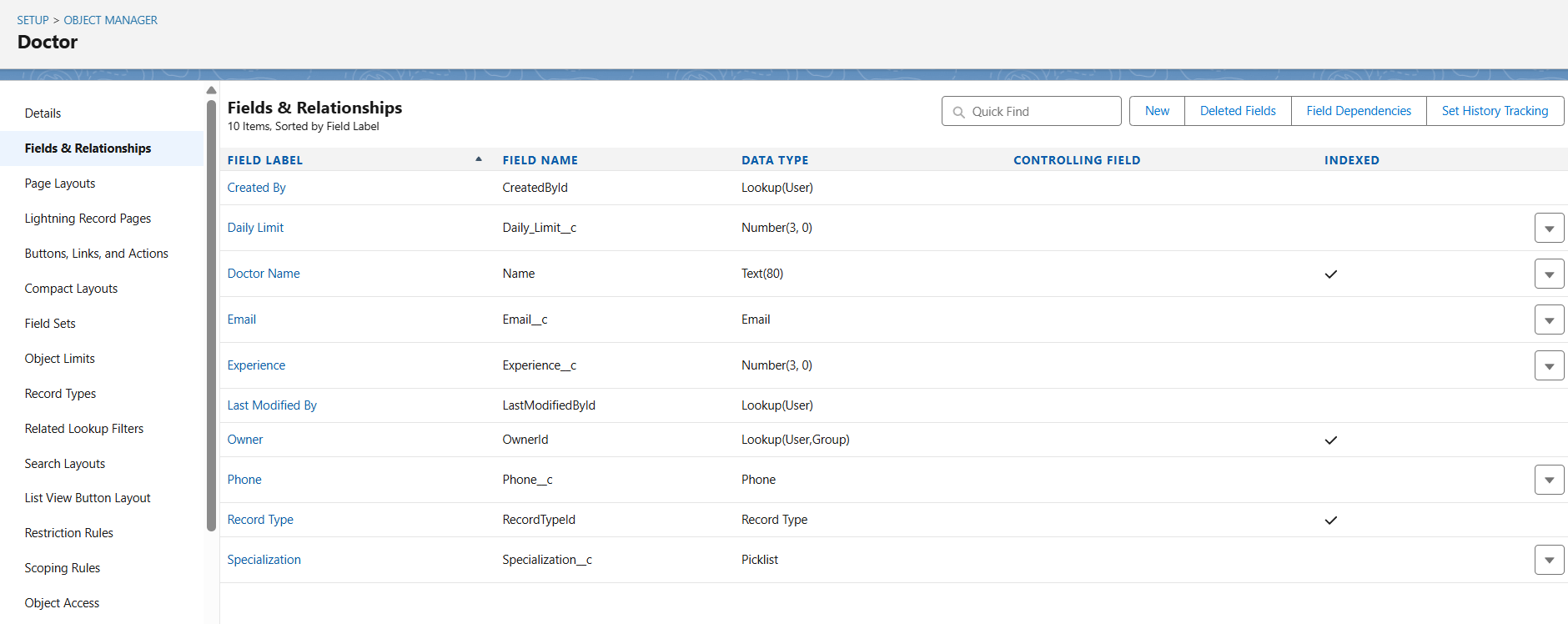
* Goal: Build the data structure for the Appointment Booking System.

## 1. Standard & Custom Objects

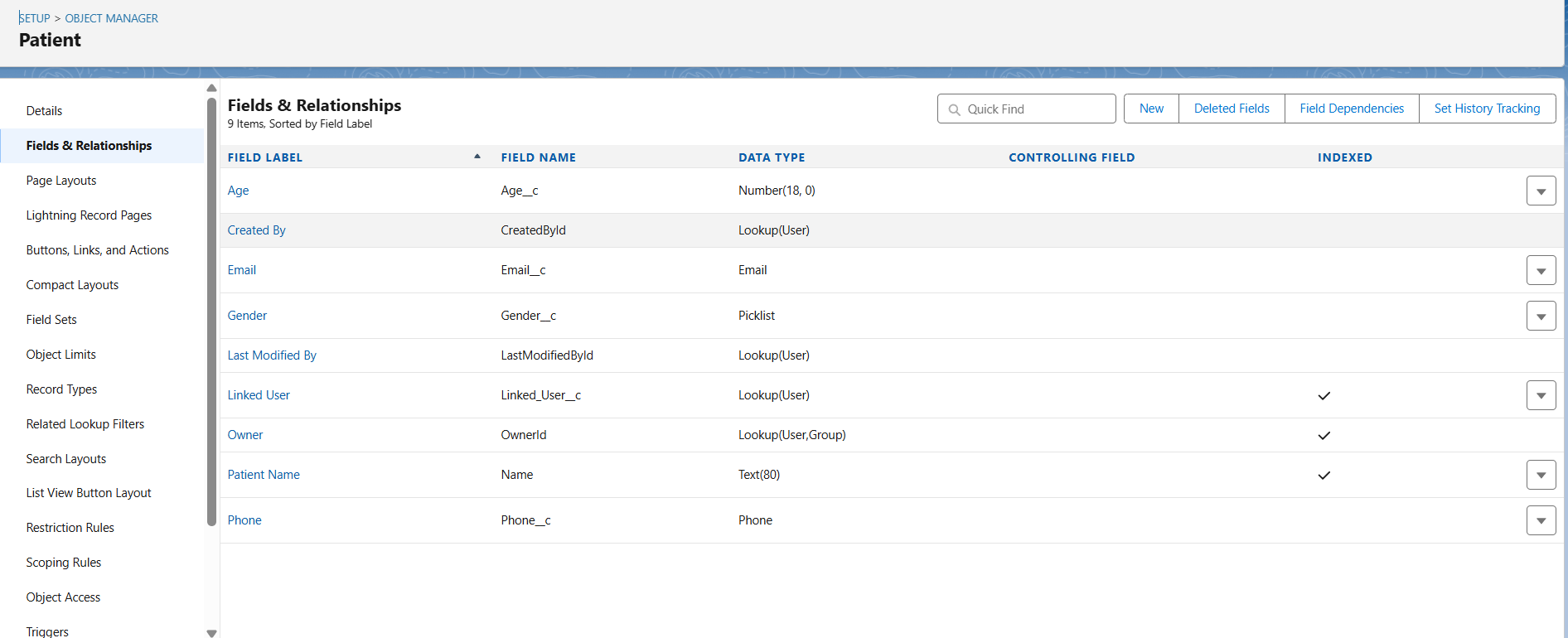
• Standard: Contact (for reuse if needed as external users).  
• Custom:  
 - Doctor → Stores information about doctors in the clinic.  
 - Patient → Stores information about patients.  
 - Appointment → Stores booking details (date, time, doctor, patient, status).

## 2. Fields

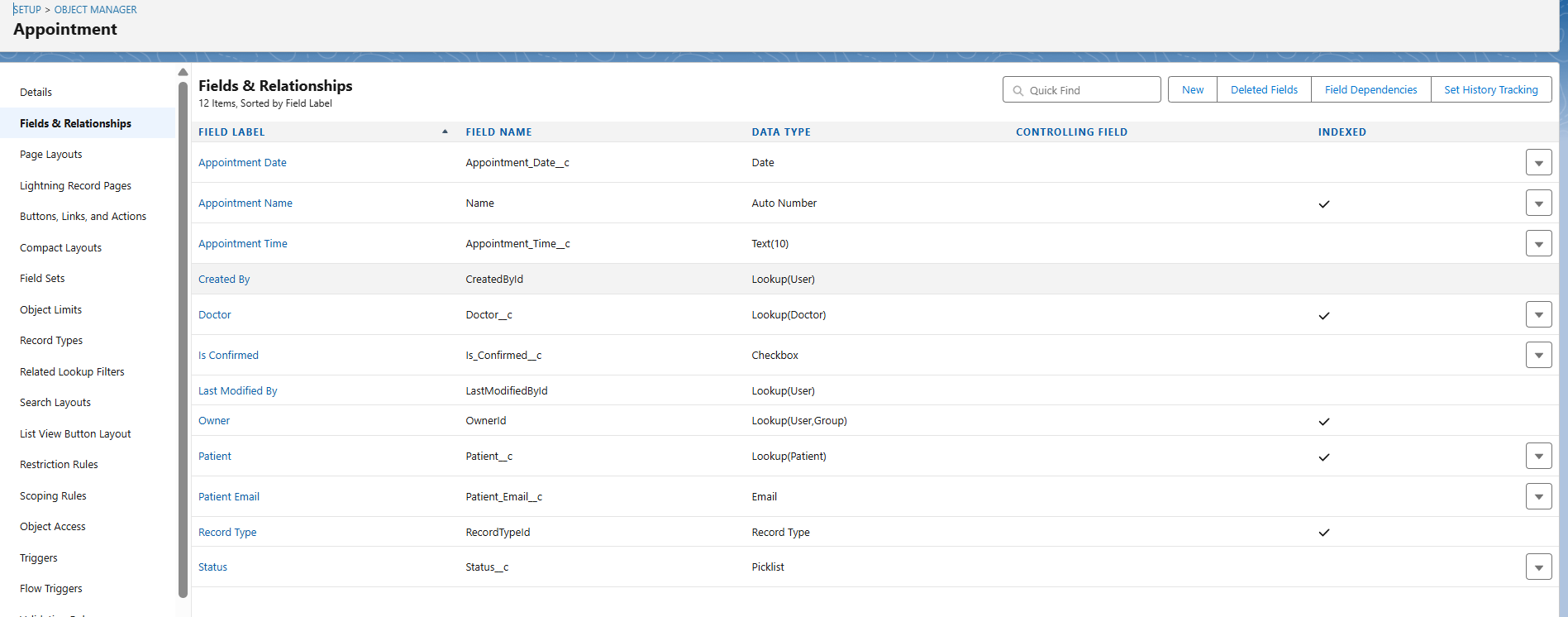
• Doctor: Name, Specialization, Experience, Email, Phone, Max Daily Appointments.



• Patient: Name, Age, Gender, Email, Phone, Linked User.

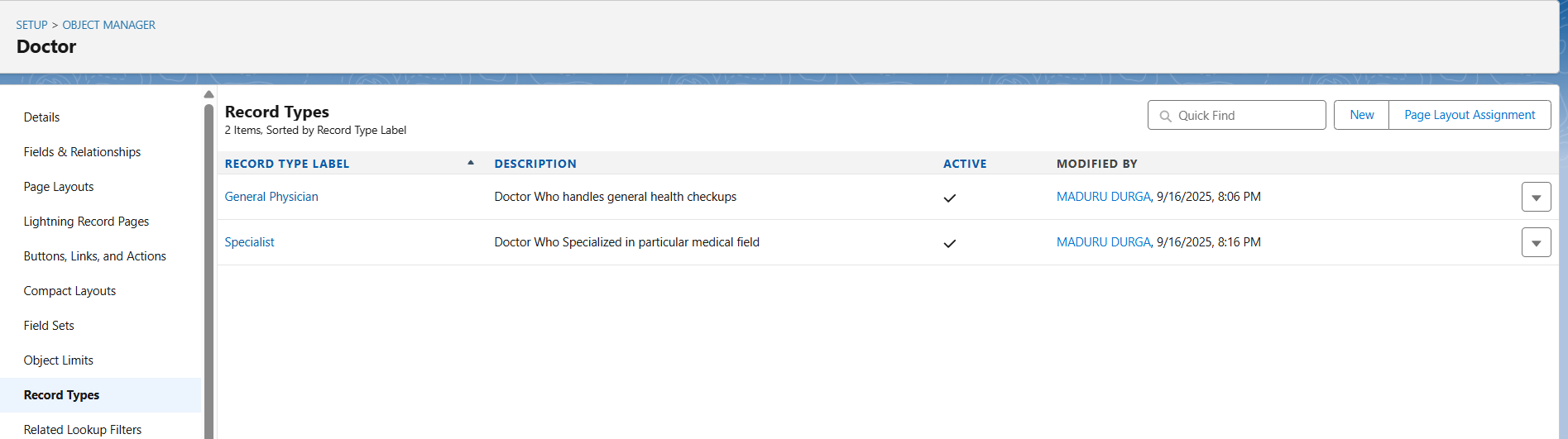


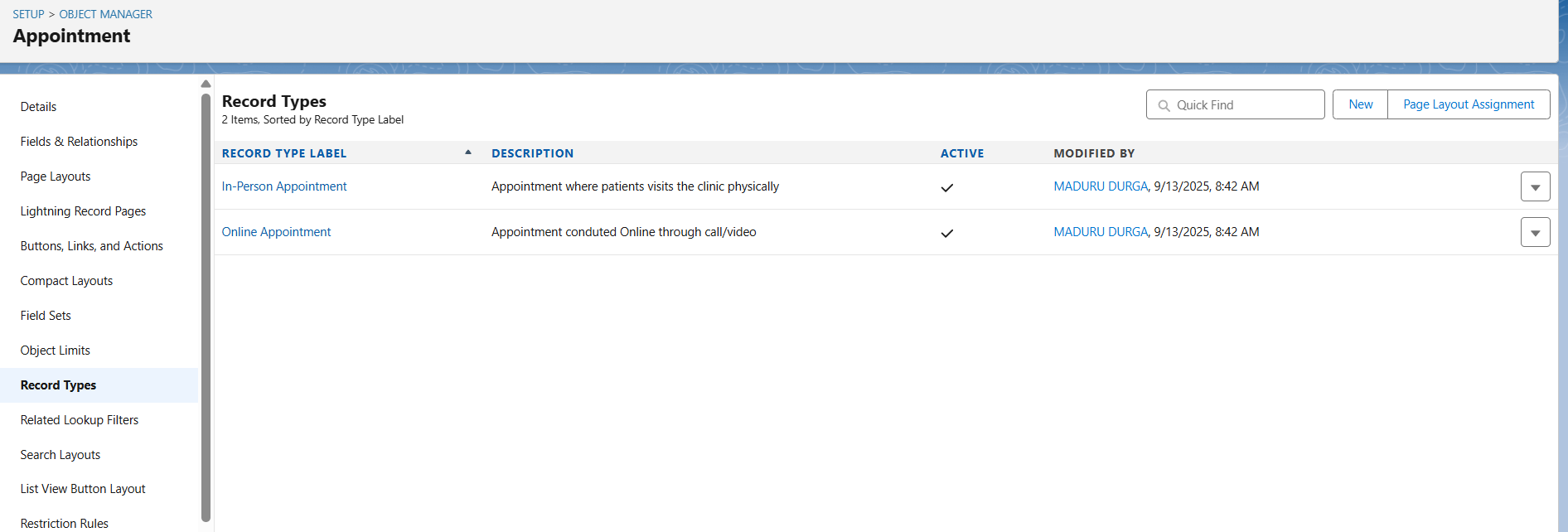
• Appointment: Appointment Number, Appointment Date, Appointment Time, Status, Doctor (Lookup), Patient (Lookup).



## 3. Record Types

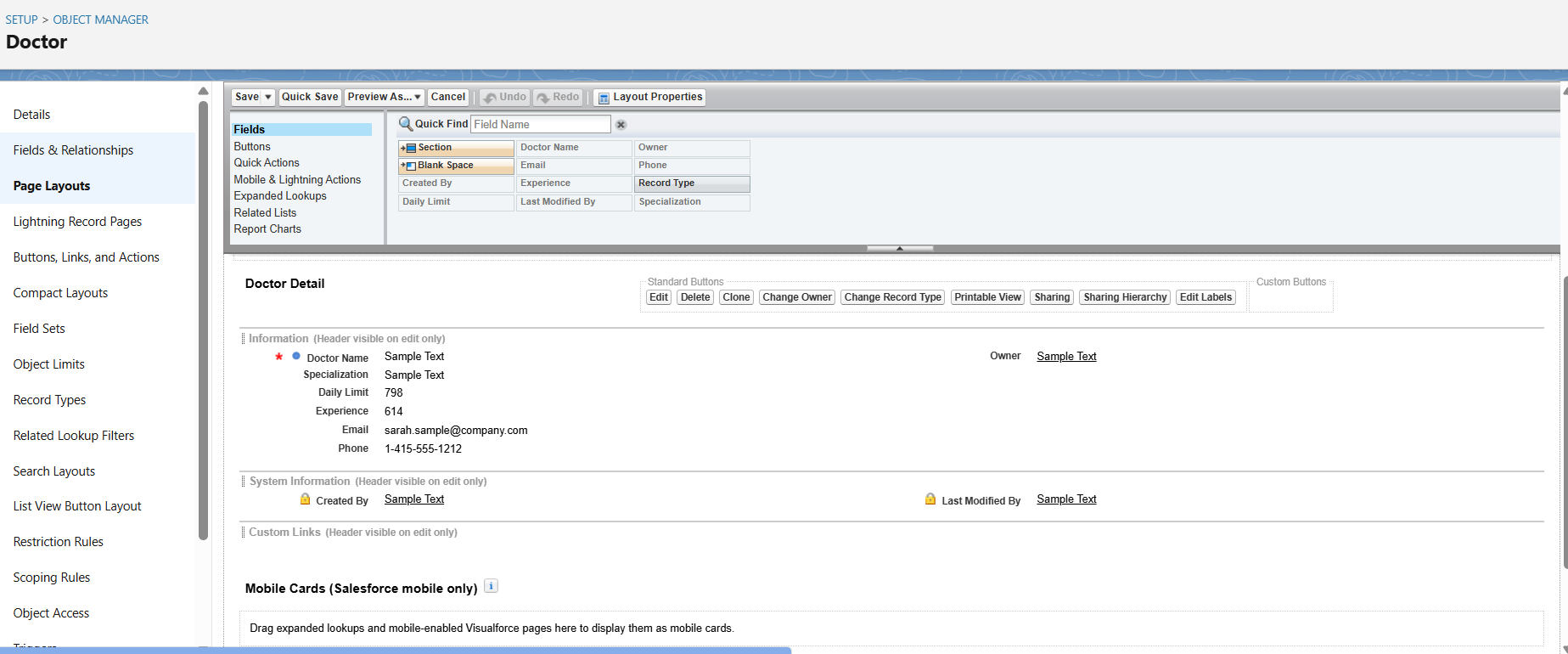
• Doctor:  
 - General Physician → Handles general checkups.  
 - Specialist → Handles specific treatments (e.g., Cardiologist).

  
• Appointment:  
 - In-Person Appointment → Patient visits the clinic physically.  
 - Online Appointment → Appointment done virtually (video/phone).

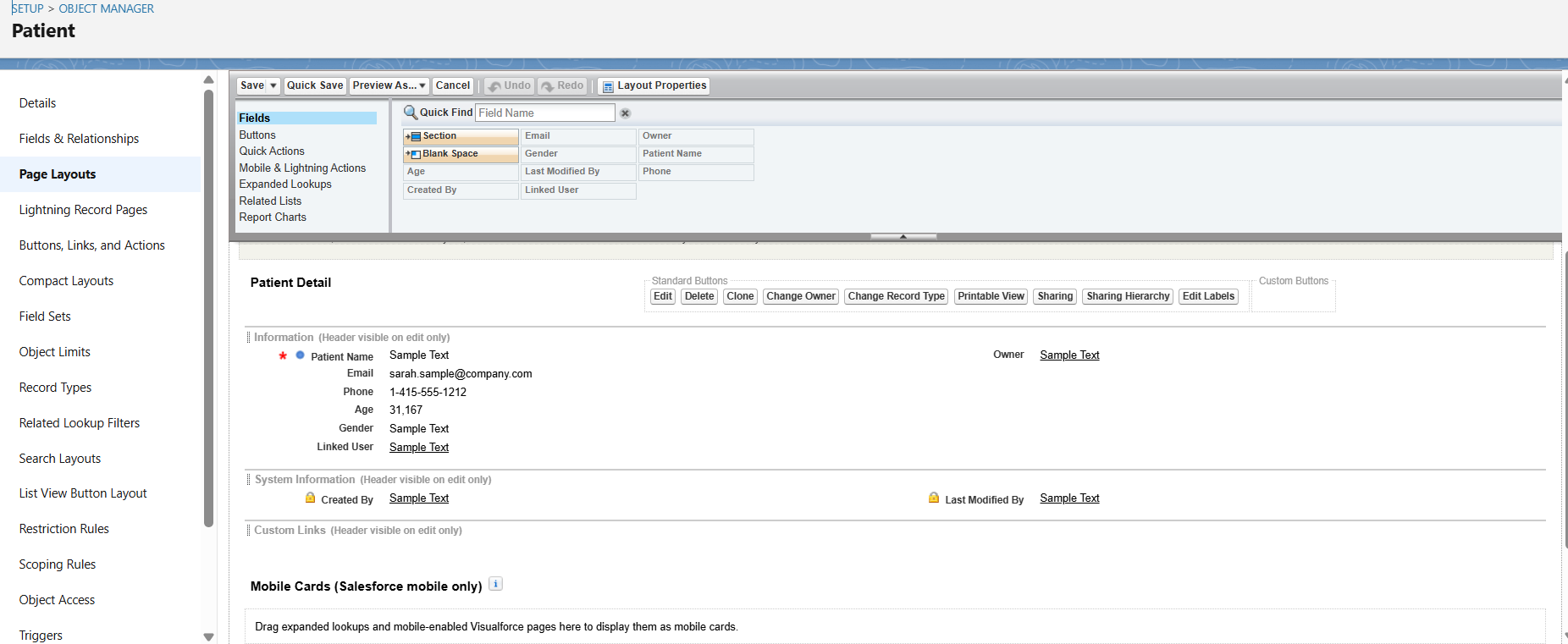


## 4. Page Layouts

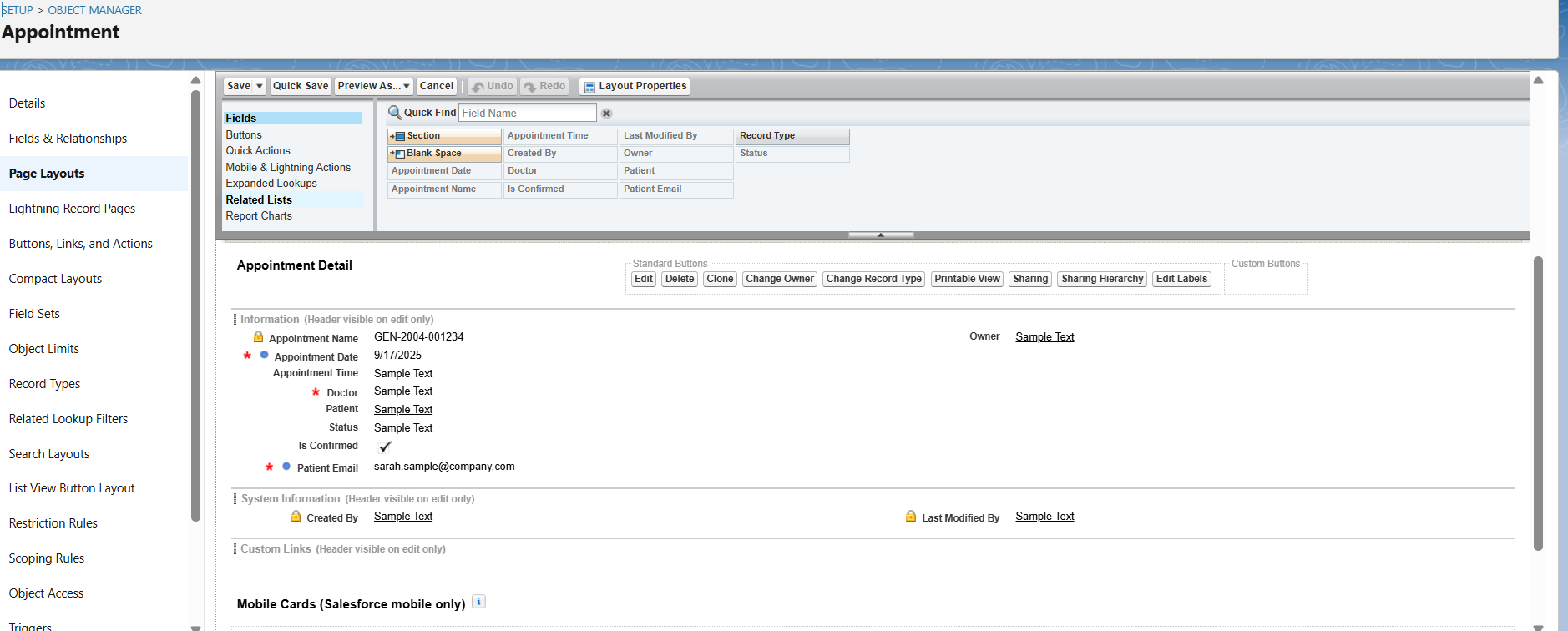
• Doctor Page: Shows all doctor details + related list of Appointments.



• Patient Page: Shows patient details + related list of Appointments.

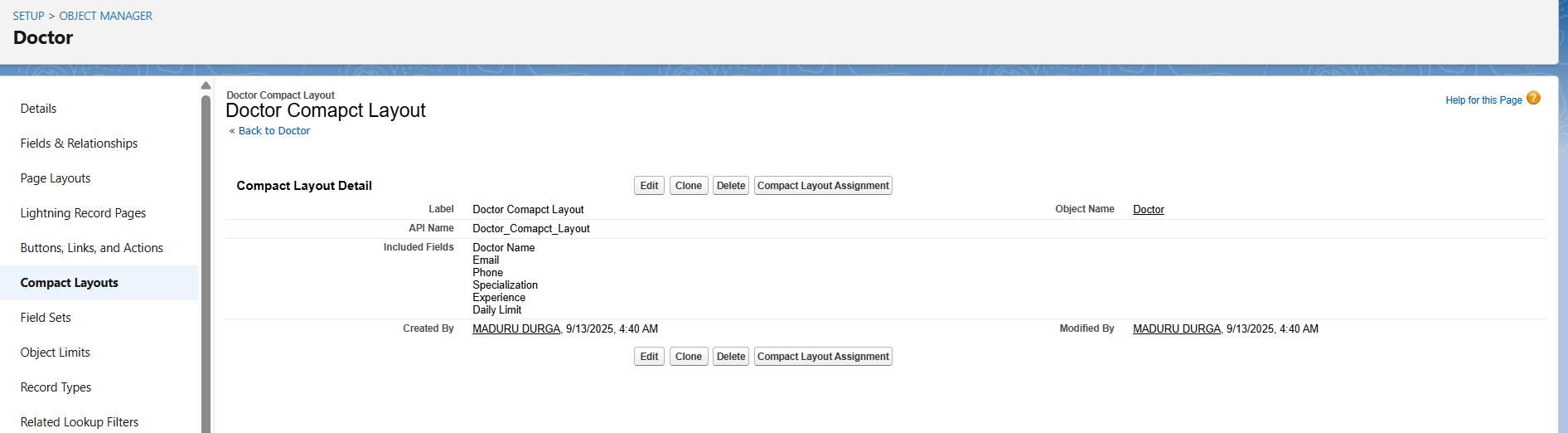


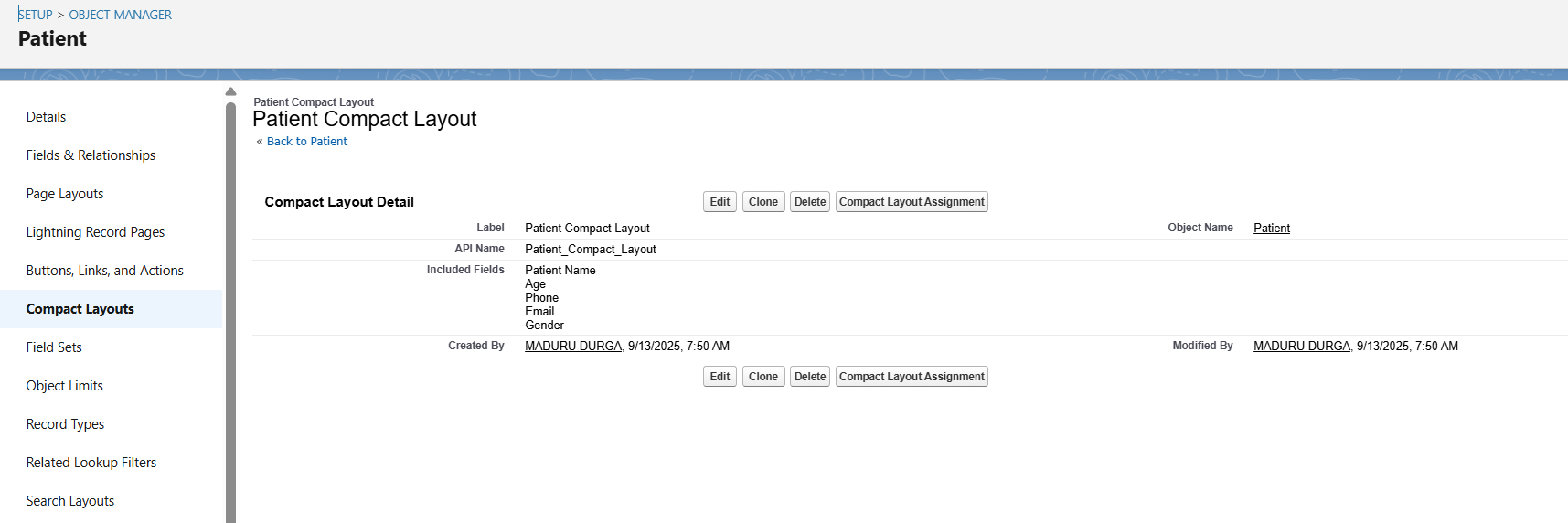
• Appointment Page: Shows booking details, with lookups to Doctor and Patient.

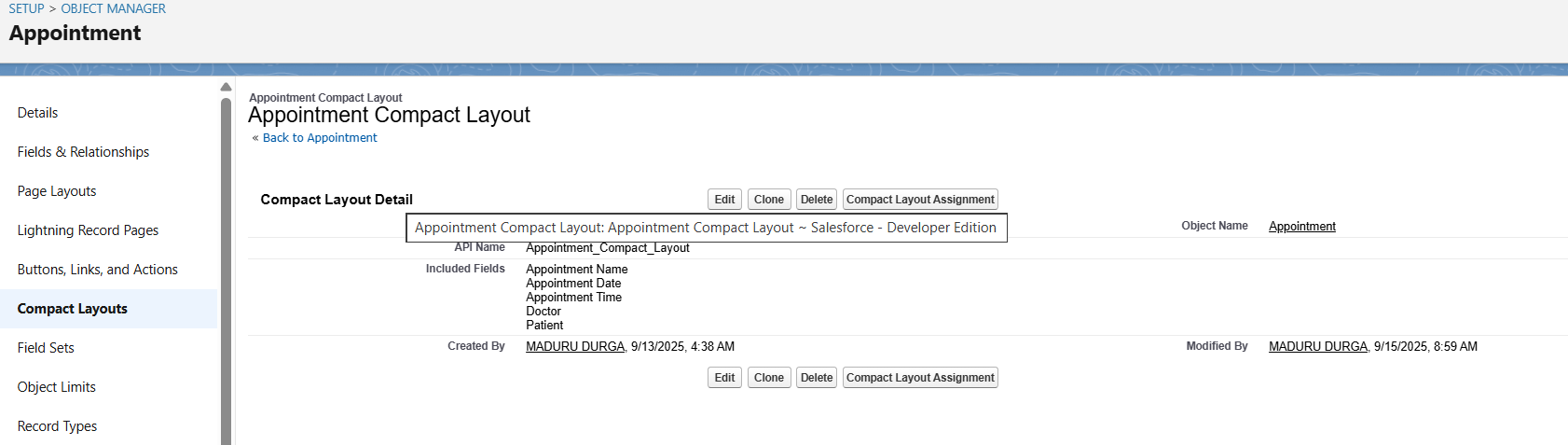


## 5. Compact Layouts

• Doctor Compact Layout: Doctor Name, Specialization, Phone, Email, Experience, Daily Limit.

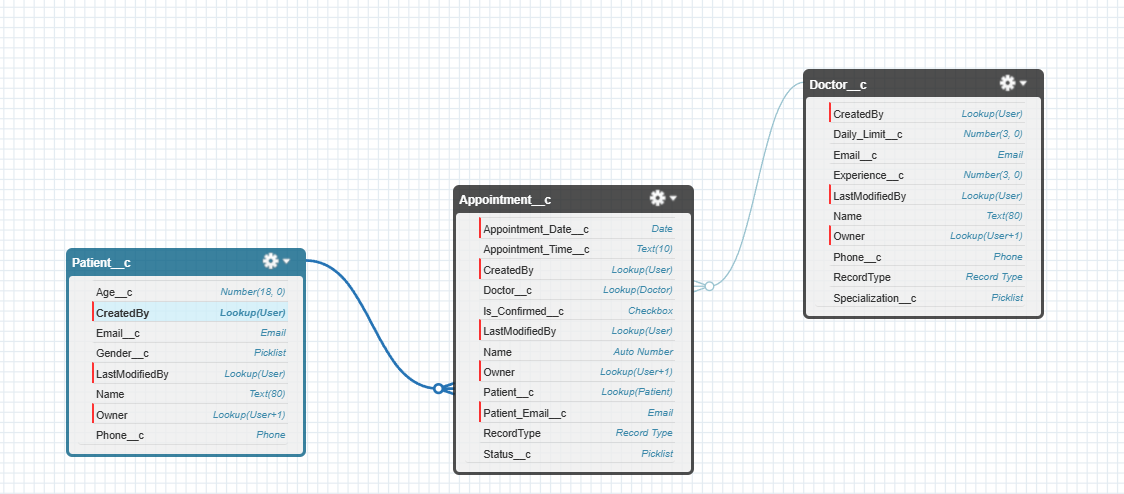
  
• Patient Compact Layout: Patient Name, Age, Phone, Email.

  
• Appointment Compact Layout: Appointment Number, Appointment Date, Status, Doctor.



## 6. Schema Builder

• Visualized objects Doctor, Patient, Appointment with lookup relationships:  
 - Appointment → Doctor (Lookup).  
 - Appointment → Patient (Lookup).  
• Screenshot taken to show relationships for the report.



## 7. Lookup vs Master-Detail vs Hierarchical

• Appointment ↔ Doctor: Lookup (appointments reference doctors, but do not own them).  
• Appointment ↔ Patient: Lookup (appointments reference patients, but patients exist independently).  
• No Master-Detail needed because records can exist independently.  
• No Hierarchical used (that is only for user object).

## 8. Junction Objects

• Not needed in this system, because one appointment is always linked to one doctor and one patient.  
• If in future one appointment could involve multiple doctors, a junction object (e.g., Appointment Participants) would be required.

## 9. External Objects

• Not used here.  
• Could be considered in future if the clinic integrates with external systems (e.g., insurance providers or external hospital DB).