**Project Title: Healthcare Follow-up & Preventive Care CRM**

**Phase 1:-** **Problem Understanding & Industry Analysis**

**Problem Statement**

Hospitals and clinics face significant challenges in ensuring patients attend follow-up consultations and participate in preventive healthcare programs such as vaccinations, routine health checkups, and chronic illness monitoring. Manual tracking of patient visits and communication often leads to missed appointments, delayed treatments, and poor patient engagement.

These issues not only affect patient health outcomes but also result in inefficiencies and revenue loss for healthcare providers. There is a growing need for a **centralized CRM solution** that can automate appointment scheduling, send timely reminders, and provide visibility into patient follow-up compliance and preventive care participation.

By implementing a Salesforce-based CRM system, hospitals can:

* Manage patient records efficiently.
* Automate reminders for follow-ups and preventive care.
* Provide doctors and staff with real-time access to treatment history.
* Enable management to monitor performance with dashboards and reports.

**Project Overview**

The proposed solution is a Salesforce-based **Healthcare Follow-up & Preventive Care CRM** designed to help hospitals and clinics improve patient engagement. The system will manage patient records, automate appointment scheduling, and send timely reminders for follow-ups and preventive care programs such as vaccinations, health checkups, and chronic disease monitoring.

**Who will use it?**

* **Doctors & Nurses** → to track patient treatment plans and follow-ups.
* **Hospital Admin Staff** → to schedule appointments, manage records, and analyze reports.
* **Patients** → to receive reminders, check appointments, and stay engaged with preventive care.

**How it will help?**  
This solution ensures that patients do not miss important medical appointments, improves preventive care participation, and reduces manual work for hospital staff. It will provide real-time dashboards to management, helping them monitor patient inflow, follow-up completion rates, and overall hospital efficiency.

**Objectives**

The main goals of the project are:

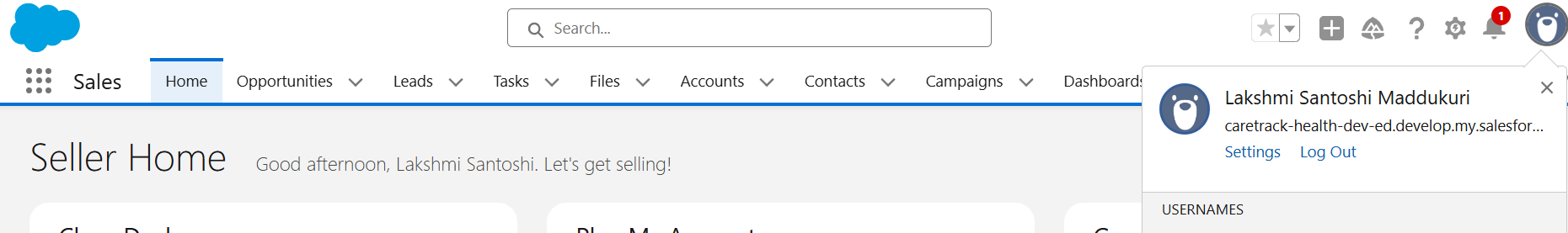
1. **Improve Efficiency** – Automate appointment scheduling and follow-up reminders.
2. **Automate Manual Tasks** – Reduce manual tracking of patient visits and preventive care programs.
3. **Provide Better Reporting** – Generate dashboards and reports on patient inflow, missed vs. completed follow-ups, and preventive care participation.
4. **Ensure Data Accuracy** – Maintain complete patient medical history and treatment details in one place.
5. **Enhance Patient Experience** – Keep patients engaged through timely notifications, reminders, and easy access to their appointment details.

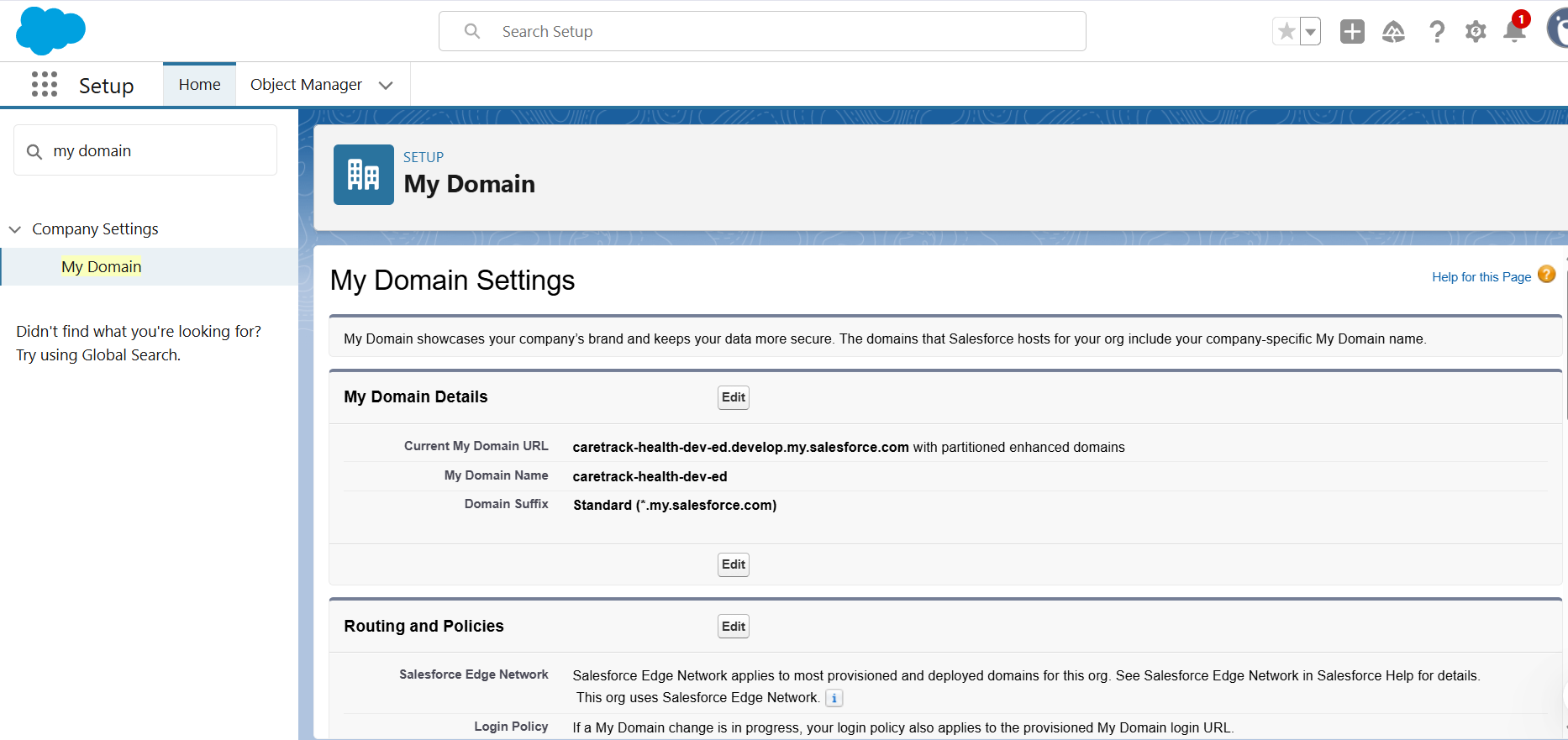
**Phase 2:- Documentation — Org Setup & Configuration**

## ****1. Salesforce Edition & Dev Org Setup****

We used a **Salesforce Developer Edition Org** to build this project.

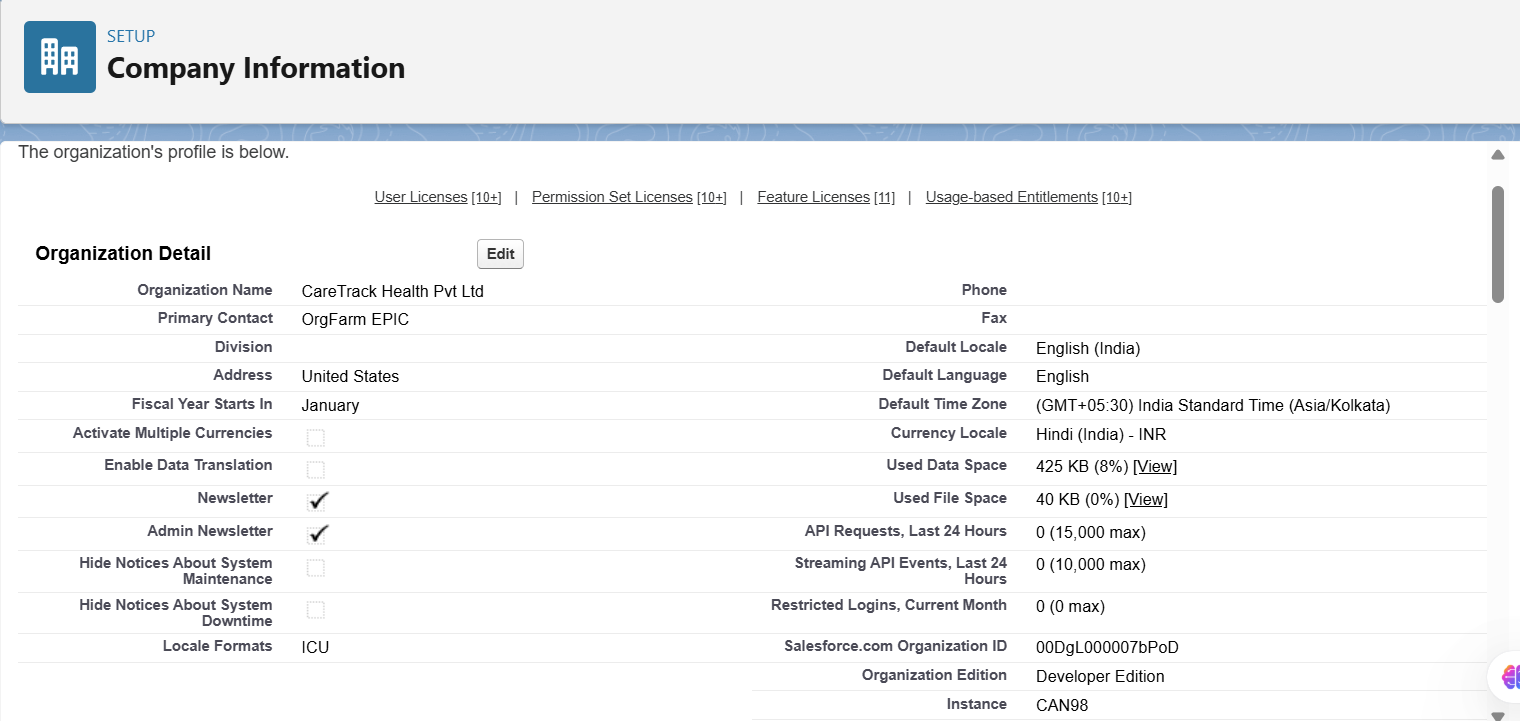
* Enabled **My Domain** and deployed successfully.
* Verified login using the new domain.



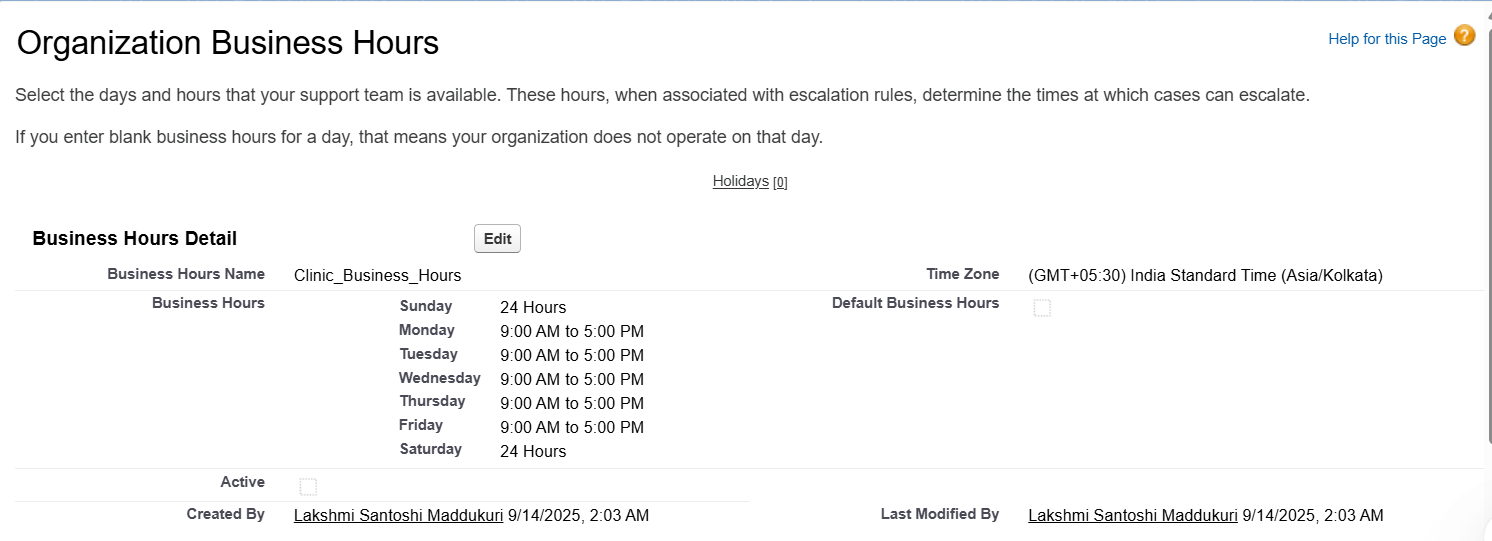
****

## ****2. Company Profile Setup****

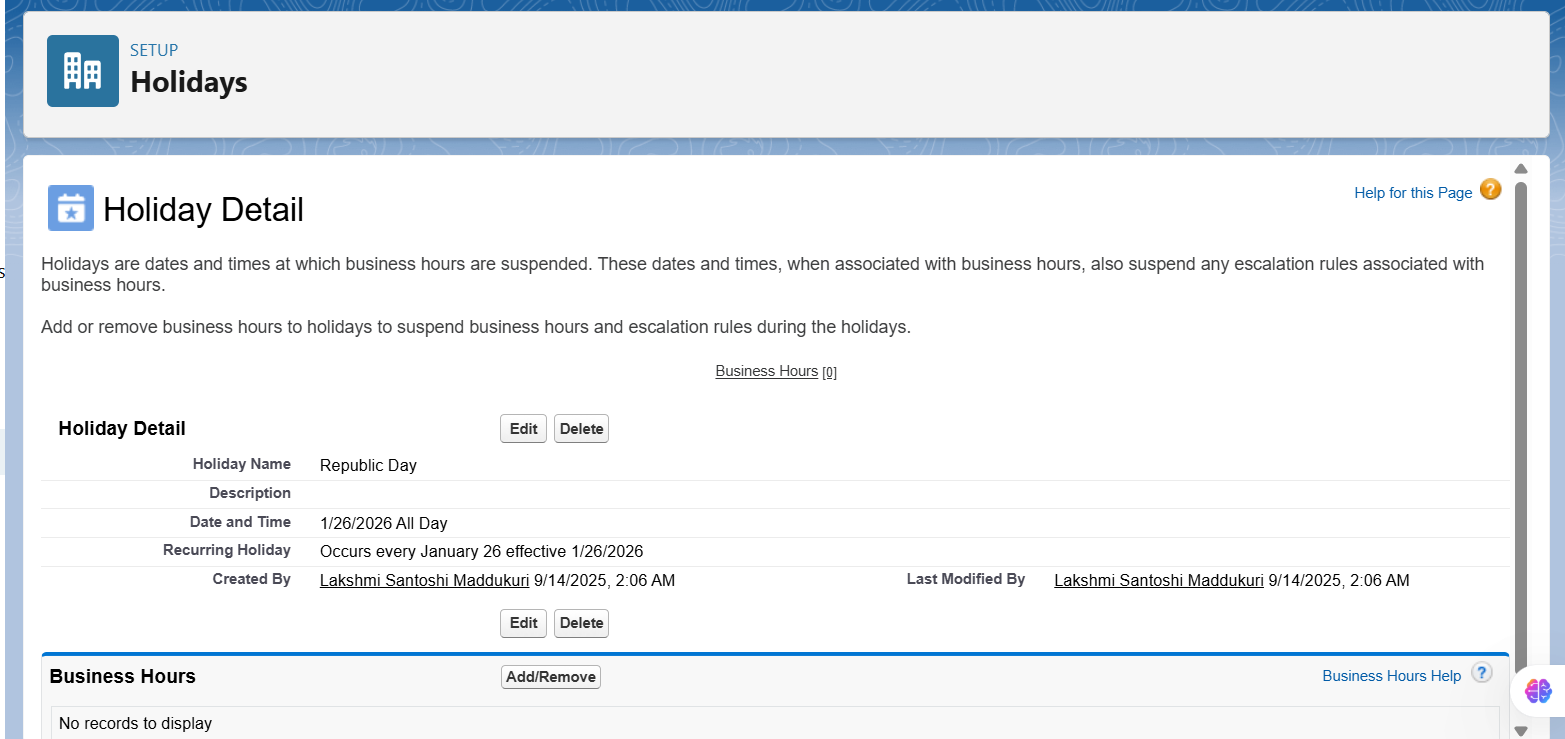
* Company Name: CareTrack Health Pvt Ltd
* Default Time Zone: Asia/Kolkata (IST)
* Fiscal Year: Standard (Jan–Dec)

**

* Business Hours: 9 AM – 6 PM IST, Mon–Sat

**

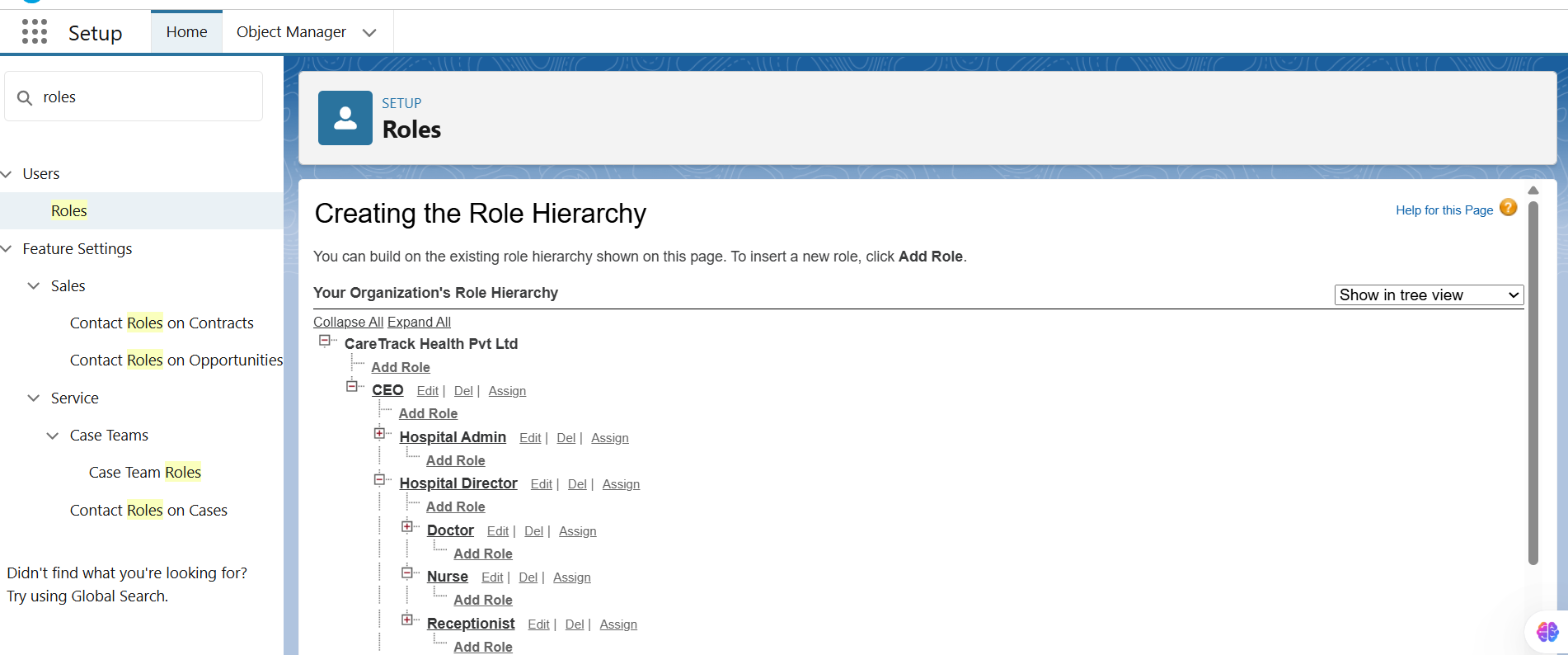
* Holidays Configured: Republic Day, Independence Day



## ****3. Roles (Role Hierarchy)****

We created a role hierarchy to reflect hospital structure:

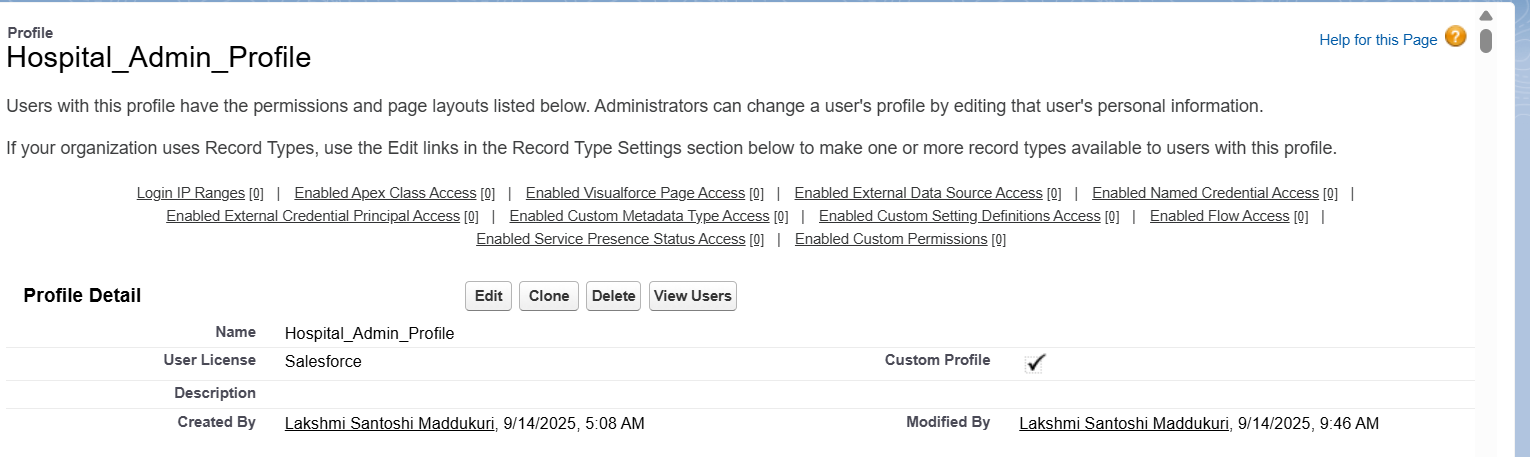
CEO  
↳ Hospital Admin  
 ↳ Hospital Director  
  ↳ Doctor  
  ↳ Nurse  
  ↳ Receptionist

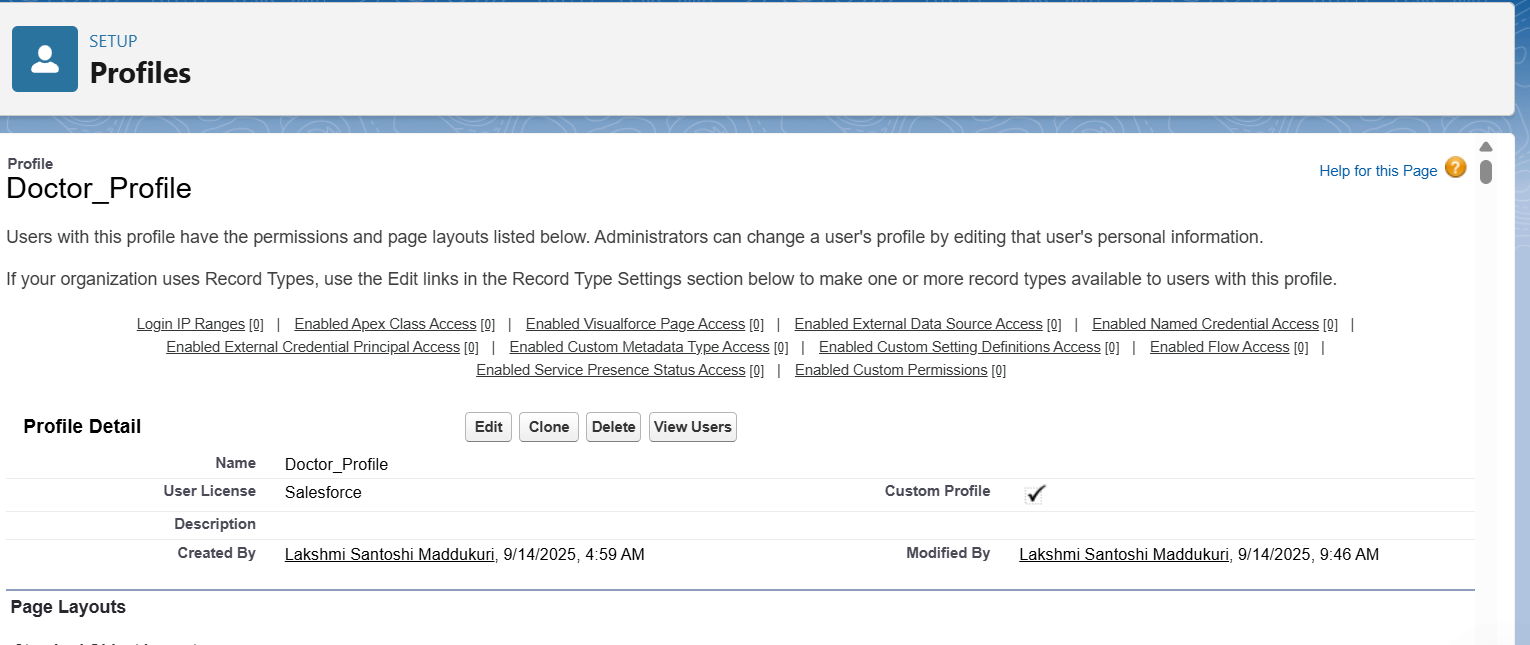


## ****4. Profiles****

Created/Cloned profiles to control object and field-level access.

* **Hospital\_Admin\_Profile** → cloned from System Administrator.
* **Doctor\_Profile** → cloned from Standard User.
* **Nurse\_Profile**.
* **Receptionist\_Profile**.

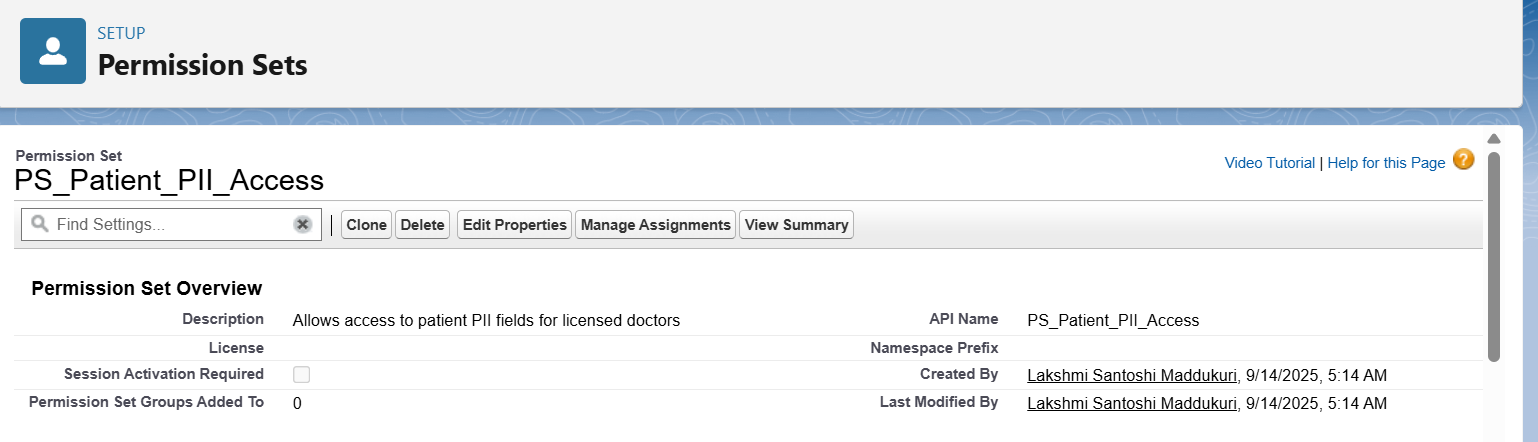
****



## ****5. Permission Sets****

Created permission sets for additional flexibility:

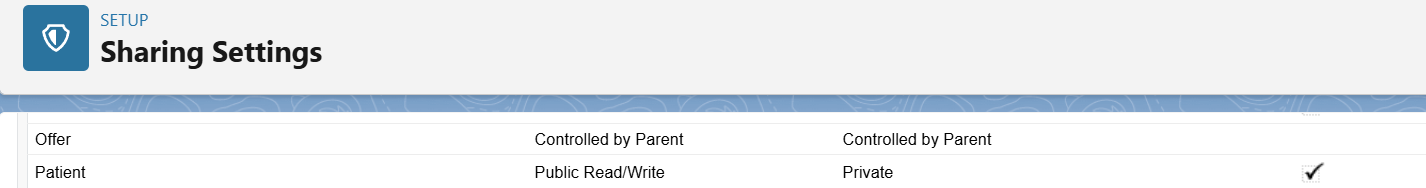
* **PS\_Patient\_PII\_Access** → Sensitive patient fields (Doctors only).
* **PS\_Bulk\_Data\_Importer** → Planned (will be required for Admin bulk data operations in later phases).
* **PS\_Experience\_User** → Planned (will be required when enabling the Patient Portal in Phase 6).



## ****6. Organization-Wide Defaults (OWD)****

Configured strict privacy for medical data.

* Patient\_\_c → Private
* Appointment\_\_c → Controlled by Parent
* Treatment\_Plan\_\_c → Controlled by Parent
* FollowUp\_Schedule\_\_c → Controlled by Parent

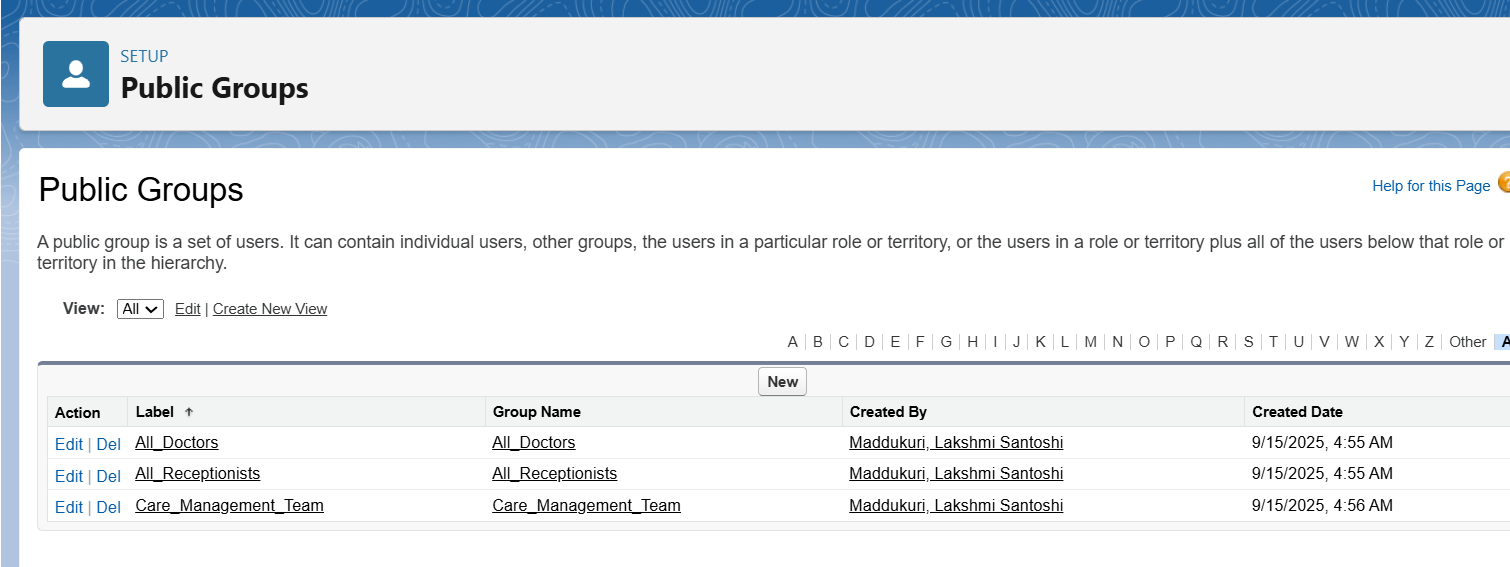




## ****7. Public Groups****

Created groups for sharing rules and collaboration.

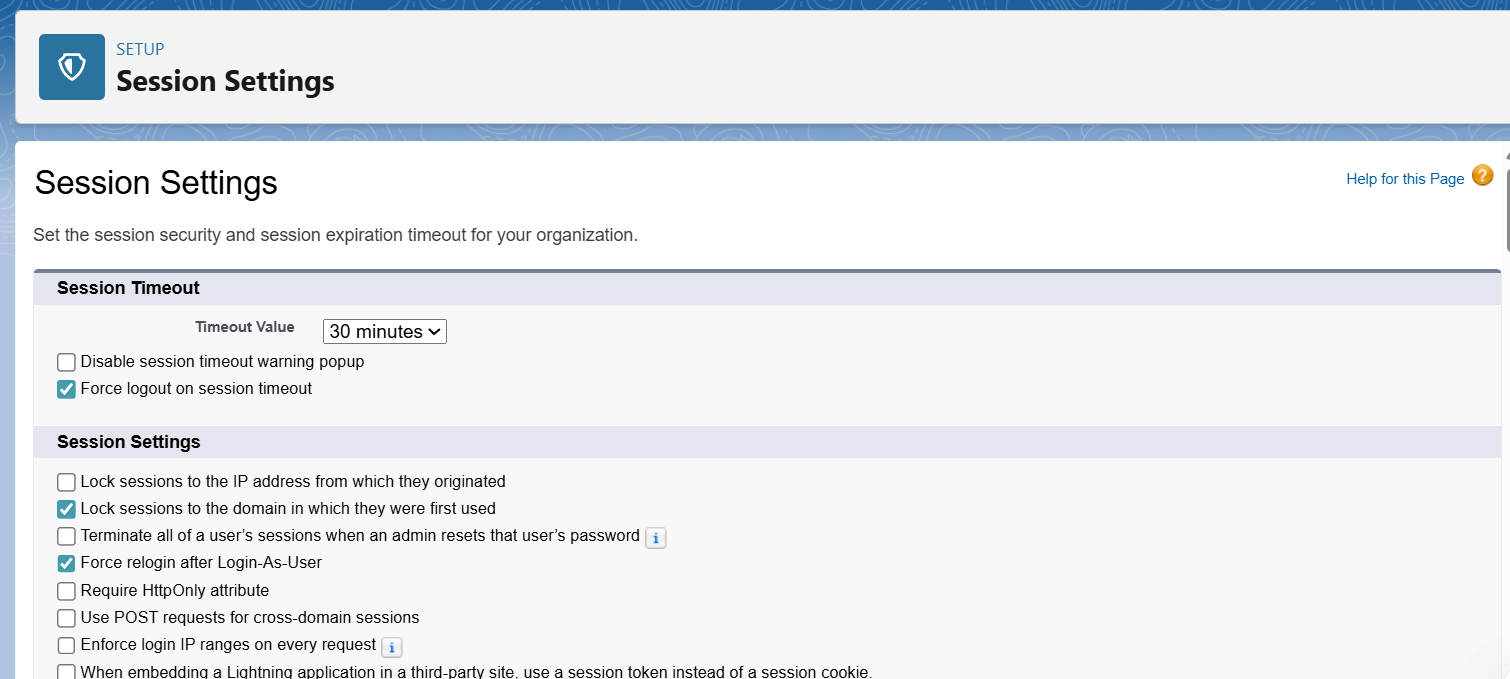
* All\_Doctors → includes Role: Doctor.
* All\_Receptionists → includes Role: Receptionist.
* Care\_Management\_Team → includes Hospital Admin + Dept Heads.

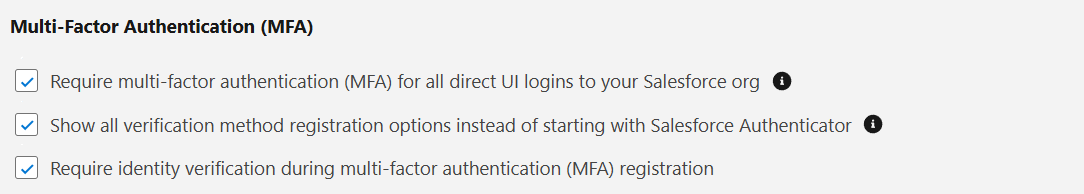


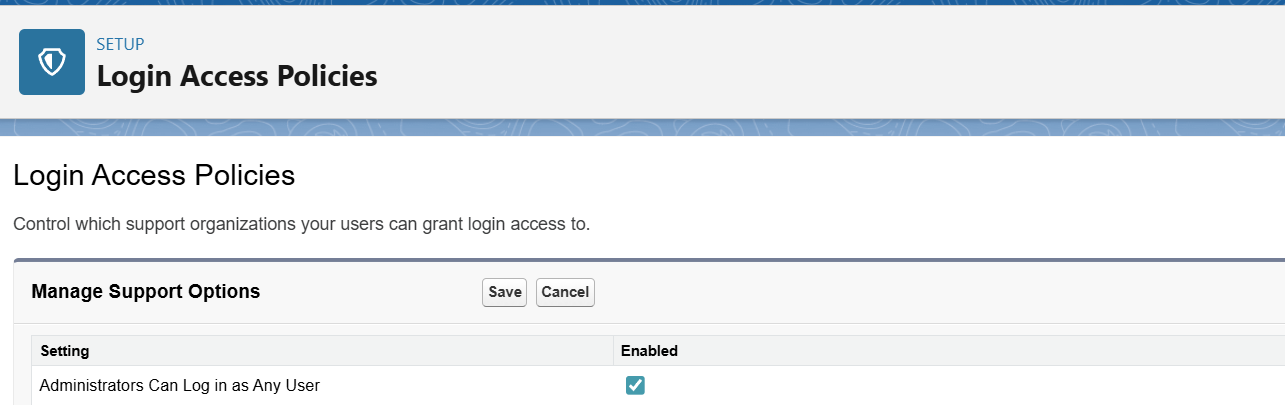
## ****8. Login Access & Security****

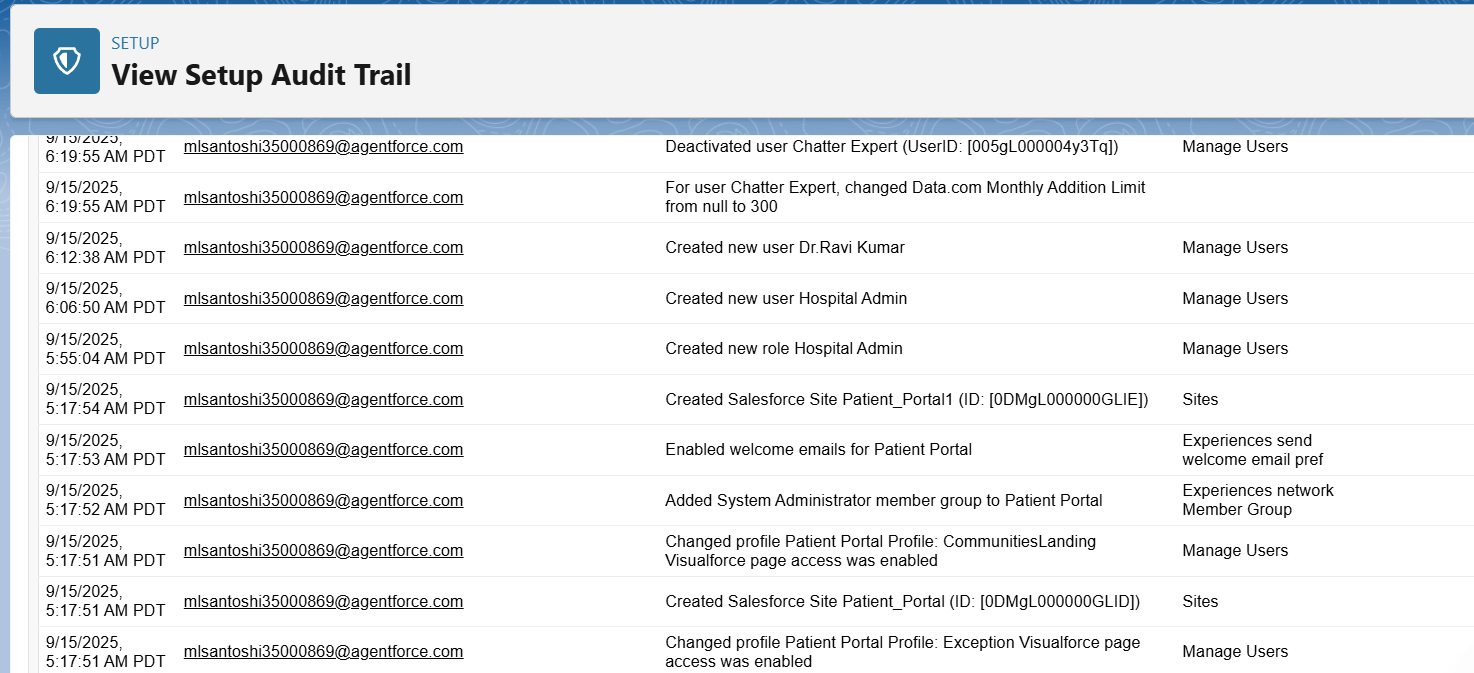
Security settings were configured to ensure safe access:

* Session Timeout: 30 minutes idle.
* Multi-Factor Authentication (MFA): Enabled for internal users.
* Login Access Policies: Admin can log in as any user (testing).
* Audit Trail & Login History: Enabled for monitoring.



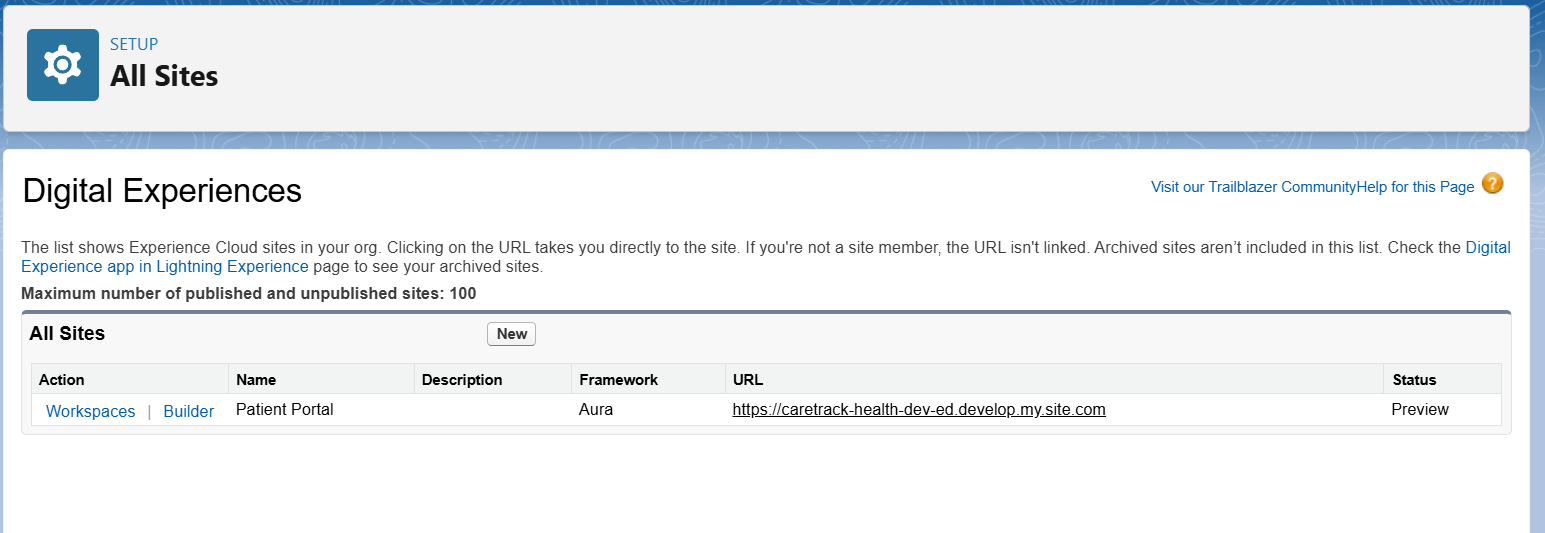






## ****9. Dev Org Setup Specifics****

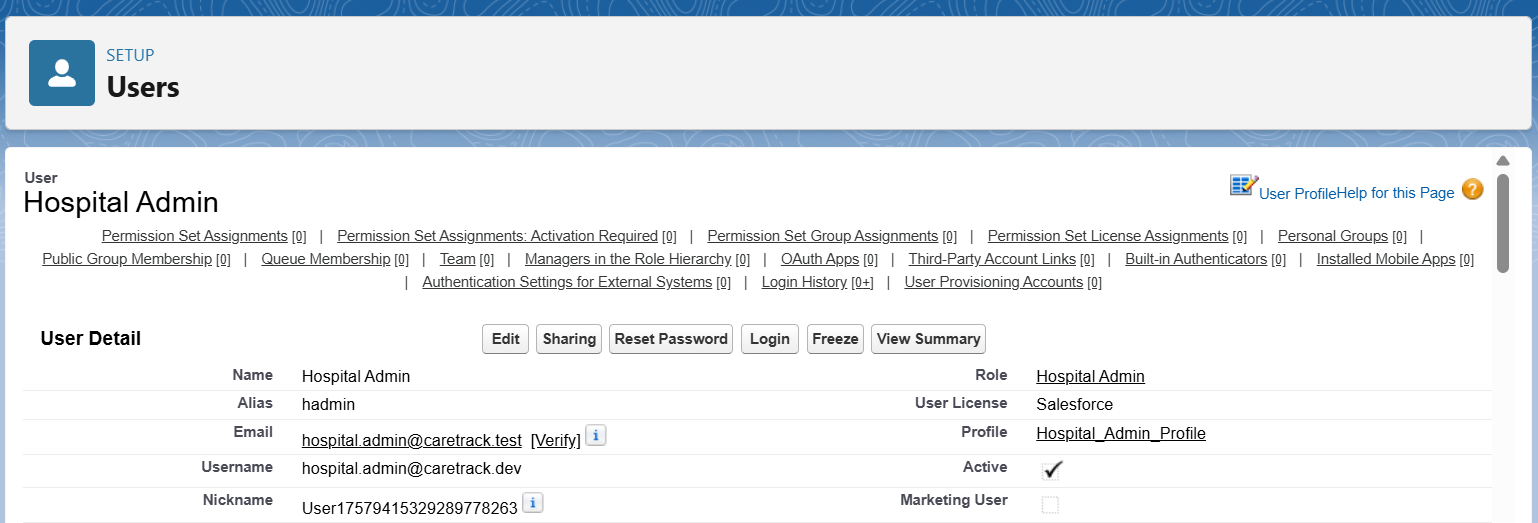
* My Domain: Enabled & deployed.
* Experience Cloud: Enabled → “Patient Portal” created (Draft).
* API Access: Confirmed for System Administrator profile.

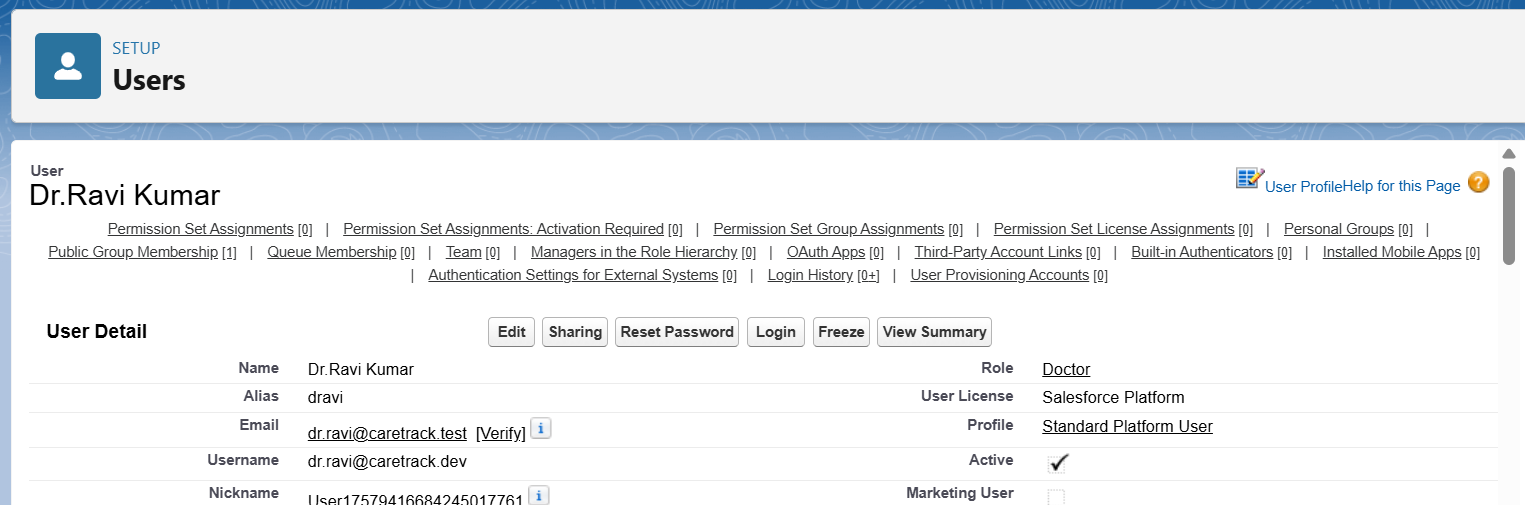


## ****10. Users****

Sample users created for testing:

* Hospital Admin → hospital.admin@caretrack.dev (System Admin).
* Doctor Ravi → dr.ravi@caretrack.dev (Doctor\_Profile, Role = Doctor).
* Nurse Anu → Planned (license limitation).
* Receptionist Maya → Planned (license limitation).





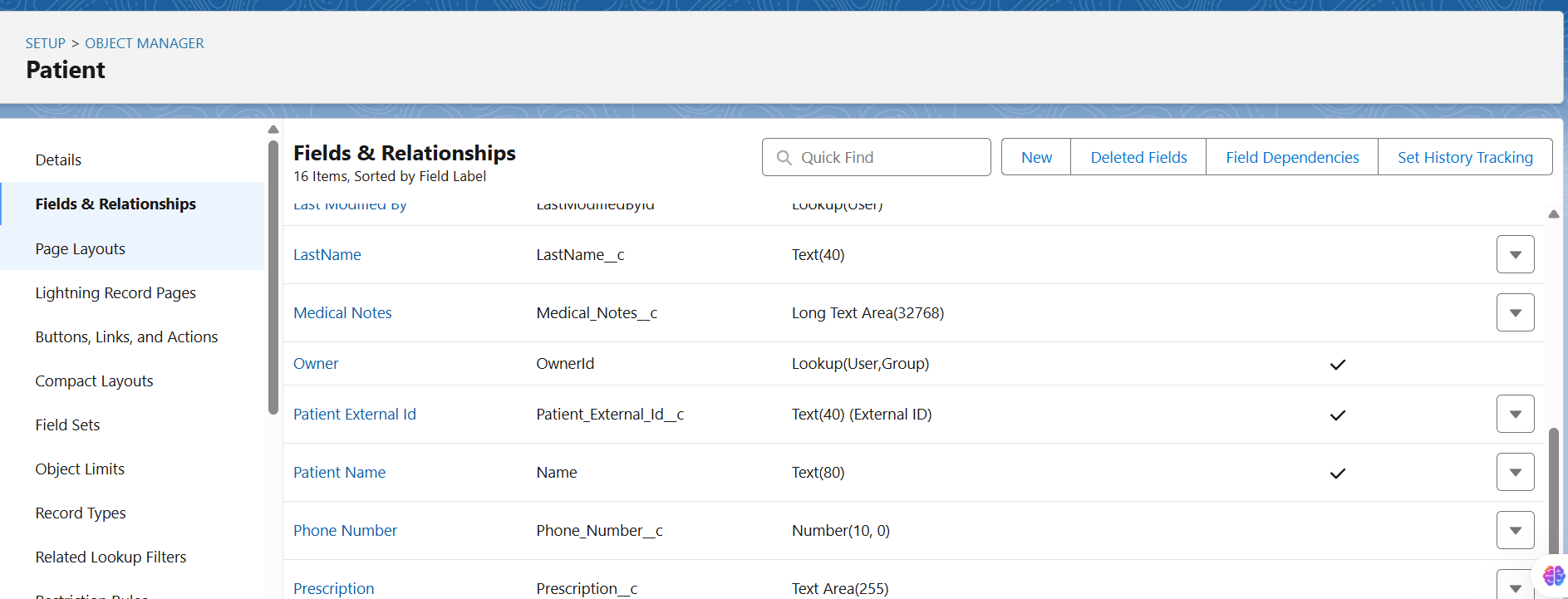
**Phase 3 :- Data Modeling & Relationships**

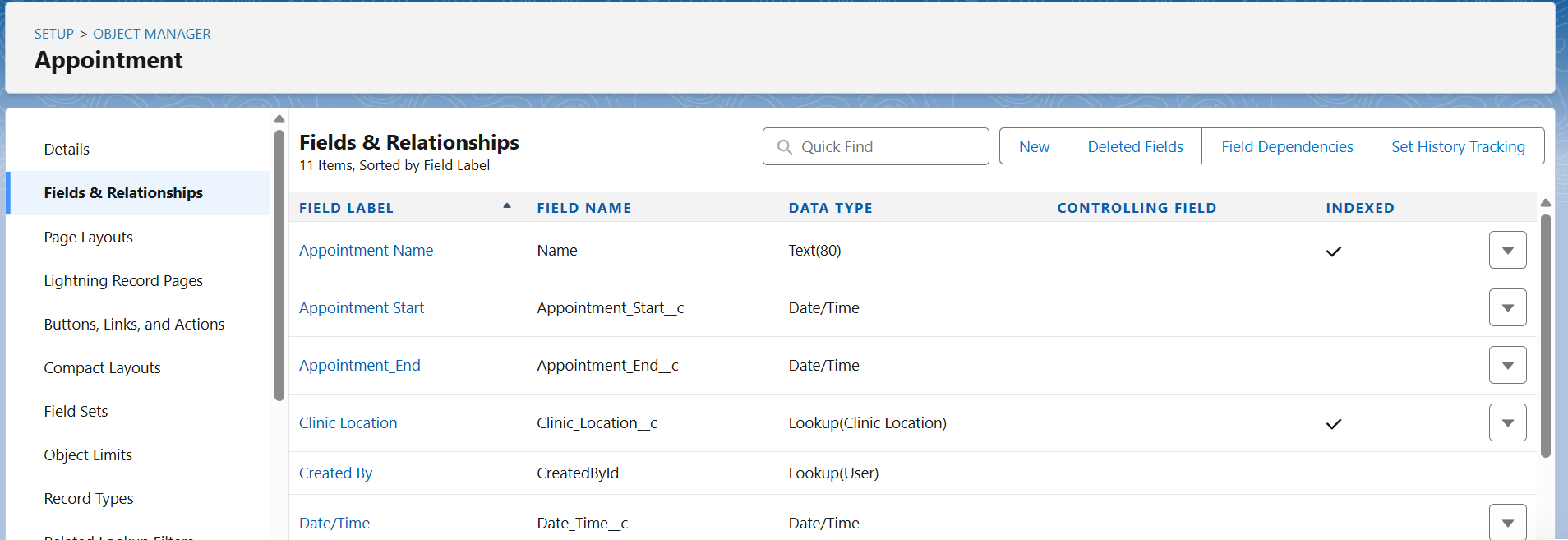
### 1. Standard & Custom Objects

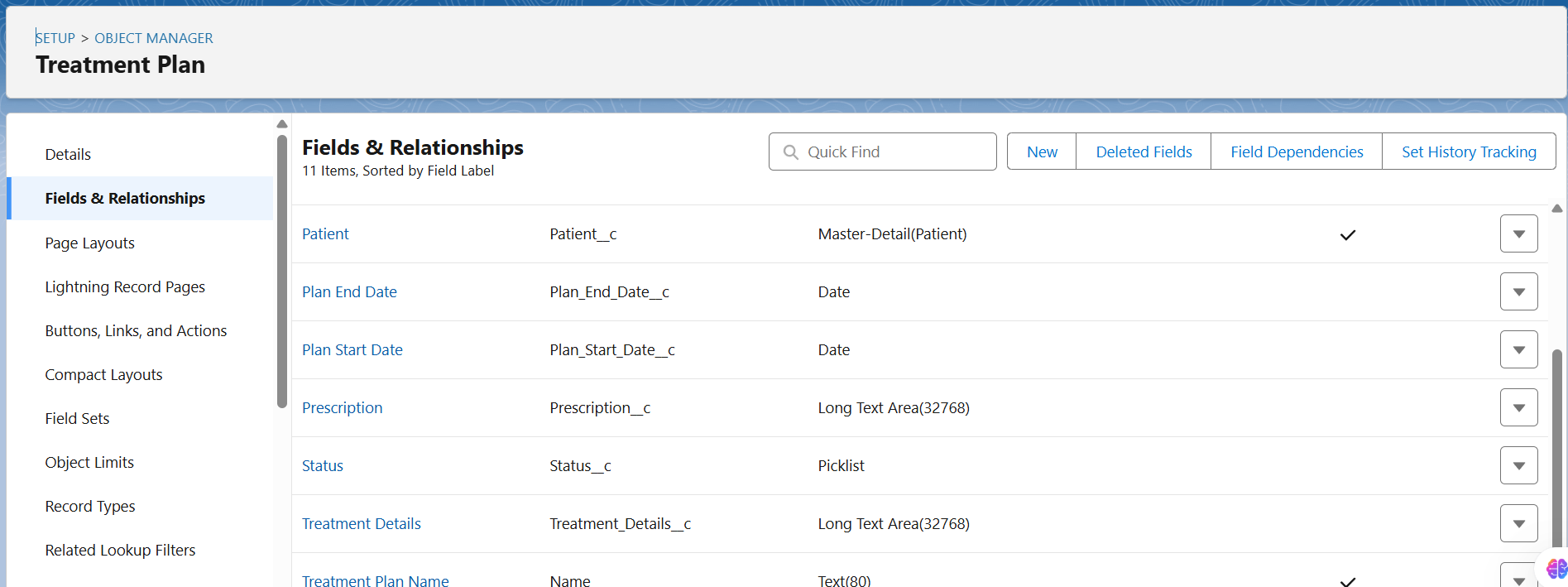
* Used standard objects: **User** (Doctors, Nurses, Admins).
* Created custom objects:
  + **Patient\_\_c** (master record for patients)
  + **Appointment\_\_c** (linked to Patient)
  + **Treatment\_Plan\_\_c** (linked to Patient)
  + **FollowUp\_Schedule\_\_c** (linked to Patient)
  + **Clinic\_Location\_\_c** (referenced from Appointment)

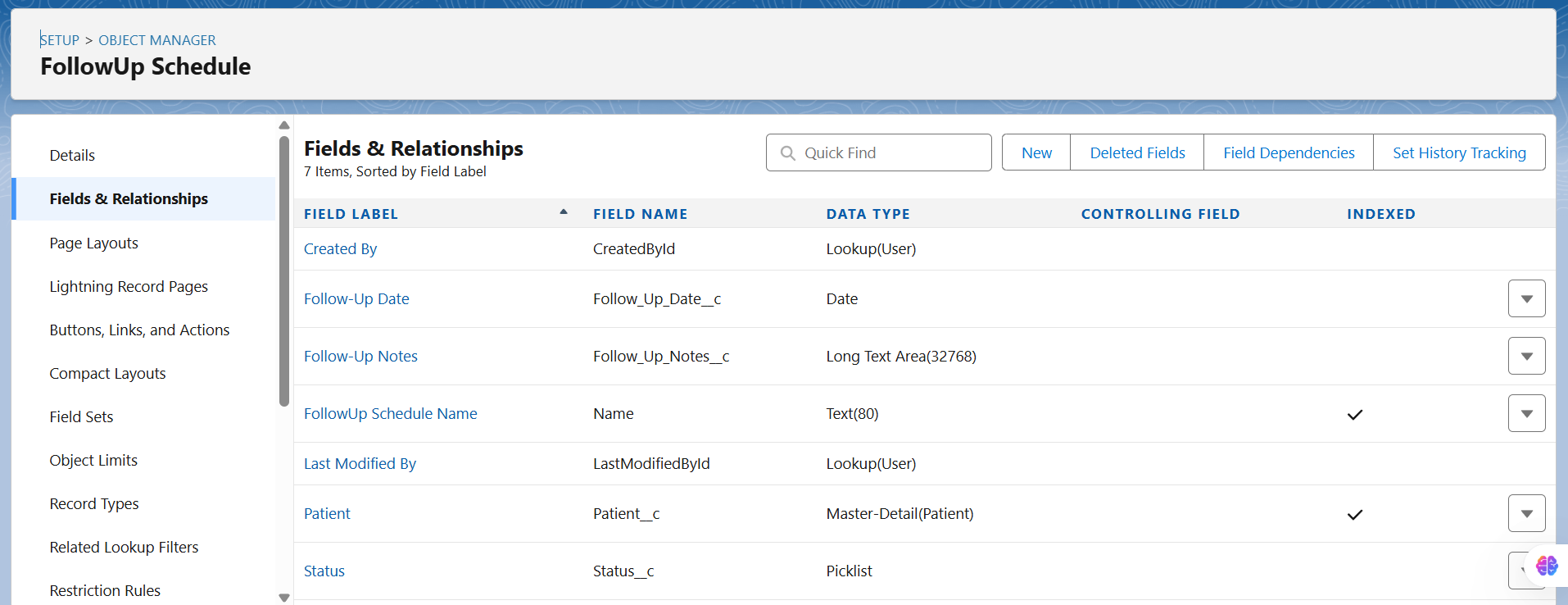
### 2. Fields

* Added key fields for each object:
  + Patient\_\_c → FirstName, LastName (required), Gender, DOB, Phone, Email, Assigned\_Doctor\_\_c (Lookup → User).
  + Appointment\_\_c → Appointment\_Date\_\_c, Status\_\_c (Scheduled/Completed/Missed), Clinic\_Location\_\_c (Lookup → Clinic\_Location\_\_c).
  + Treatment\_Plan\_\_c → Diagnosis\_\_c, Medication\_\_c, Next\_Review\_Date\_\_c.
  + FollowUp\_Schedule\_\_c → FollowUp\_Date\_\_c, Notes\_\_c.

****

****

****

****

### 3. Record Types

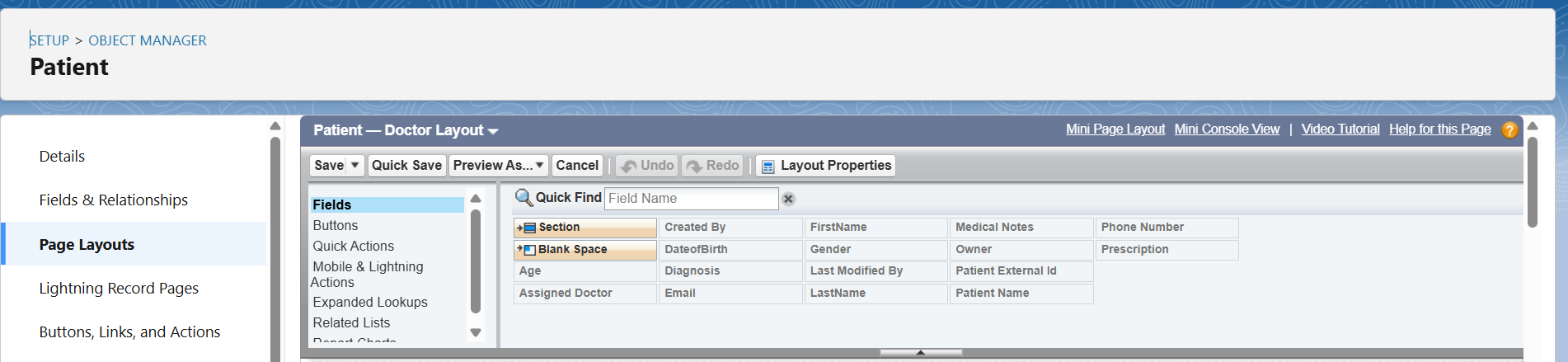
* Created Record Types for **Patient**:
  + Outpatient
  + Inpatient
* Assigned Record Types to profiles:
  + Doctor\_Profile → both Outpatient & Inpatient.
  + Receptionist\_Profile → Outpatient only.

### 

### 

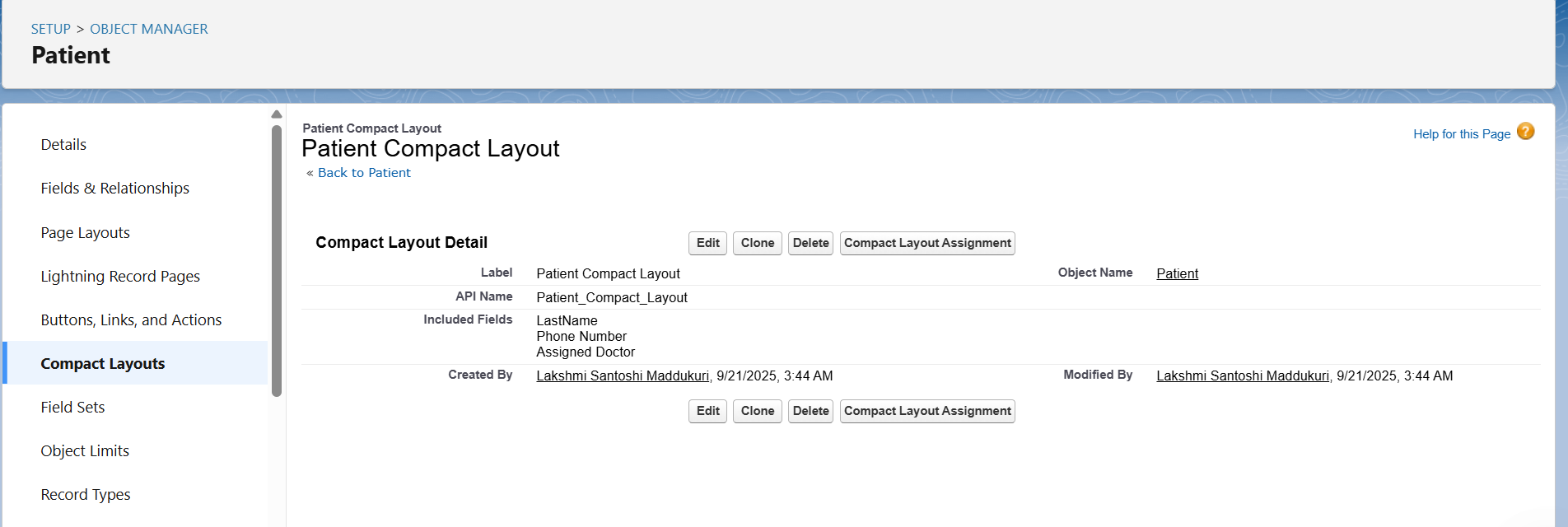
### 4. Page Layouts

* Created role-based layouts:
  + **Doctor Layout** → includes Medical Details, Appointments, Treatment Plans, FollowUp Schedules.
  + **Nurse Layout** → shows Medical Details but hides sensitive free text fields.
  + **Receptionist Layout** → only Personal Info + Appointments.

****

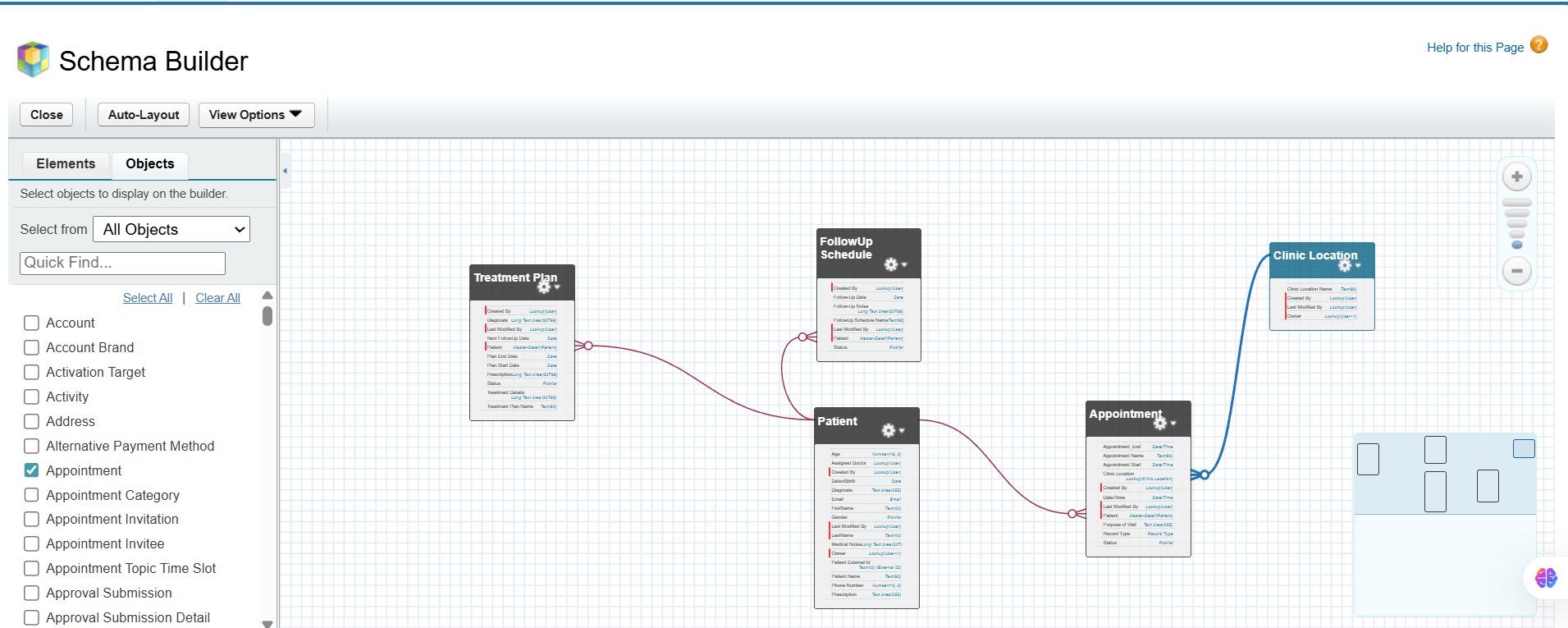
### 5. Compact Layouts

* Configured compact layout for **Patient**:
  + Fields: LastName, Phone, Assigned\_Doctor\_\_c, Next\_FollowUp\_Date\_\_c.
* Compact layout set as **Primary** for all profiles.

****

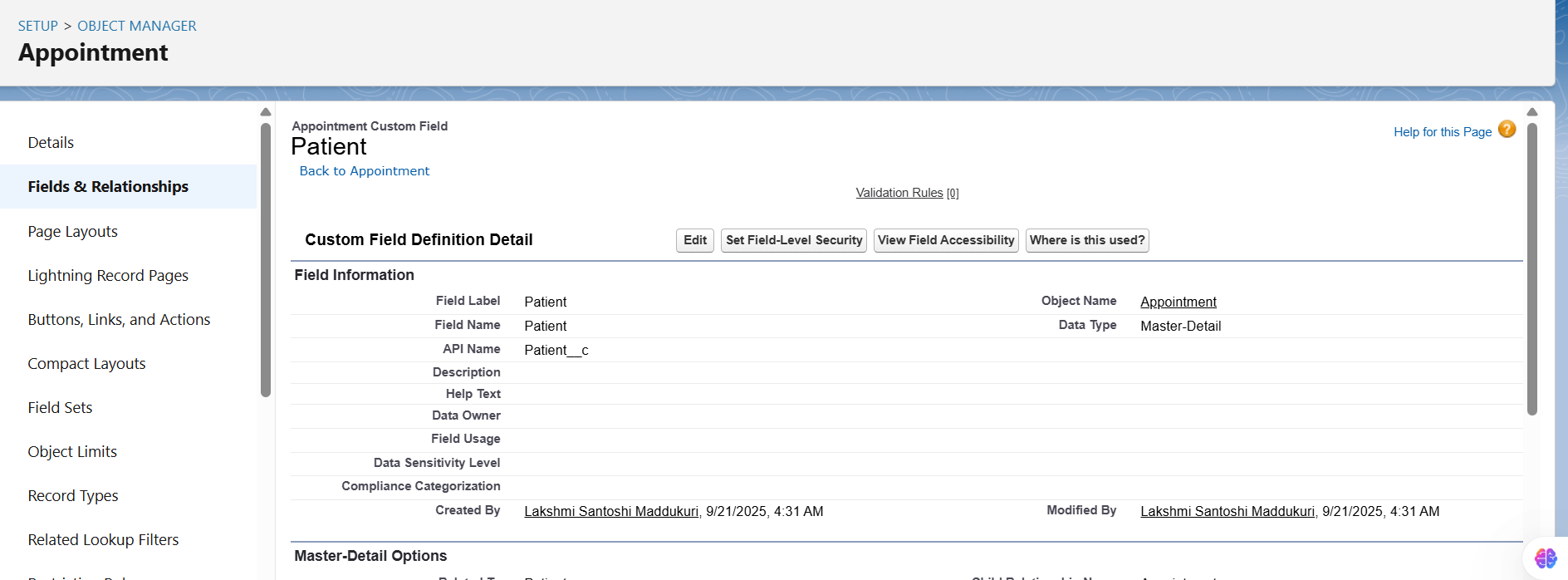
### 6. Schema Builder

* Used Schema Builder to visualize relationships:
  + Patient\_\_c (Master) → Appointment\_\_c, Treatment\_Plan\_\_c, FollowUp\_Schedule\_\_c (Detail).
  + Appointment\_\_c → Lookup → Clinic\_Location\_\_c.

****

### 7. Relationships (Lookup vs Master-Detail vs Hierarchical)

* **Master-Detail:** Patient\_\_c → Appointment\_\_c, Treatment\_Plan\_\_c, FollowUp\_Schedule\_\_c.
* **Lookup:** Appointment\_\_c → Clinic\_Location\_\_c.
* **Hierarchical:** Not used (only applies to User object).

****

## 8. Junction Objects

We created a **junction object** to handle many-to-many relationships.

**Use Case in Healthcare CRM:**

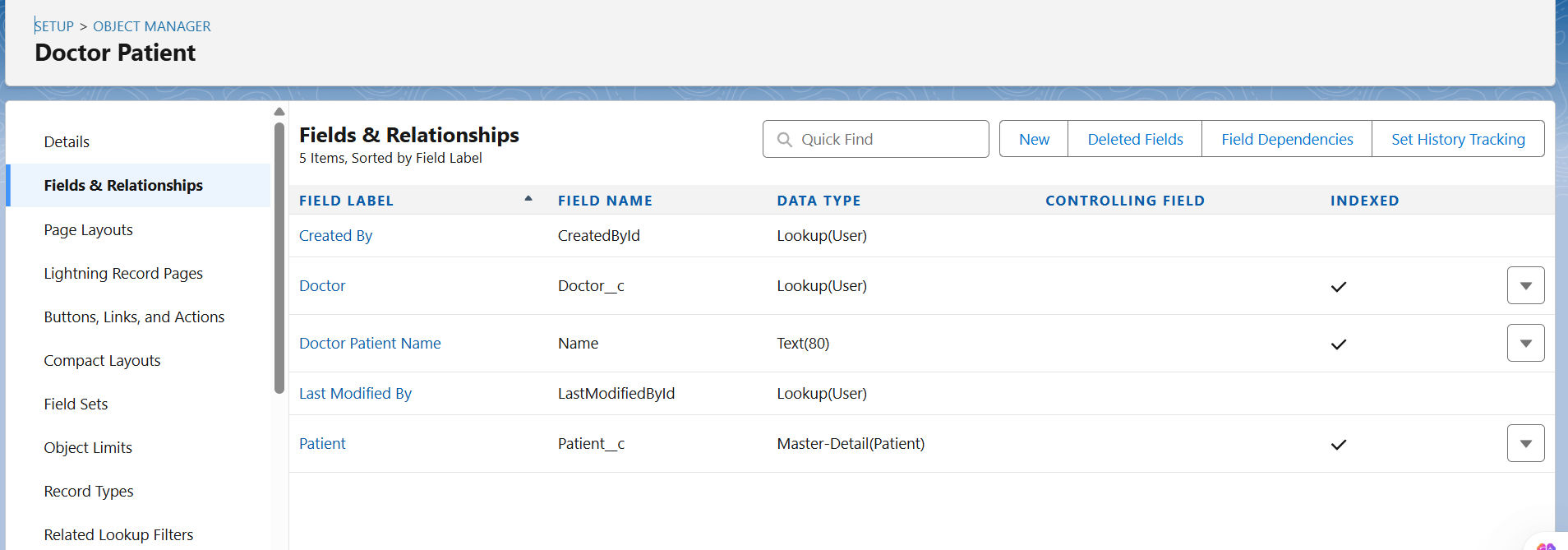
* A patient can be treated by multiple doctors (specialists).
* A doctor can treat multiple patients.  
  👉 To model this, we created a **Patient\_Doctor\_Assignment\_\_c** junction object.

**Configuration:**

* Custom Object: Patient\_Doctor\_Assignment\_\_c
* Fields:
  + **Patient\_\_c** (Master-Detail → Patient)
  + **Doctor\_\_c** (Master-Detail → User)

**Result:**

* Many-to-Many between Patient and Doctor via Patient\_Doctor\_Assignment.
* Used in reports/dashboards to see “which doctor is treating which patients.”



## 9. External Objects

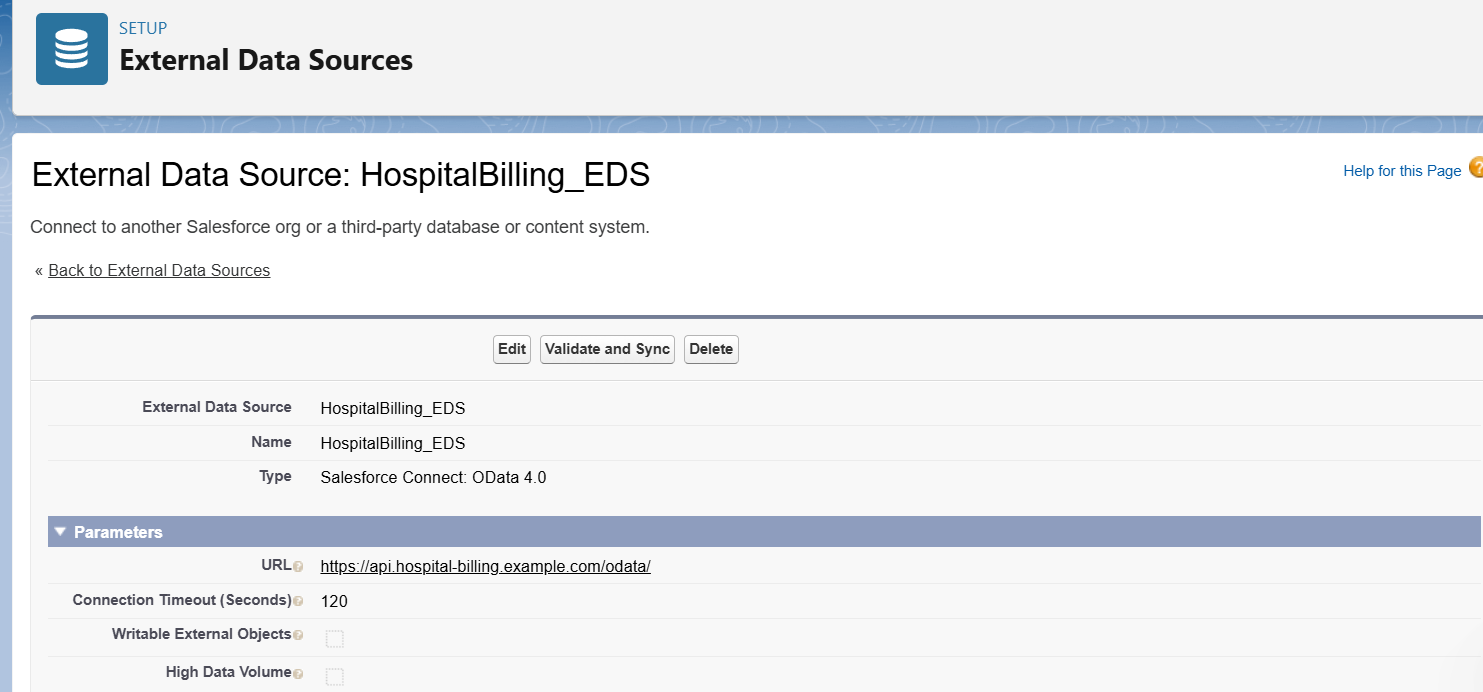
We configured an **External Data Source** to prepare for integration with hospital billing/lab systems.

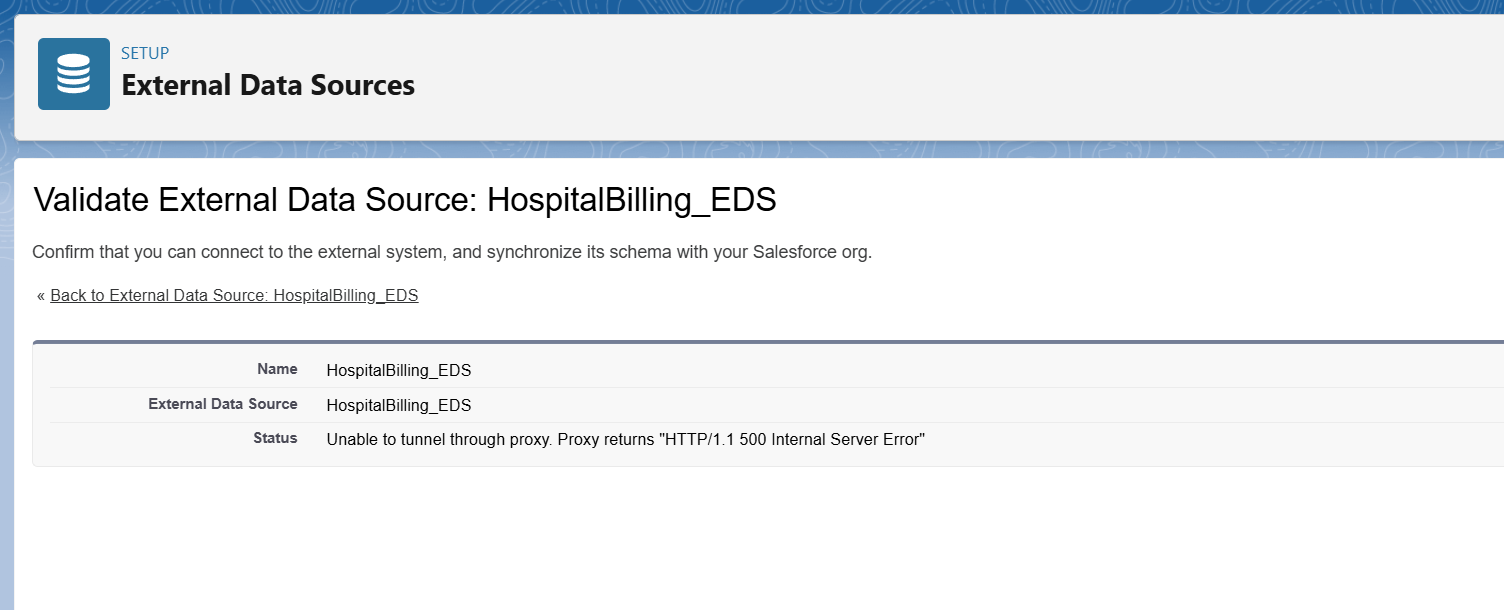
**Steps:**

1. Created Named Credential: HospitalBilling\_NC (No Authentication, placeholder).
2. Created External Data Source: HospitalBilling\_EDS (OData 4.0).
3. Attempted Validate & Sync → error due to Developer Org proxy.

**Result:**

* External Objects (Invoice\_\_x, Customer\_\_x) were planned but not synced in Dev Org.
* This step will be completed in **Phase 7 (Integrations)**.





# **Phase 4 — Process Automation (Admin)**

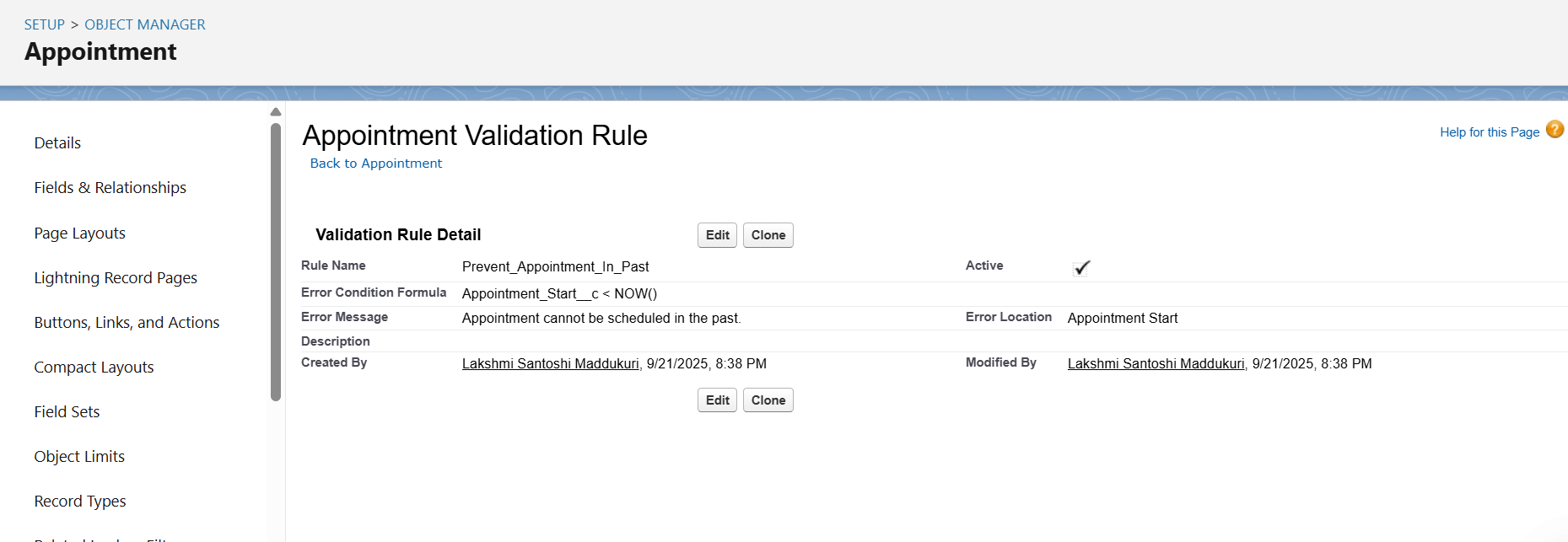
**Goal:** Automate validation, reminders, approvals and routine actions so patients don’t miss follow-ups and staff get notified automatically.

**1. Validation Rules**

**Purpose:** To enforce data quality by preventing invalid records.  
**Example:** Prevent users from creating appointments in the past.

**Steps:**

1. Setup → Object Manager → Appointment → Validation Rules → New.
2. Rule Name: Prevent\_Appointment\_In\_Past.
3. Formula:
4. Appointment\_Start\_\_c < NOW()
5. Error Message: *Appointment cannot be scheduled in the past.*

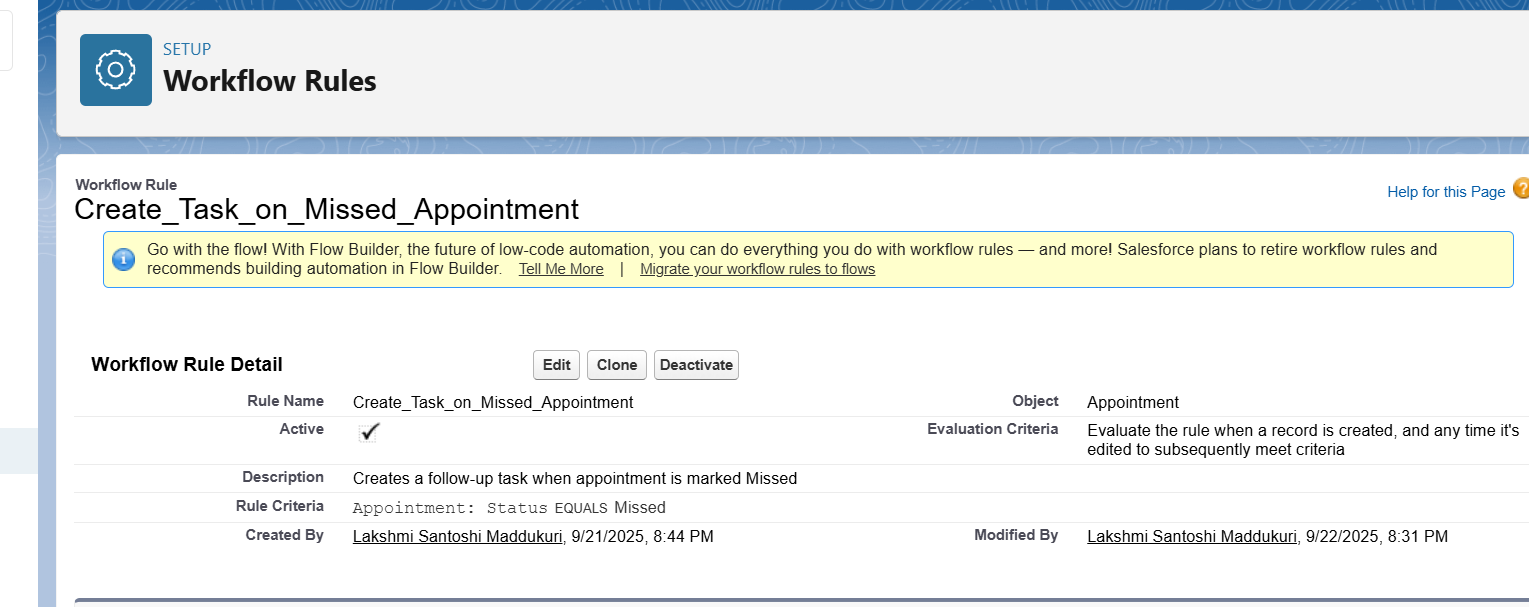
**

## ****2. Workflow Rules (Legacy)****

**Purpose:** Legacy automation (not recommended for new builds, but demonstrated).  
**Example:** When an appointment is marked "Missed", create a follow-up Task.

**Steps:**

1. Setup → Workflow Rules → New Rule → Object = Appointment.
2. Criteria: Appointment\_Status\_\_c = 'Missed'.
3. Immediate Action → New Task.
   * Subject: Follow up missed appointment
   * Assigned To: User/Queue
   * Priority: High
4. Save & Activate.

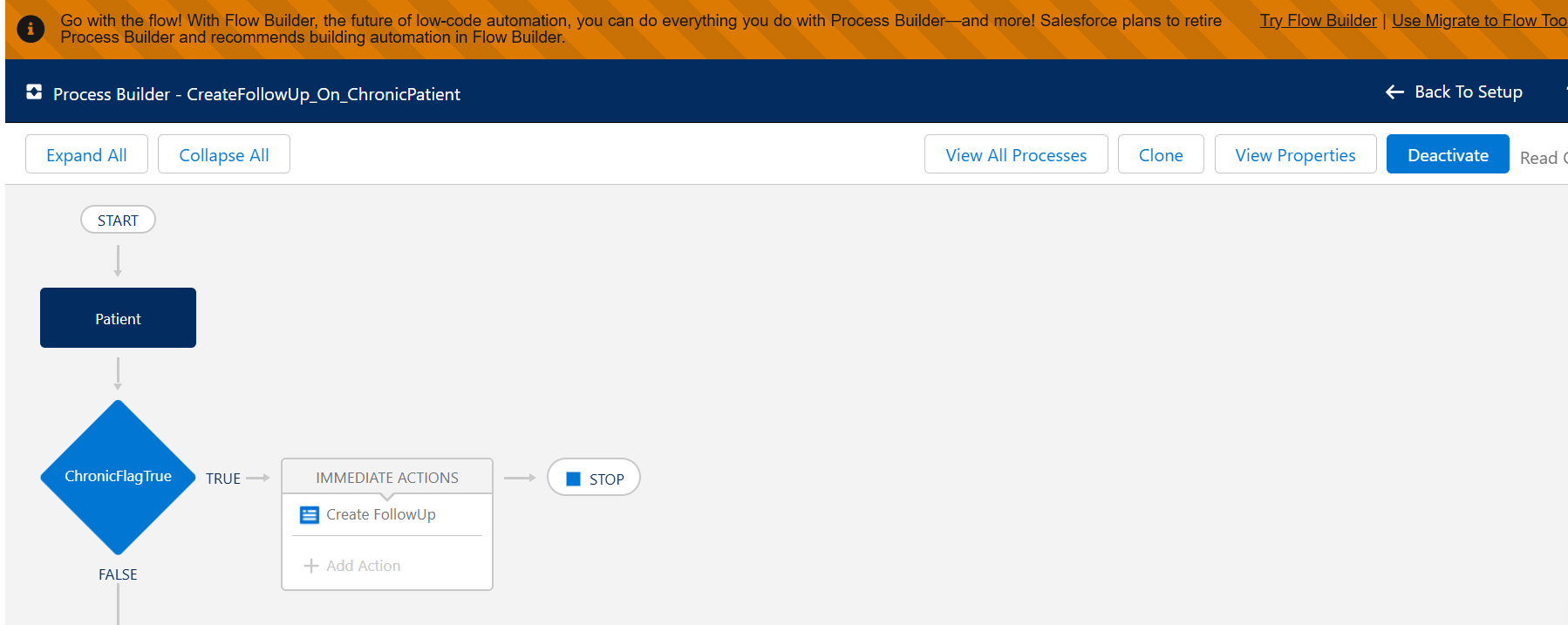


## ****3.Process Builder (Legacy)****

**Purpose:** Automate creation of follow-up schedule for patients with chronic disease.

**Steps:**

1. Setup → Process Builder → New.
2. Object: Patient, Start when record is created/edited.
3. Criteria: Chronic\_Disease\_\_c = TRUE.
4. Immediate Action → Create Record → FollowUp\_Schedule\_\_c.
   * Patient\_\_c = [Patient].Id
   * Scheduled\_Date\_\_c = Today + 30.
5. Save & Activate.

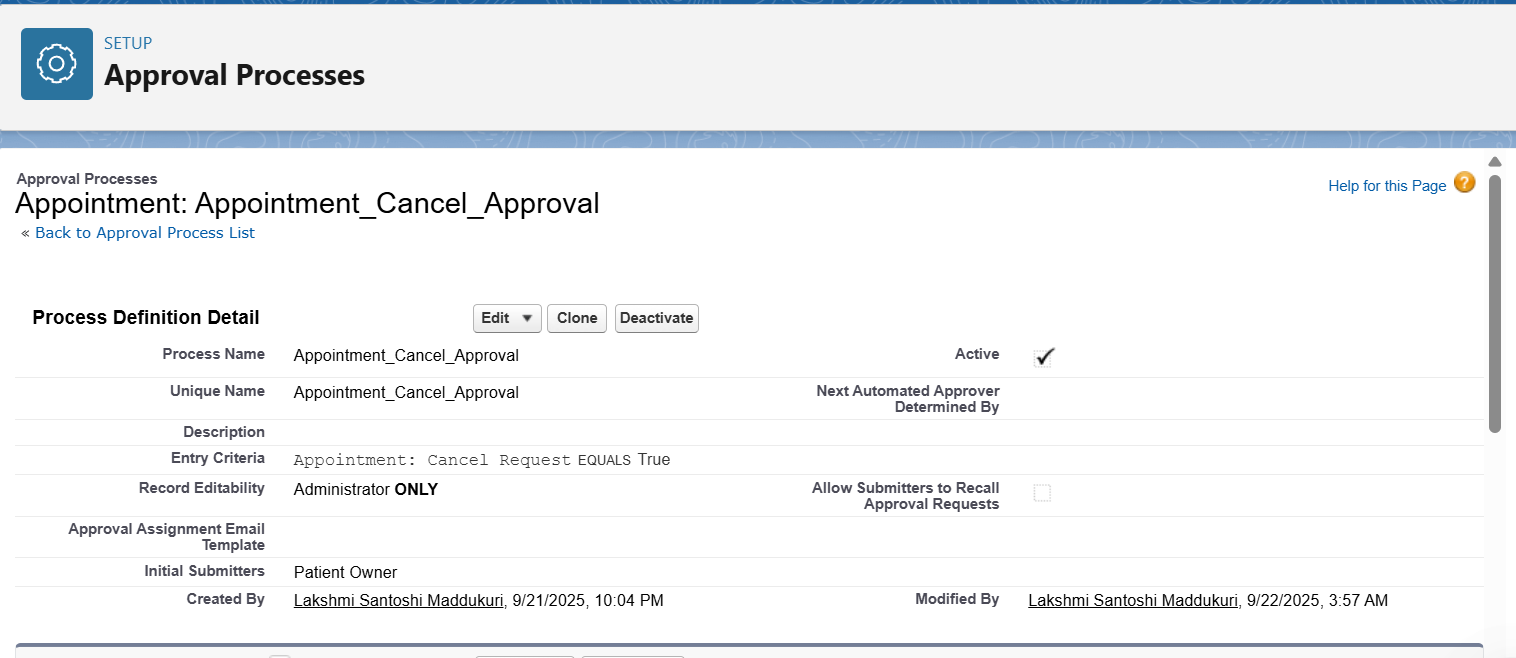


## ****4. Approval Process****

**Purpose:** Require approval before cancelling an appointment.

**Steps:**

1. Setup → Approval Processes → Object = Appointment.
2. Name: Appointment\_Cancel\_Approval.
3. Entry Criteria: Cancel\_Request\_\_c = TRUE.
4. Approver: Hospital Admin (or assigned role).
5. Final Approval Action:
   * Field Update → Appointment\_Status\_\_c = Cancelled.
   * Email Alert → Patient.
6. Final Rejection Action: Reset Cancel\_Request\_\_c = FALSE.
7. Add **Submit for Approval** to Appointment Page Layout.
8. Save & Activate.

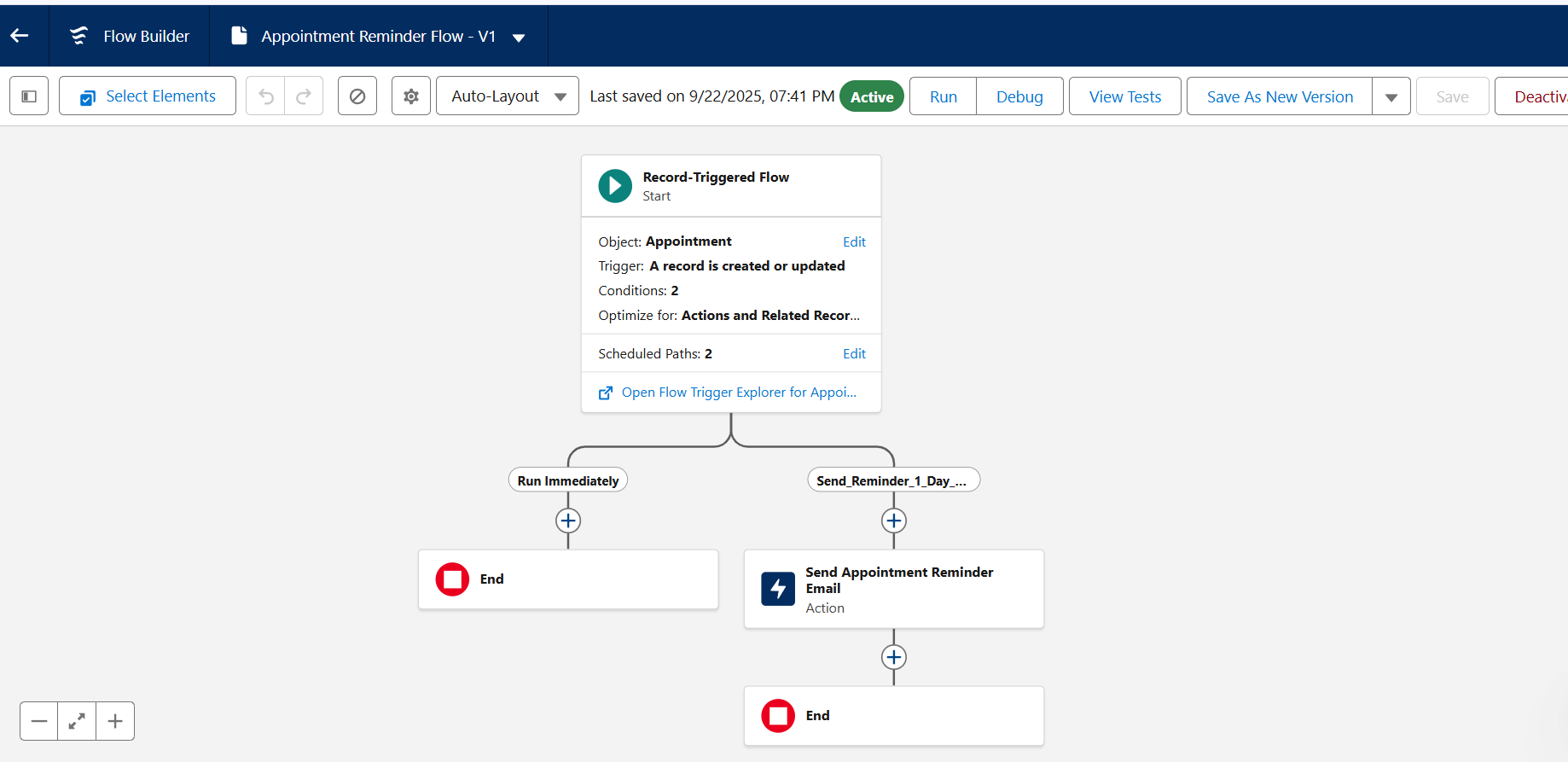


**5. Flow Builder**

Flows are the recommended automation tool. We built **four types** of flows.

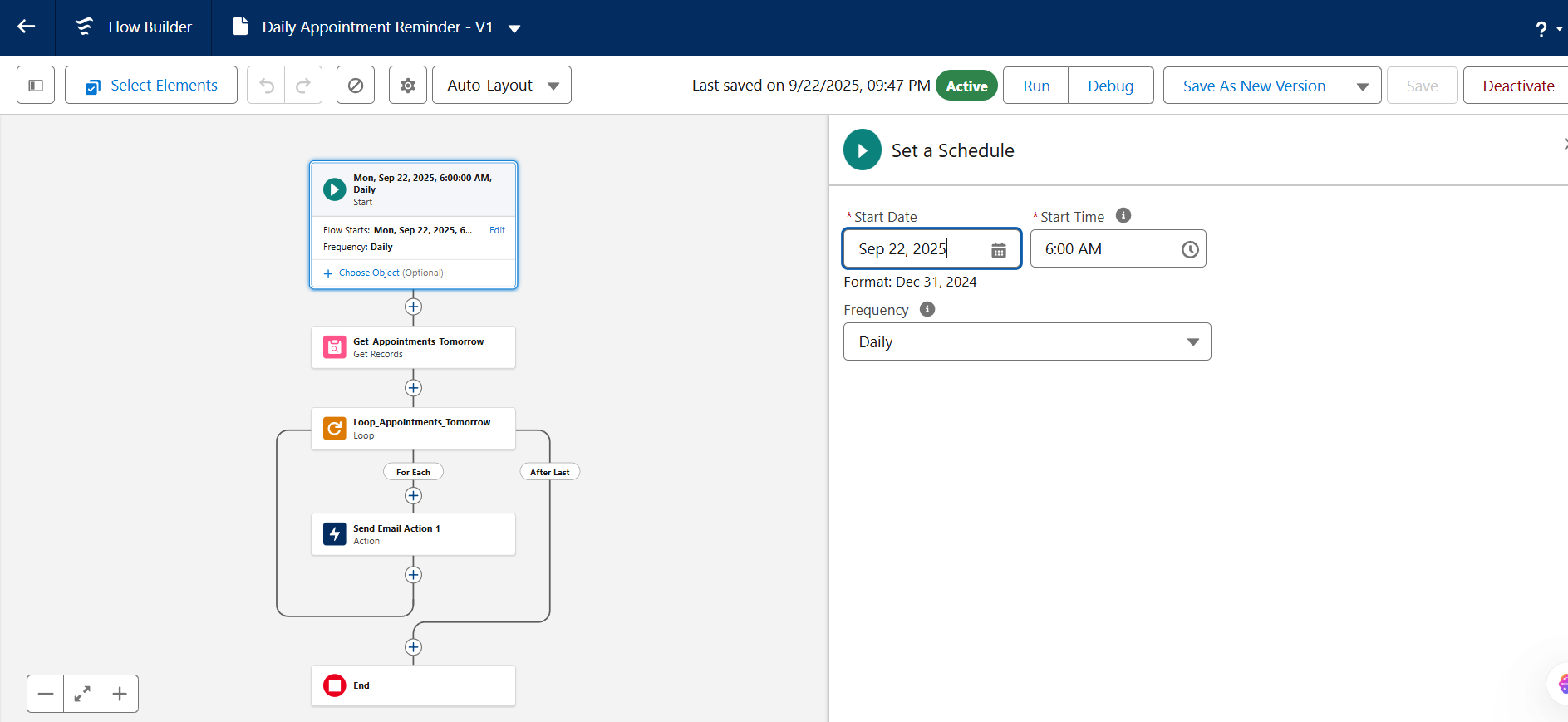
**A) Record-Triggered Flow — Appointment Reminder (24 hours before)**

1. Flow → New → Record-Triggered Flow.
2. Object = Appointment.
3. Entry Conditions:
   * Appointment\_Status\_\_c = Scheduled
   * Appointment\_Start\_\_c != null
4. Add Scheduled Path → Offset = -1 Day before Appointment\_Start\_\_c.
5. Action: Send Email Alert (Appointment\_Reminder\_Template).
6. Save & Activate.



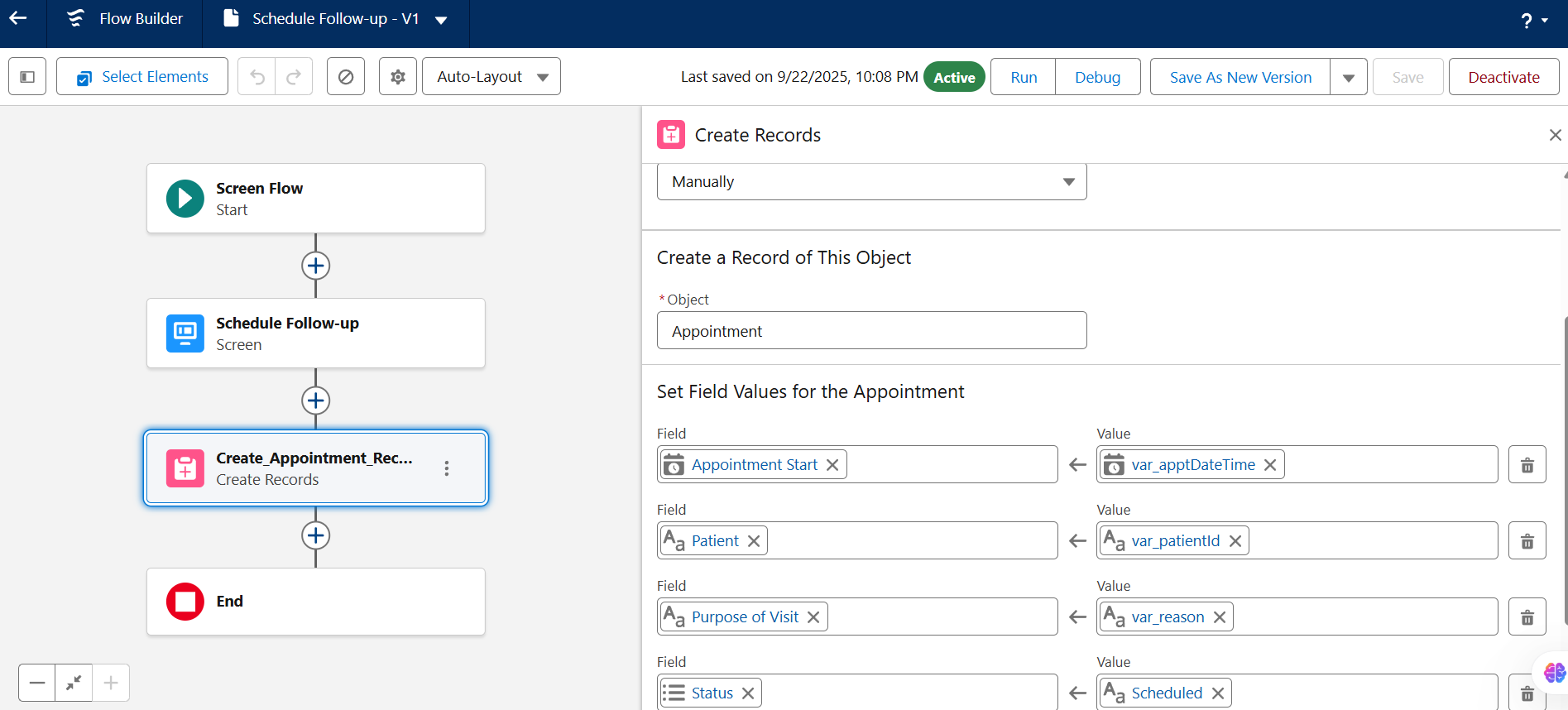
**B) Schedule-Triggered Flow — Daily Batch Reminders**

1. Flow → New → Schedule-Triggered Flow.
2. Schedule: Daily at 6:00 AM.
3. Get Records: Appointments where Appointment\_Start\_\_c = Tomorrow and Status = Scheduled.
4. Loop through records.
5. Inside loop: Send Email / Create Task / Custom Notification.
6. Save & Activate.



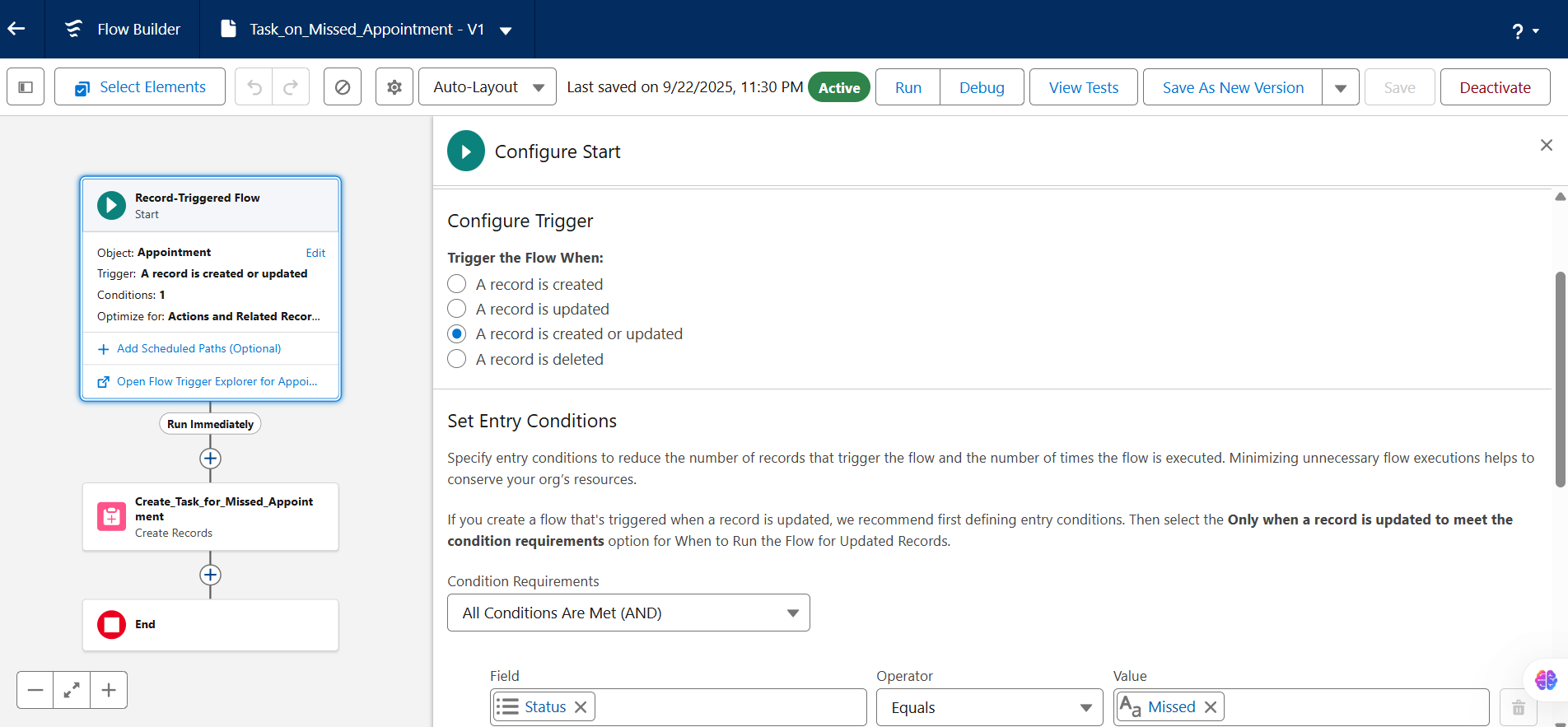
**C) Screen Flow — Receptionist Quick Schedule**

1. Flow → New → Screen Flow.
2. Add Screen:
   * Record Choice Set → Patient.
   * Date/Time Picker → Appointment Date.
   * Text → Reason.
3. Create Records: Appointment.
   * Patient\_\_c = {!PatientChoice}
   * Appointment\_Start\_\_c = {!apptDateTime}
   * Status = Scheduled.
4. Save & Activate.
5. Add as Quick Action to Patient record page.



**D) Auto-Launched Flow — Reusable Logic**

1. Flow → New → Auto-Launched Flow.
2. Variable: patientId (Text, Available for Input).
3. Formula: Follow\_Up\_Date = TODAY() + 30.
4. Create Records: FollowUp\_Schedule\_\_c.
   * Patient\_\_c = {!patientId}
   * Follow\_Up\_Date\_\_c = {!Follow\_Up\_Date}.
5. Save & Activate.



**6. Email Alerts**

**Purpose:** To notify patients of upcoming appointments.

**Steps:**

1. Setup → Email Templates → New → Lightning Email Template.
   * Related Entity = Appointment.
   * Subject: Reminder: Appointment on {!Appointment.Appointment\_Start\_\_c}.
   * Body: Hello {!Appointment.Patient\_\_r.Name}, your appointment is scheduled for {!Appointment.Appointment\_Start\_\_c}.
2. Setup → Email Alerts → New.
   * Object: Appointment.
   * Email Template: Appointment\_Reminder\_Template.
   * Recipient: Patient.Email.
3. Save.



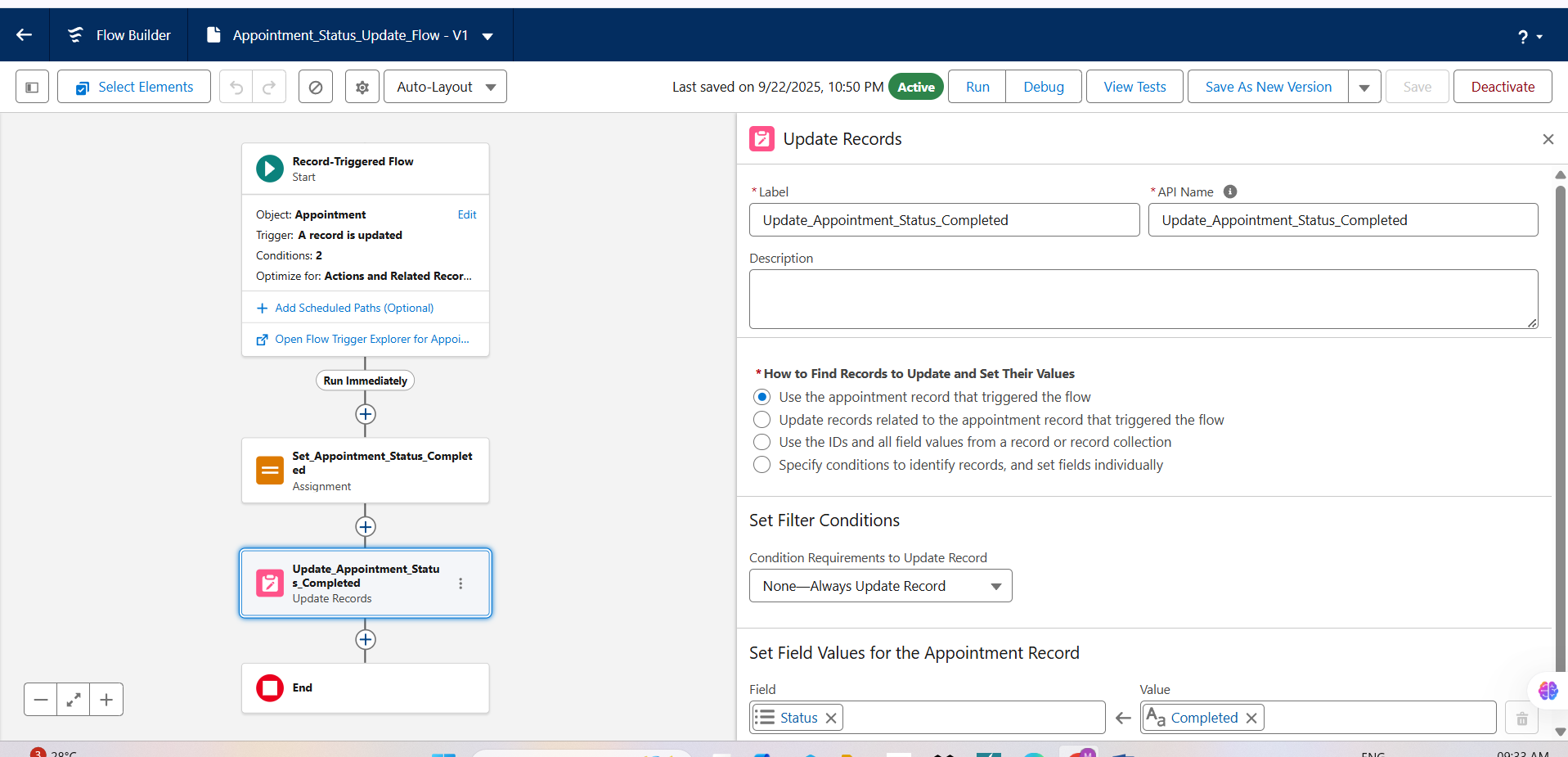
****

**7. Field Updates**

**Purpose:** Mark appointments as completed when follow-up is done.

**Steps:**

1. Flow → New → Record-Triggered Flow.
2. Object: Appointment.
3. Entry Condition: FollowUp\_Completed\_\_c = TRUE.
4. Action: Update Records (or Assignment in before-save).
   * Set Appointment\_Status\_\_c = Completed.
5. Save & Activate.

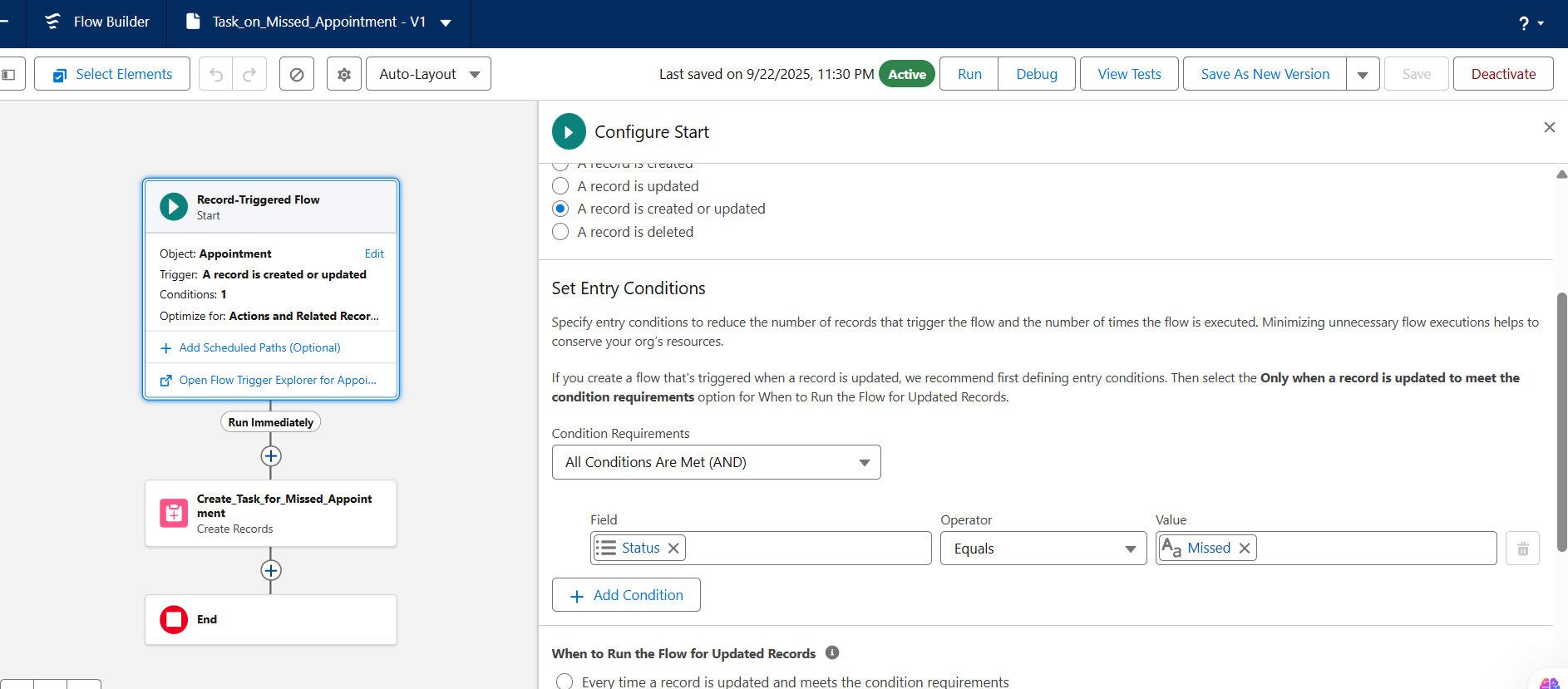


**8. Tasks**

**Purpose:** Create Tasks when appointments are missed.

**Steps:**

1. Flow → New → Record-Triggered Flow.
2. Object: Appointment.
3. Condition: Appointment\_Status\_\_c = Missed.
4. Action: Create Task.
   * Subject: Follow up missed appointment for {!$Record.Patient\_\_r.Name}.
   * Priority: High.
   * Status: Not Started.
   * Due Date: Today + 1.
   * WhatId = Patient\_\_c.
   * OwnerId = Receptionist/Owner.
5. Save & Activate.



**9. Custom Notifications**

**Purpose:** Send in-app/mobile reminders.

**Steps:**

1. Setup → Notification Builder → Custom Notifications → New.
   * Name: Appointment\_Reminder.
   * Channels: Desktop & Mobile.
2. Flow Action → Send Custom Notification.
   * Type: Appointment\_Reminder.
   * Title: Appointment Reminder.
   * Body: Appointment tomorrow at {!$Record.Appointment\_Start\_\_c}.
   * Recipient: User Id (admin or assigned).
   * Target Record Id = Appointment Id.
3. Save & Activate.

