MediAid Healthcare Project

Project: MediAid – Smart Medicine Reminder App

Phase 1: Problem Understanding & Industry Analysis

1. Problem Statement

Medication adherence is a major challenge, especially for **elderly patients, individuals with chronic illnesses, and patients on multiple prescriptions**. Missing doses can reduce treatment effectiveness and lead to serious health complications.

Currently, patients rely on **manual methods** such as alarms, paper notes, or memory, which often result in:

- Missed or incorrect medicine intake
- Lack of monitoring by caregivers or doctors
- No central tracking of medicine history
- Difficulty managing multiple prescriptions

Need: A simple, user-friendly digital solution to **remind patients about medicines, notify caregivers, and maintain a history of adherence**.

2. Objectives

The MediAid system aims to:

1. Provide **automated reminders** for medicine intake.
2. Enable **prescription entry** (manual upload or doctor's input).
3. **Notify caregivers/family members** if doses are missed.
4. Generate **reports for doctors** on patient adherence.
5. Support **multi-language UI** for accessibility.

3. Stakeholder Analysis

Stakeholder	Role in System
Patient	Adds prescriptions, receives reminders, tracks intake
Caregiver/Family	Monitors patient adherence, gets alerts for missed doses

Stakeholder	Role in System
Doctor	Uploads prescriptions, reviews adherence reports
Pharmacy	Provides refill alerts and medicine stock info
Admin	Manages users, system roles, and settings

4. Business Process Mapping

Current Process (Manual):

1. Doctor prescribes medicines → patient writes them down or relies on memory.
2. Patient uses alarms or manual tracking.
3. Caregivers have no visibility on patient adherence.

Proposed Process (MediAid):

1. Patient or doctor enters prescription into the MediAid app.
2. App generates a **medicine reminder schedule**.
3. Patient receives **timely reminders** via mobile notifications or SMS.
4. Caregiver is notified if the patient **misses a dose**.
5. Reports are stored and shared with **doctors and family members**.

5. Industry-Specific Use Case Analysis

Healthcare Industry Challenge:
Medication non-adherence leads to **treatment failures, hospital readmissions, and increased healthcare costs**.

Use Case Example:

- A diabetic patient enters their prescription into MediAid.
- App sends reminders at **scheduled times**.
- If a dose is missed, the **caregiver is alerted immediately**.
- Doctor reviews **weekly adherence reports** and adjusts treatment accordingly.

6. AppExchange Exploration

- Reviewed Salesforce AppExchange to identify similar apps.

- **Findings:**
    - Multiple healthcare apps exist (patient engagement, appointment scheduling, hospital management).
    - Few apps focus on **personalized medicine reminders** with **real-time caregiver alerts**.
    - Most existing apps are generic healthcare management solutions.

Conclusion:
MediAid can **fill this market gap** by focusing on **medicine adherence, family involvement, and simple usability**, making it unique and highly valuable.

7. Expected Outcomes

- Improved **medicine adherence** among patients.
- Reduced chances of **missed or incorrect doses**.
- Better **caregiver and doctor monitoring**.
- Enhanced **patient safety and treatment effectiveness**.
- Contribution to **digital health adoption**.

Phase 2: Org Setup & Configuration

1. Salesforce Editions

- Use **Developer Edition (Free)** for implementation.
- **Enterprise Edition** for real-time projects (advanced security, APIs, integrations).
- For MediAid: **Developer Org** is enough since you'll build, test, and demo.

2. Company Profile Setup

- Company Name: **MediAid Healthcare Pvt Ltd**
- Default Locale: **English (India)**
- Time Zone: **Asia/Kolkata**
- Currency: **INR**

3. Business Hours & Holidays

- **Business Hours:** 9:00 AM – 9:00 PM (All days).
- **Holidays:** Diwali, Christmas, New Year.
- Useful for **scheduling reminder notifications** and **support cases**.

4. Fiscal Year Settings

- Standard Fiscal Year: **Jan–Dec** (default).
- Extended Fiscal Year: **Apr–Mar** (India standard).
- Helps analyze **medicine reminder usage reports quarterly/annually**.

5. User Setup & Licenses

- **Doctor** – Standard User license.
- **Patient** – Custom Community User.
- **Pharmacist** – Standard User license.
- **Admin** – System Administrator.

6. Profiles

- **Doctor Profile** → Access patient records, prescriptions.
- **Patient Profile** → Limited to own reminders, history.
- **Pharmacist Profile** → Access inventory, prescriptions.
- **Admin Profile** → Full access.

7. Roles

- Role Hierarchy:
    - Admin
        - Doctor
            - Pharmacist
                - Patient
- Enables **record visibility** (e.g., doctor can see patient reminders).

8. Permission Sets

- **Reminder Notifications Manager** → Allows creating/editing reminders.
- **Pharmacy Integration Access** → Allows connecting with external APIs.

9. OWD (Organization-Wide Defaults)

- **Patient Data** → Private.
- **Prescription Records** → Controlled by Parent (Patient).
- **Reminder Logs** → Private.

10. Sharing Rules

- Doctors can view patient reminders assigned to them.
- Pharmacists can see prescriptions created by doctors.

11. Login Access Policies

- Enable **"Administrators Can Log in as Any User"** for troubleshooting.

- Set login hours for pharmacists (e.g., **8 AM–10 PM**).

12. Dev Org Setup

- Use **Salesforce Developer Org** as primary workspace.
- Install **Salesforce Extensions for VS Code + SFDX CLI**.
- Connect **Dev Org with VS Code**.

13. Sandbox Usage

- **Developer Sandbox** → for testing new features.
- For MediAid demo: **Developer Org is enough**.

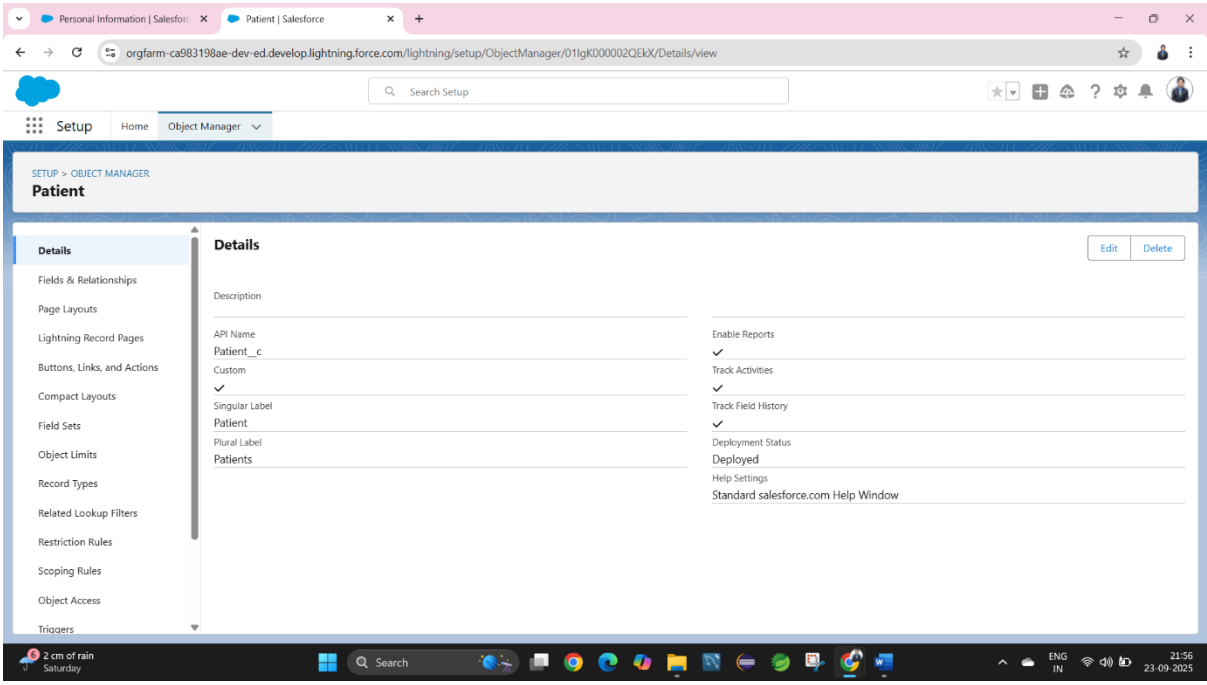
14. Deployment Basics

- Use **Change Sets** (Sandbox → Production) in real projects.
- For Dev Org project: use **SFDX CLI or GitHub** for version control.

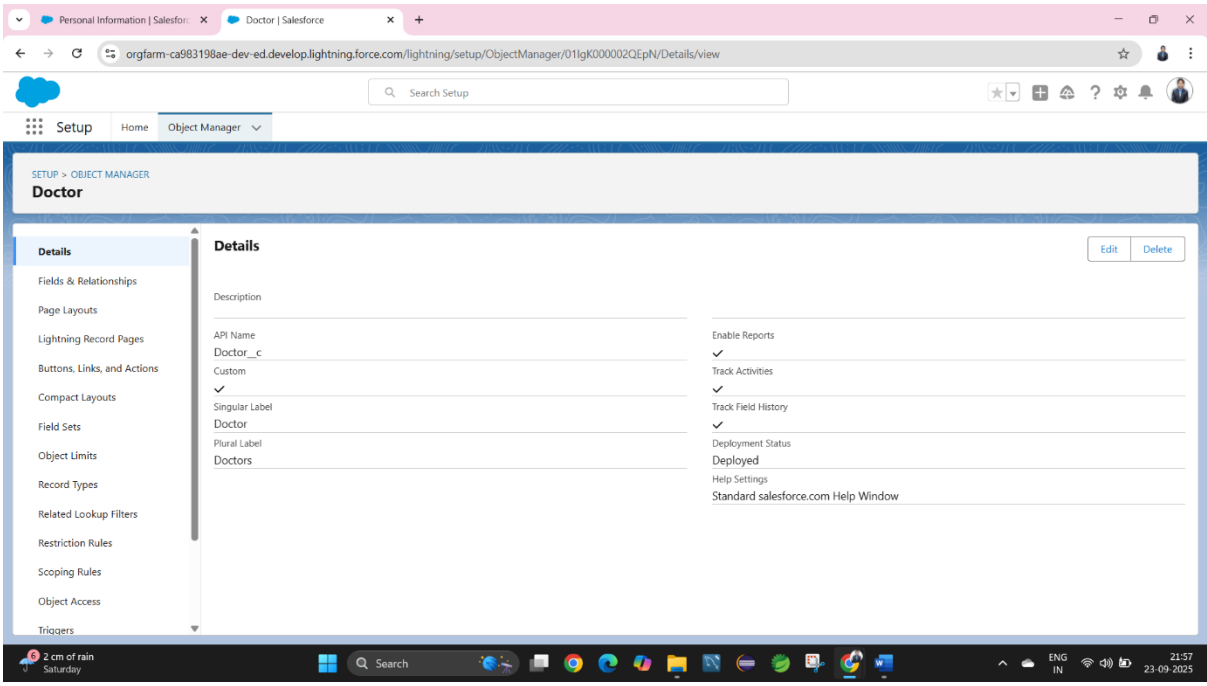
Phase 3: Data Modeling & Relationships

1. Standard & Custom Objects

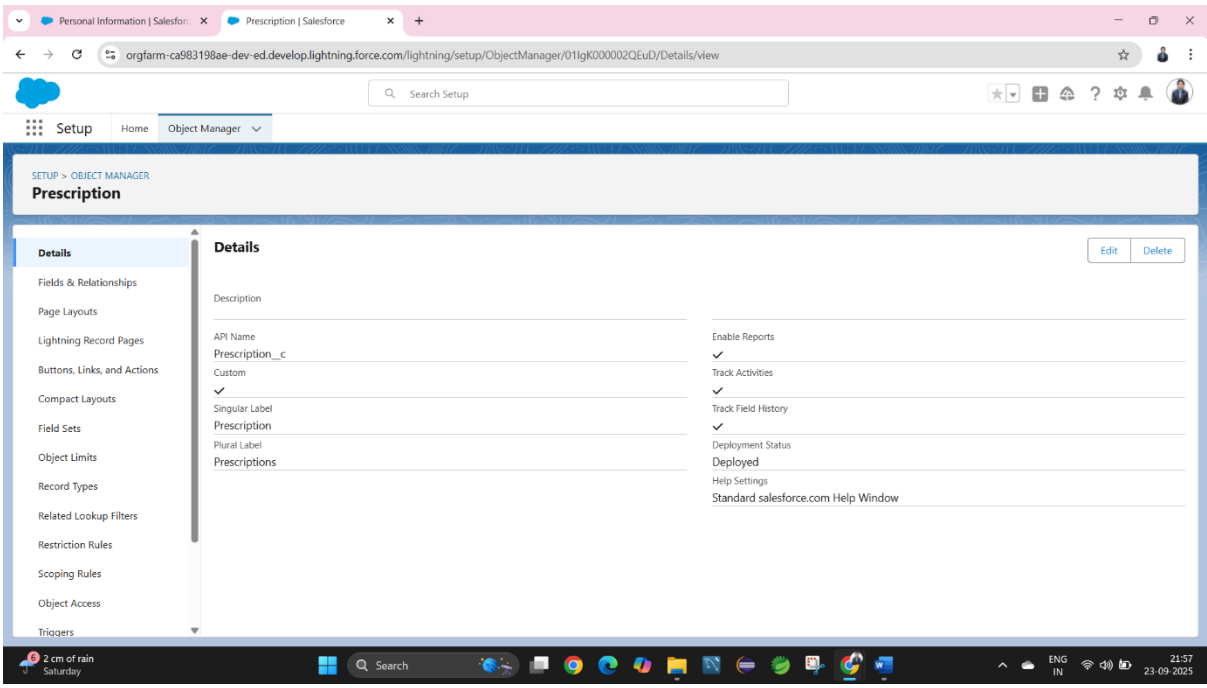
- **Standard Objects** (provided by Salesforce):
  - **User** → Doctors, Nurses, Patients, Admins.
  - **Account** → Hospitals or Clinics.
  - **Contact** → Individual patients or caregivers.
  - **Case** → Patient Issues, Complaints, or Medical Support tickets.
- **Custom Objects** (specific to MediAid):
  - **Patient__c** → Stores patient details (DOB, Gender, Contact , Blood Group).



  - **Doctor__c** → Stores doctor details (Specialization, Availability, Qualification).



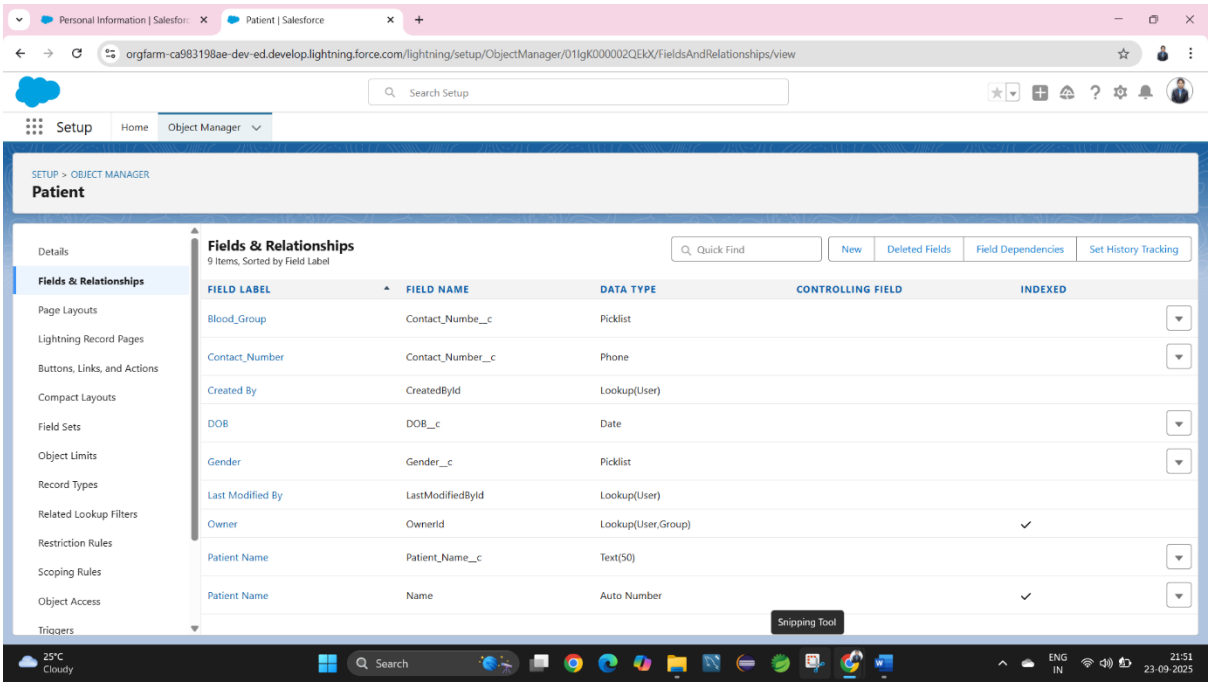
- **Appointment__c** → Links Patients & Doctors, tracks date/time/status.
- **Prescription__c** → Stores prescribed medicines, dosage, and duration.



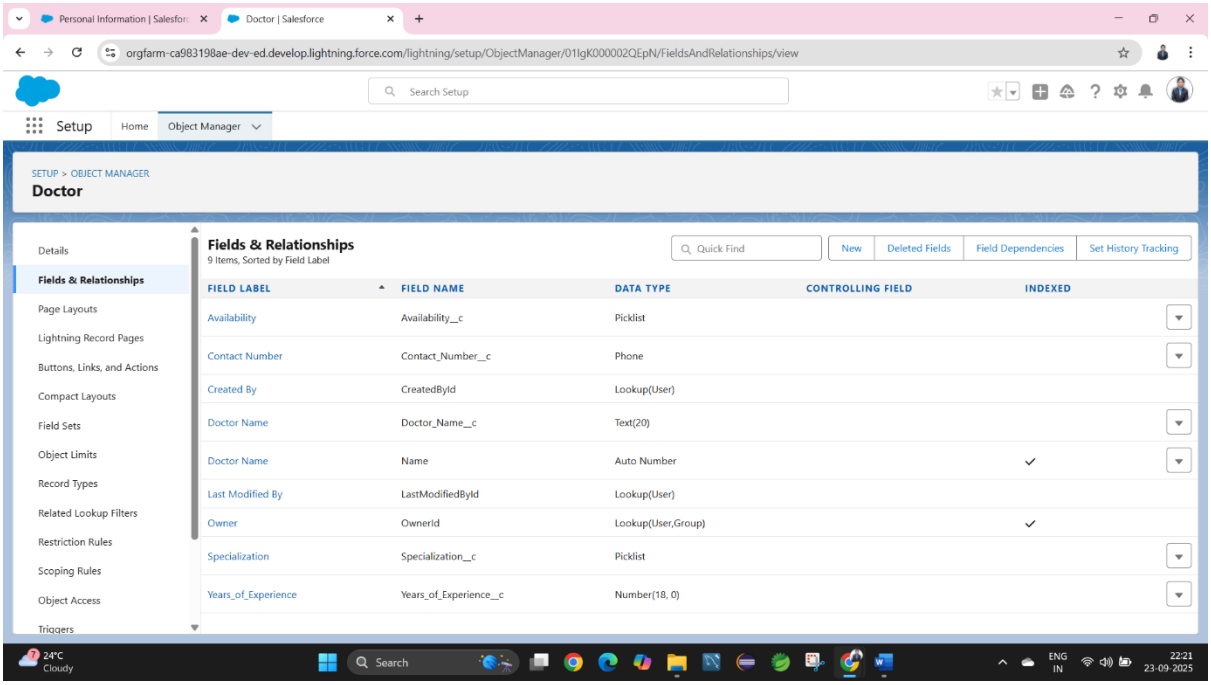
2. Fields

Each object will have **standard fields** (Name, CreatedDate, etc.) and **custom fields**.
Examples:

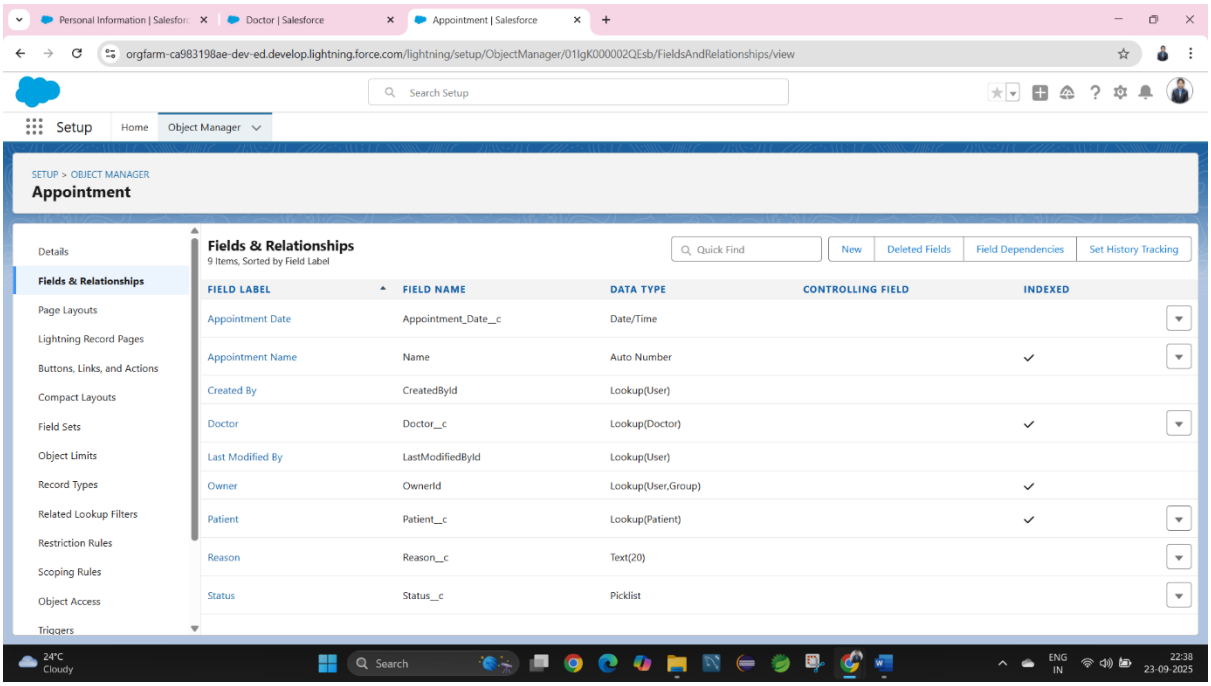
- **Patient__c** → DOB, Gender, Contact Number. Blood Group,etc



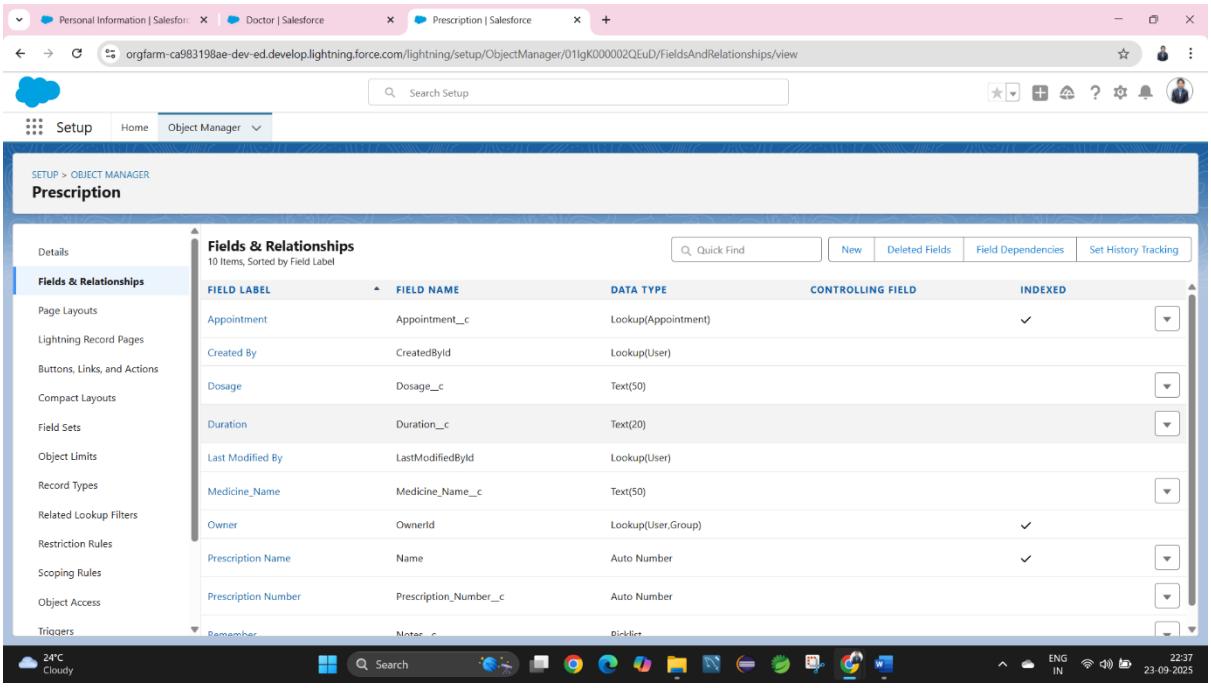
- **Doctor__c** → Availability, Specialization, Year_of_experience,etc



- **Appointment__c** → Appointment_Date, Status (Scheduled, Completed, Cancelled),Doctor,Patient,etc



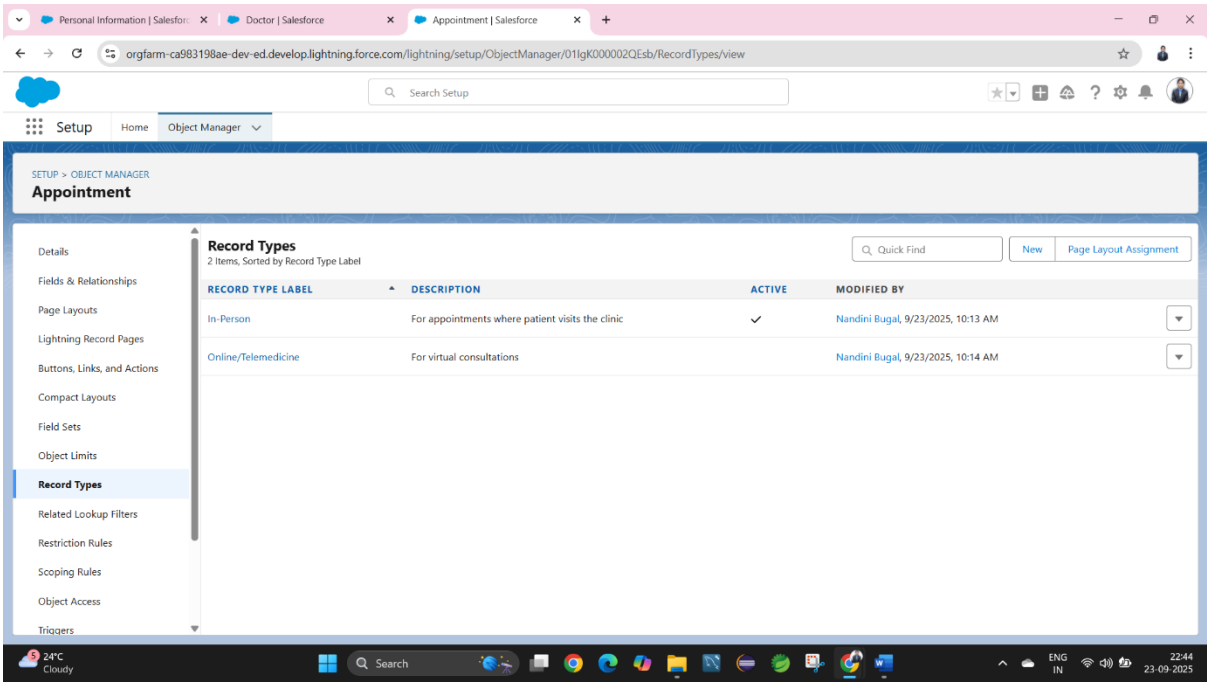
- **Prescription__c** → Medicine_Name__c, Dosage__c, Notes



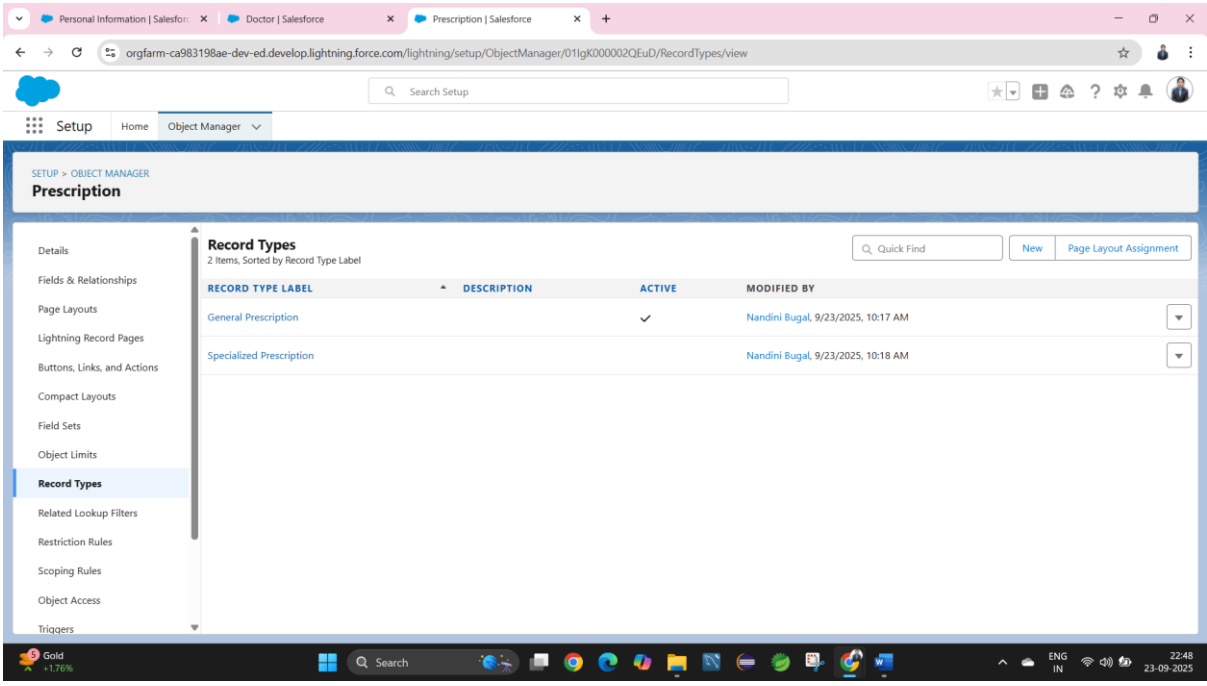
3. Record Types

Used when **different categories of the same object** need different layouts/workflows.

- **Appointment__c**:
  - **In-person**
  - **Online/Telemedicine**



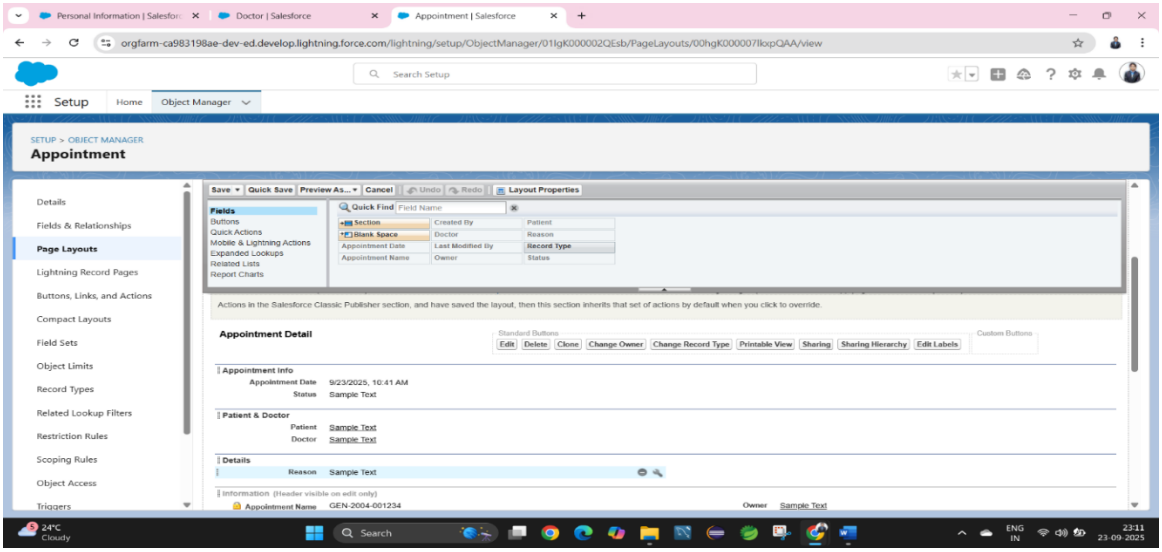
- **Prescription__c**:
  - **General Prescription**
  - **Specialized Prescription** (e.g., for chronic diseases).



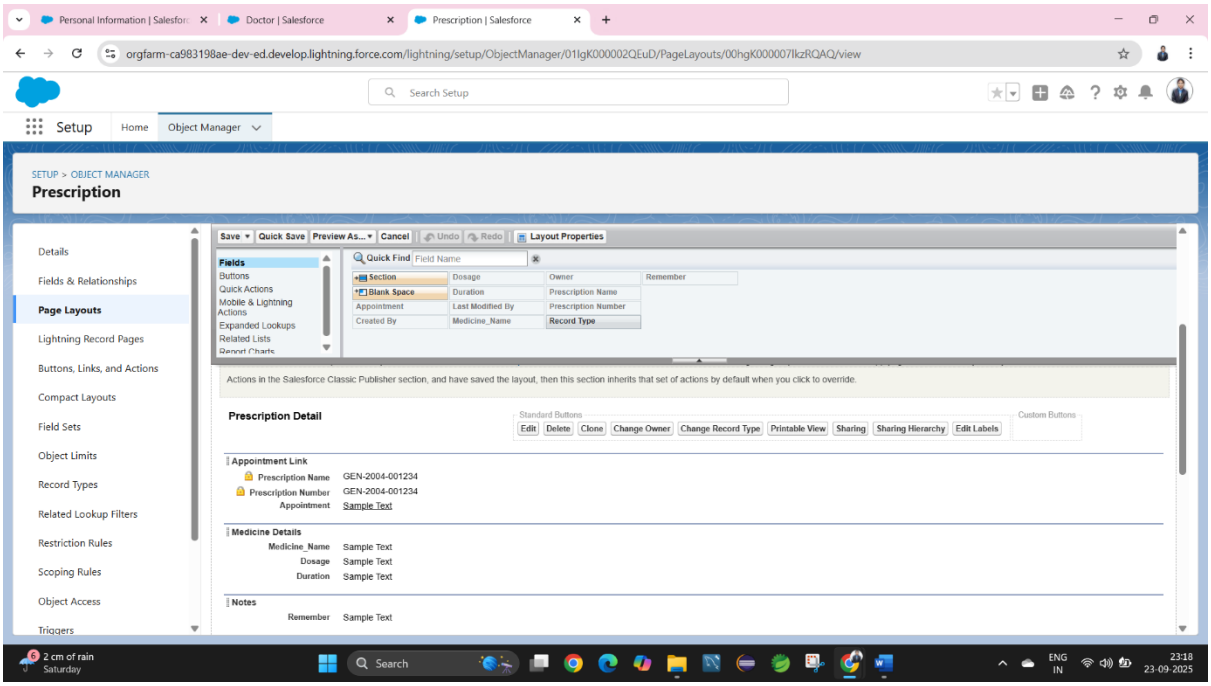
4. Page Layouts

Define **which fields appear** for each record type.

- **Doctor Layout** → Basic Info ,Contact info, Professional Info
- **Patient Layout** → Basic ,Contact ,Medical.
- **Appointment Layout** → Appointment, Patient & Doctor, Details.



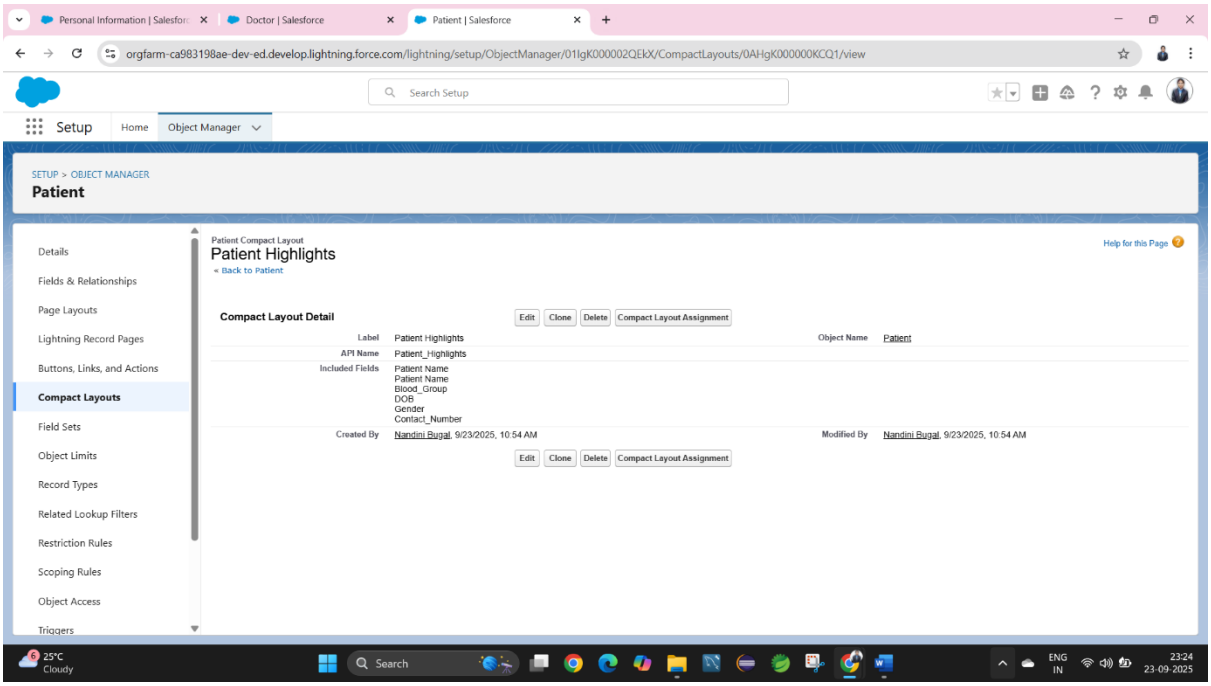
- **Prescription Layout** → Linked to Appointment, Medicine, Notes.



5. Compact Layouts

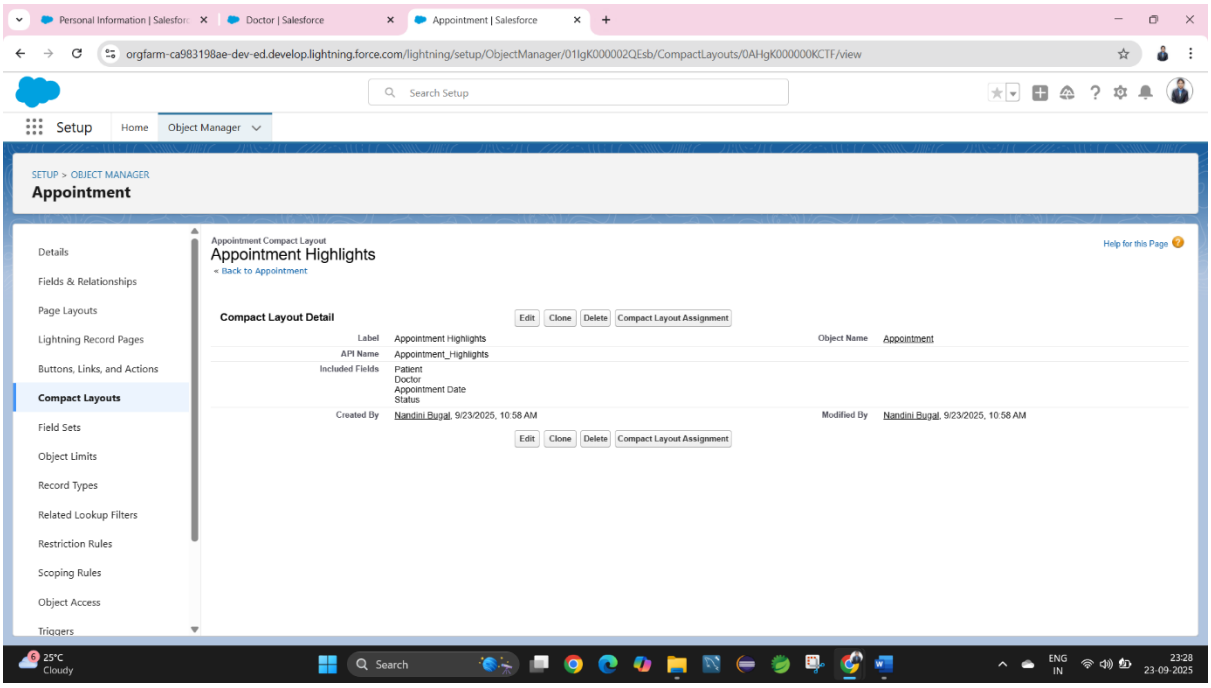
Show **key fields in highlights panel** (top of record page).

- **Patient__c** → Name, Age, Health ID, Blood Group.



- **Doctor__c** → Name, Specialization, Availability.

- **Appointment__c** → Patient Name, Doctor Name, Date, Status.

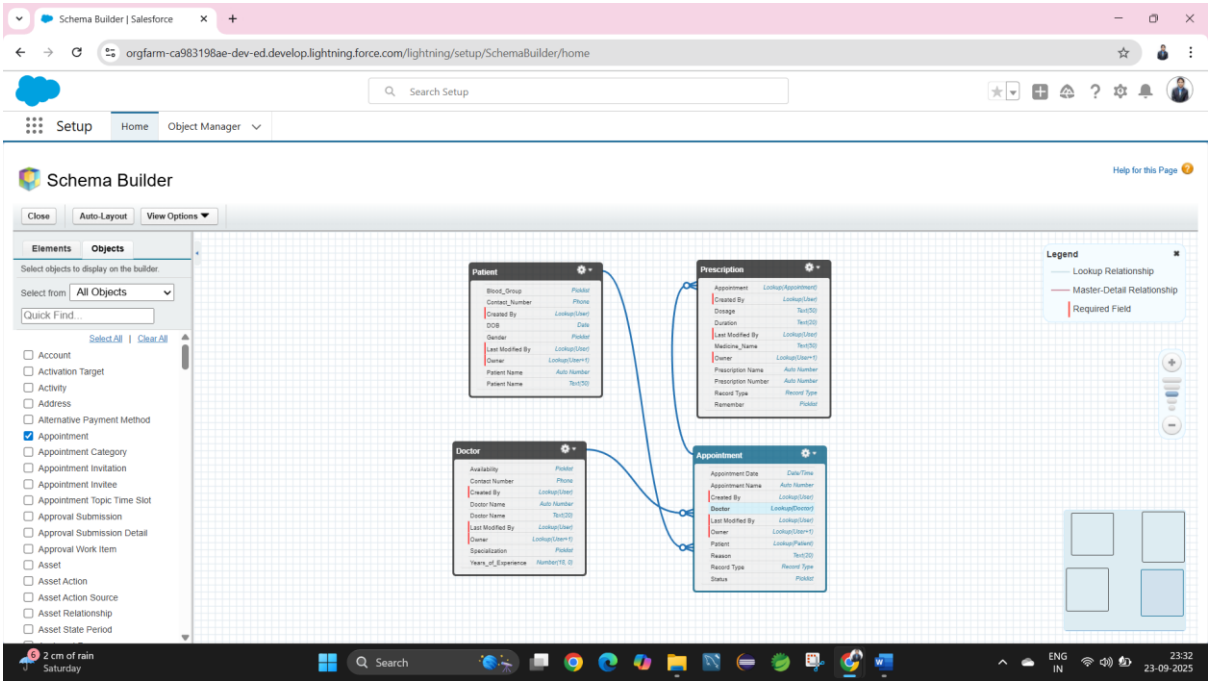


6. Schema Builder

Visual tool to design your **MediAid data model**:

- Drag and drop to show **relationships** between Patient, Doctor, Appointment, etc.

- Helps verify **lookup & master-detail connections**.



7. Lookup vs Master-Detail vs Hierarchical Relationships

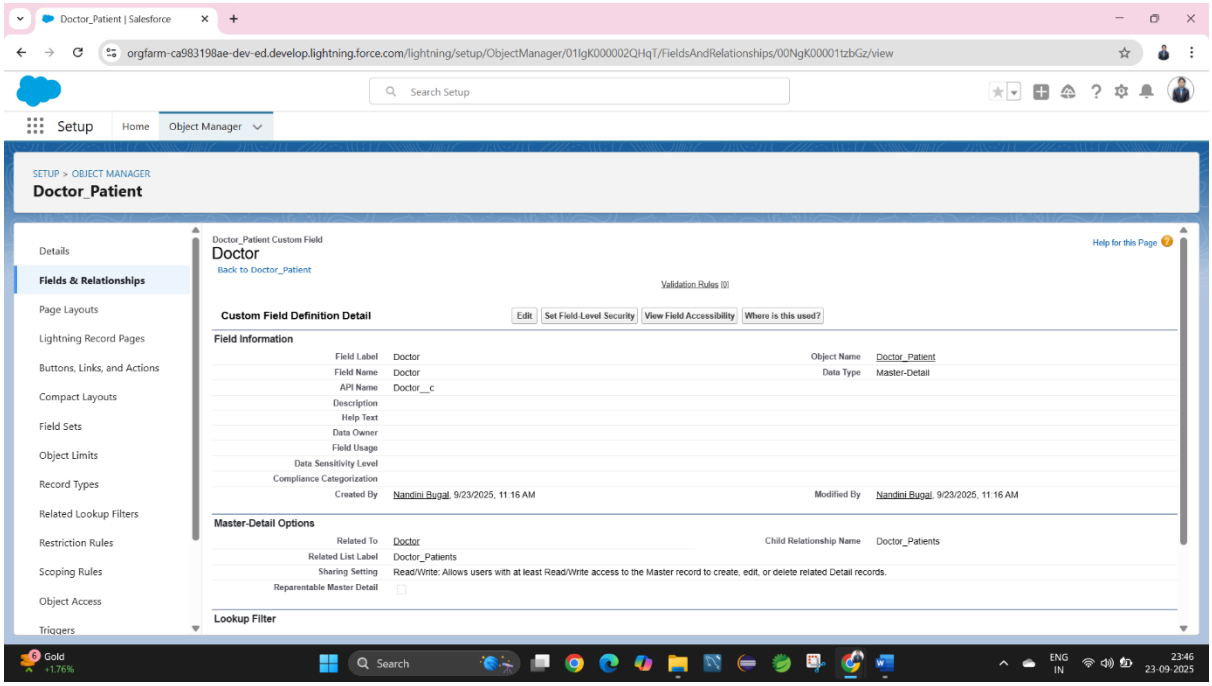
- **Lookup** → Loose connection (can exist without parent).
    - Example: **Appointment → Doctor** (appointment can exist even if doctor changes).
- **Master-Detail** → Strong connection (child record depends on parent).
    - Example: **Prescription → Appointment** (if appointment is deleted, prescription is deleted).
- **Hierarchical** → Only for **User object**.
    - Example: **Doctor → Doctor (Supervisor)** One Doctor (senior) supervises another Doctor (junior).

8. Junction Objects

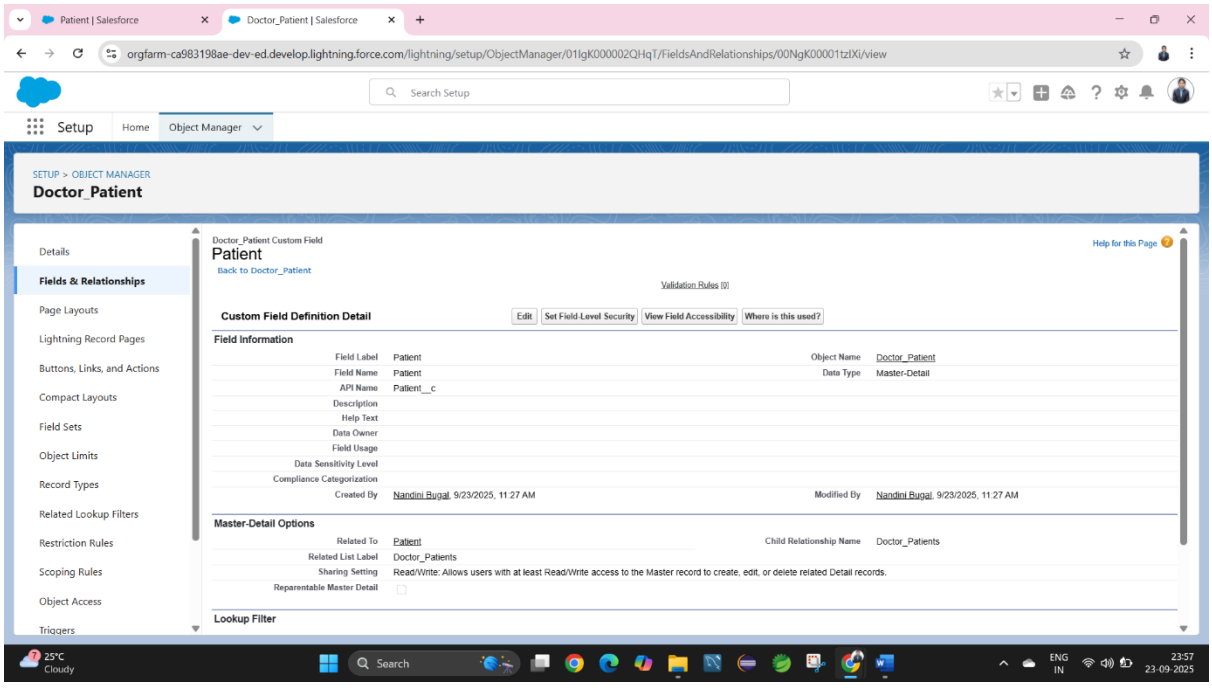
For **many-to-many relationships**.

- Example: **Doctor_Patient__c** (Junction Object).
    - One Patient can consult many Doctors.
    - One Doctor can treat many Patients.

- Junction object stores → Patient ID + Doctor ID + Additional info (First Visit Date, Consultation Type).
- **Doctor:**



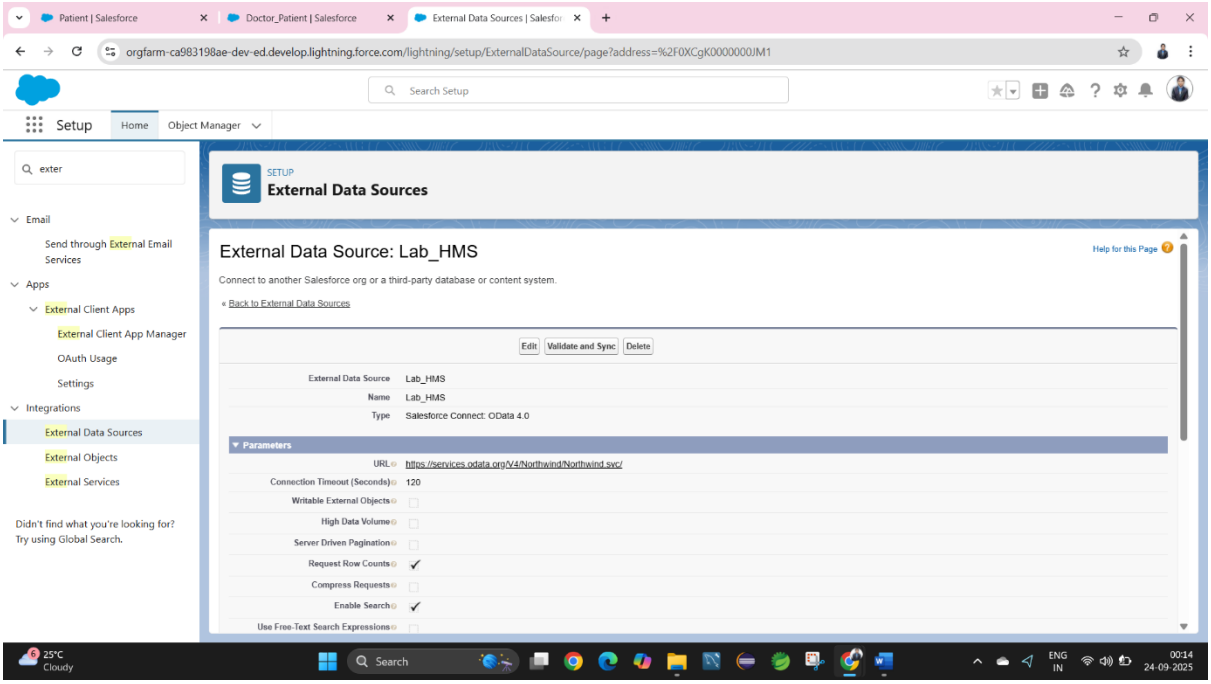
Patient:



9. External Objects

Used to **connect with external healthcare systems/databases** without storing data in Salesforce.

- Example: Lab Reports stored in an external **Hospital Management System (HMS)** → can be accessed in Salesforce as **External Object: Lab_Report__x**.
- Use **Salesforce Connect** to fetch external patient insurance or diagnostic data.



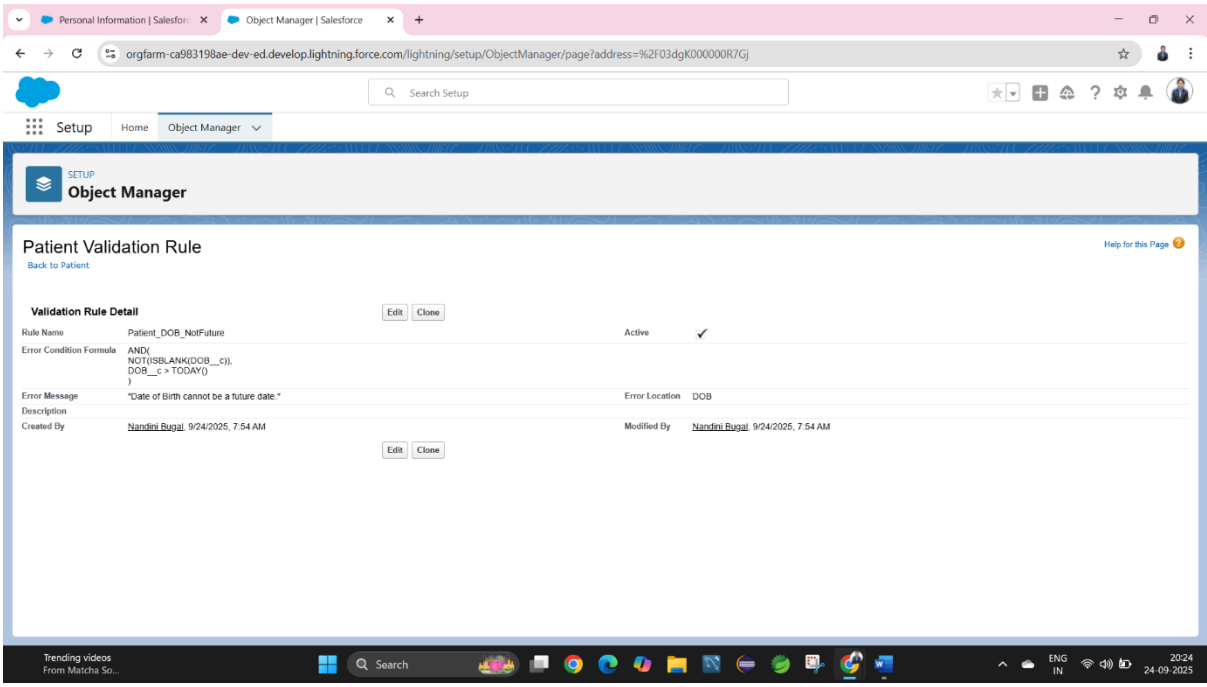
MediAid Healthcare Project

Phase 4: Process Automation (Admin)

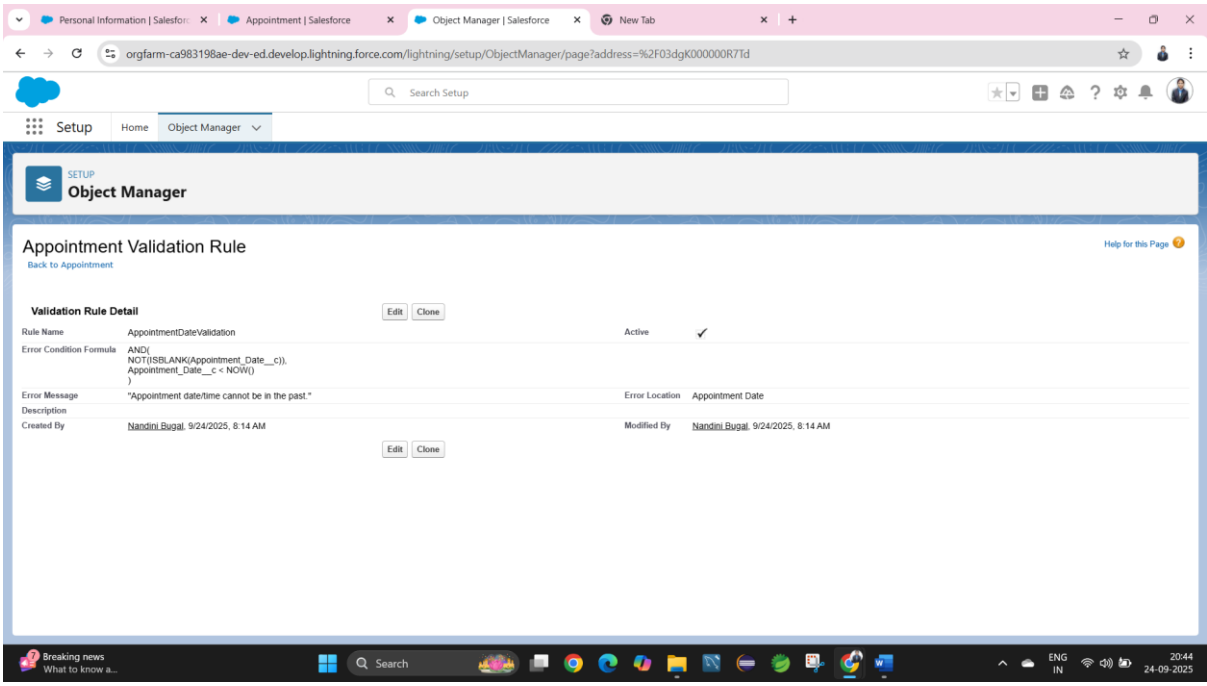
Process automation in Salesforce ensures that repetitive tasks are minimized, data quality is maintained, and business processes run smoothly with minimal manual effort.

1. Validation Rules

- Enforce correct data entry for critical fields.
- Examples:
    - Patient__c: DOB cannot be a future date.



    - Prescription__c: Dosage must be greater than 0.
    - Appointment__c: Appointment Date cannot be in the past.

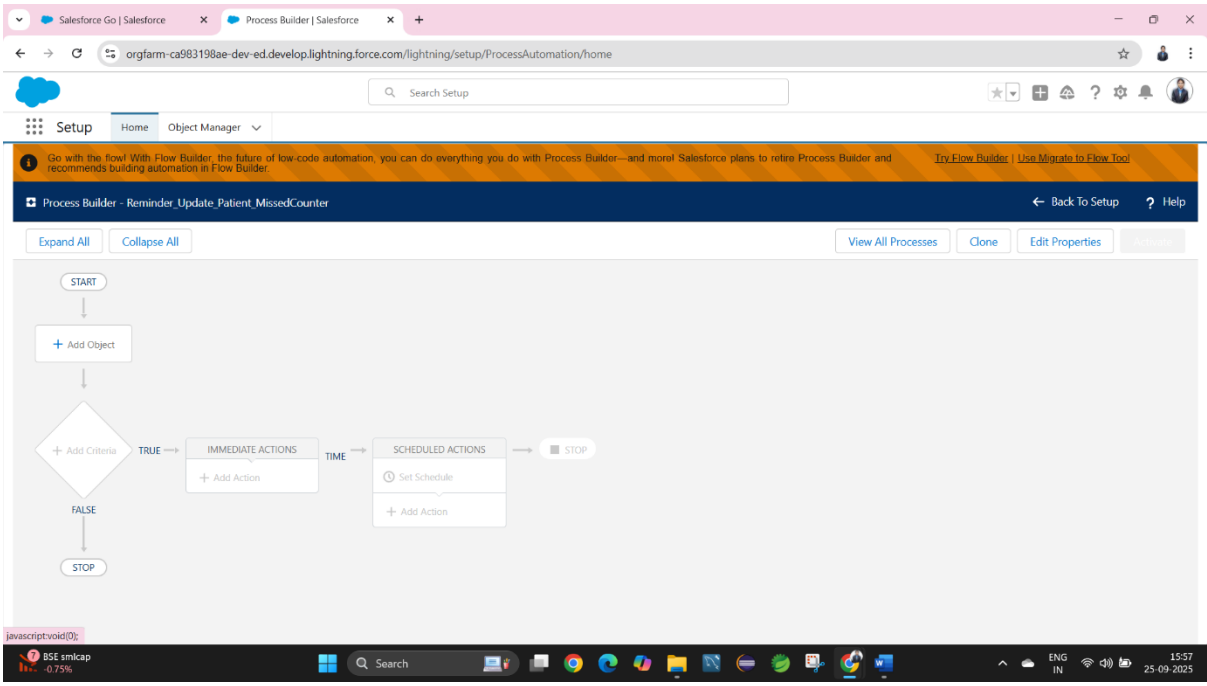


2. Workflow Rules ***(Legacy, but still useful)***

- Automate simple if/then actions.
- Examples:
    - When a **Prescription__c** is created, send an email to the assigned patient.

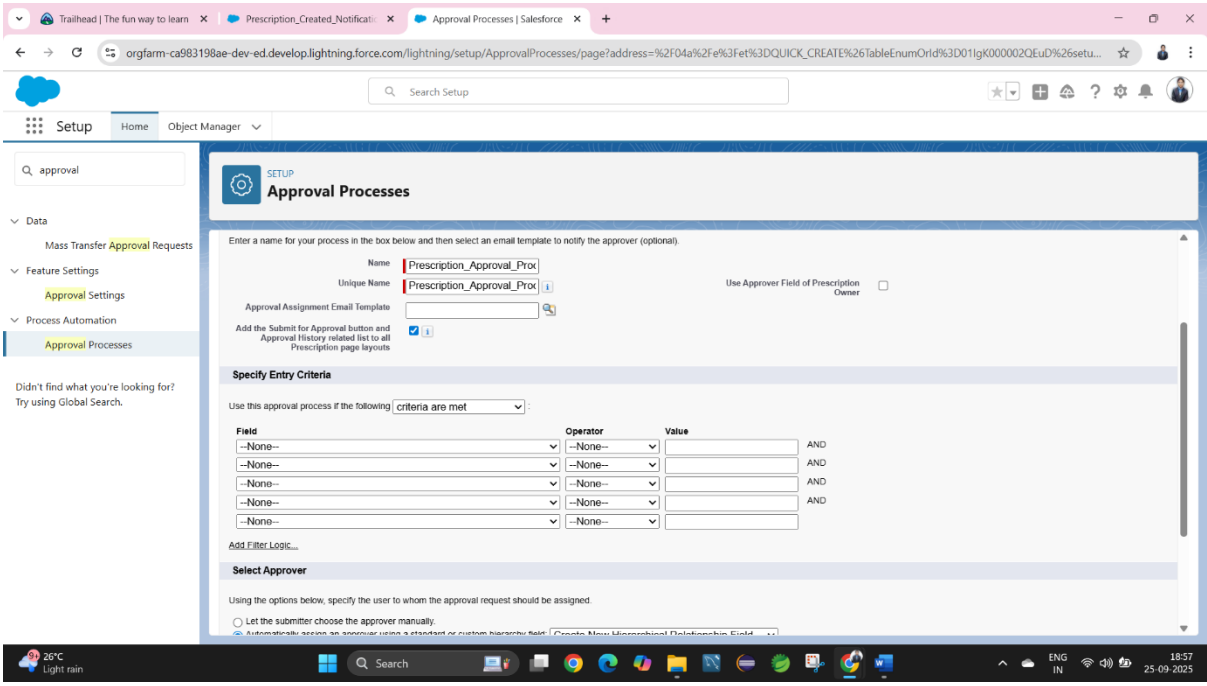
3. Process Builder

- Advanced automation with multiple conditions.
- Examples:
    - If a **Patient__c** misses 3 reminders in a row → Update Patient Status = "Non-Adherent" + Notify Caregiver.



4. Approval Process

- Structured approval workflows.
- Examples:
    - Prescription approval by **Senior Doctor** before being sent to Patient.



5. Flow Builder(Optional)

Salesforce Flow will be the **primary automation tool** going forward.

- **Screen Flow**: Patient self-service form to request a prescription refill.

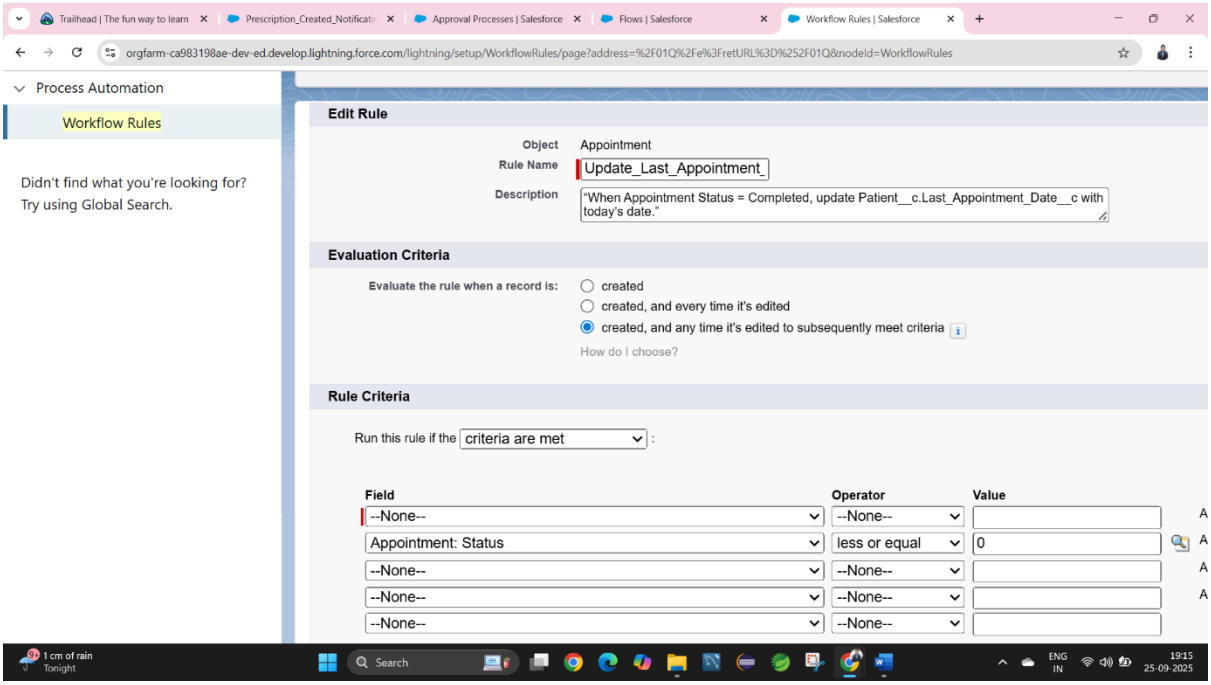
- **Record-Triggered Flow**:
    - When a **Prescription__c** is created → Send Email + Create Reminder Record.
    - When a **Patient__c** record is updated with "High Risk" flag → Notify assigned Doctor immediately.
- **Scheduled Flow**: Daily check to send summary of missed doses to caregivers.
- **Auto-launched Flow**: Triggered from Process Builder to create Reminder Logs.

6. Email Alerts

- Templates for Patients, Doctors, Caregivers.
- Examples:
    - Appointment confirmation email to Patient.
    - Prescription refill reminder to Caregiver.

7. Field Updates

- Automatic updates on record fields.
- Example: When Appointment Status = Completed → Update "Last Appointment Date" on Patient__c.

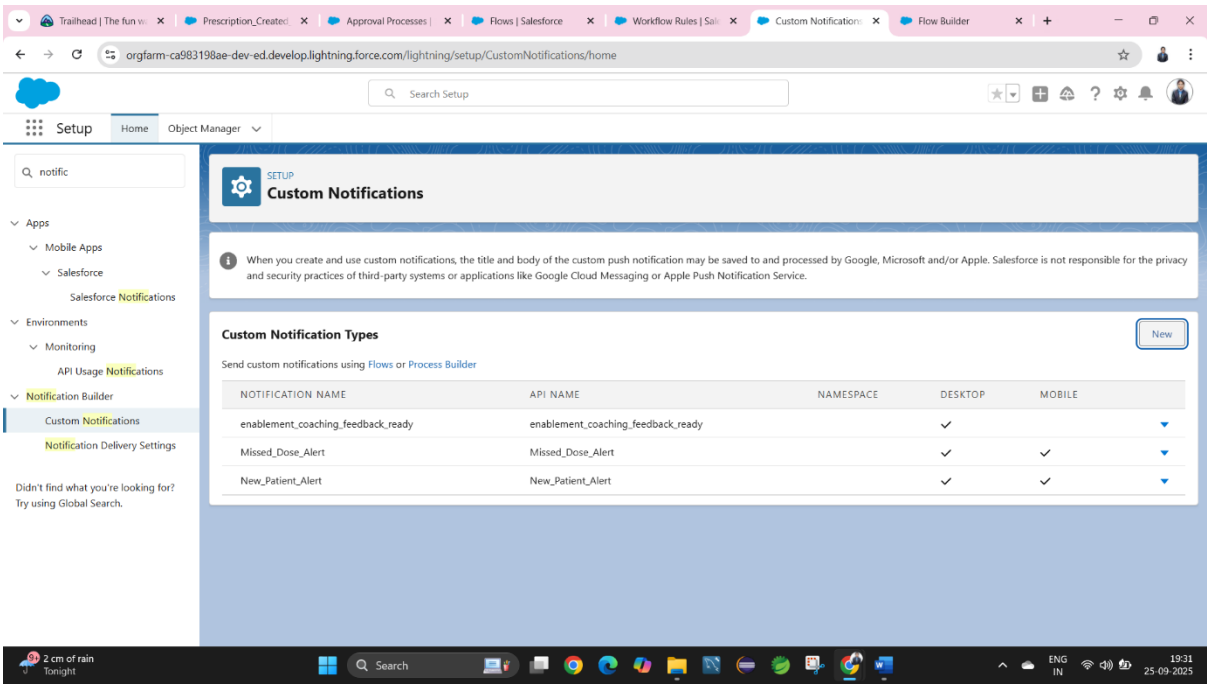


8. Tasks(Optional)

- Auto-create tasks for users.
- Example:
    - Assign task to Doctor after every missed dose report.

9. Custom Notifications

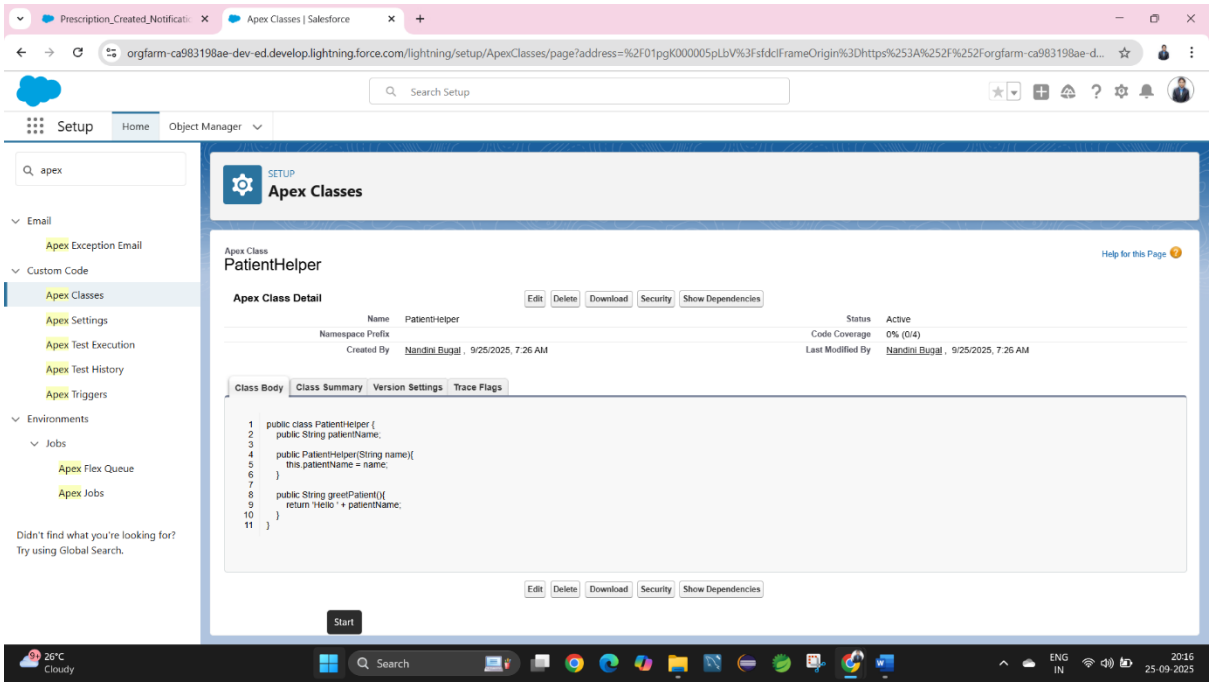
- Real-time alerts within Salesforce Mobile or Lightning.
- Examples:
    - Notify Doctor instantly when a patient misses 2 consecutive doses.



Phase 5: Apex Programming (Developer)

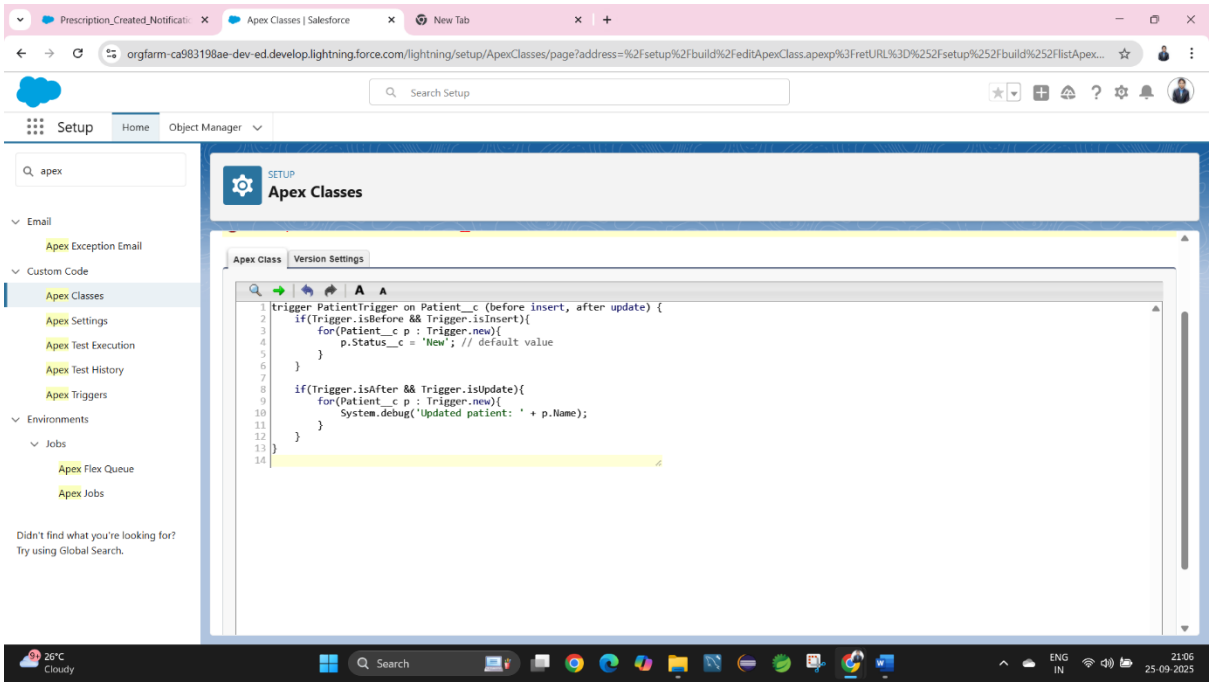
1. Classes & Objects

- Apex classes are templates for objects; contain **variables, constructors, and methods**.



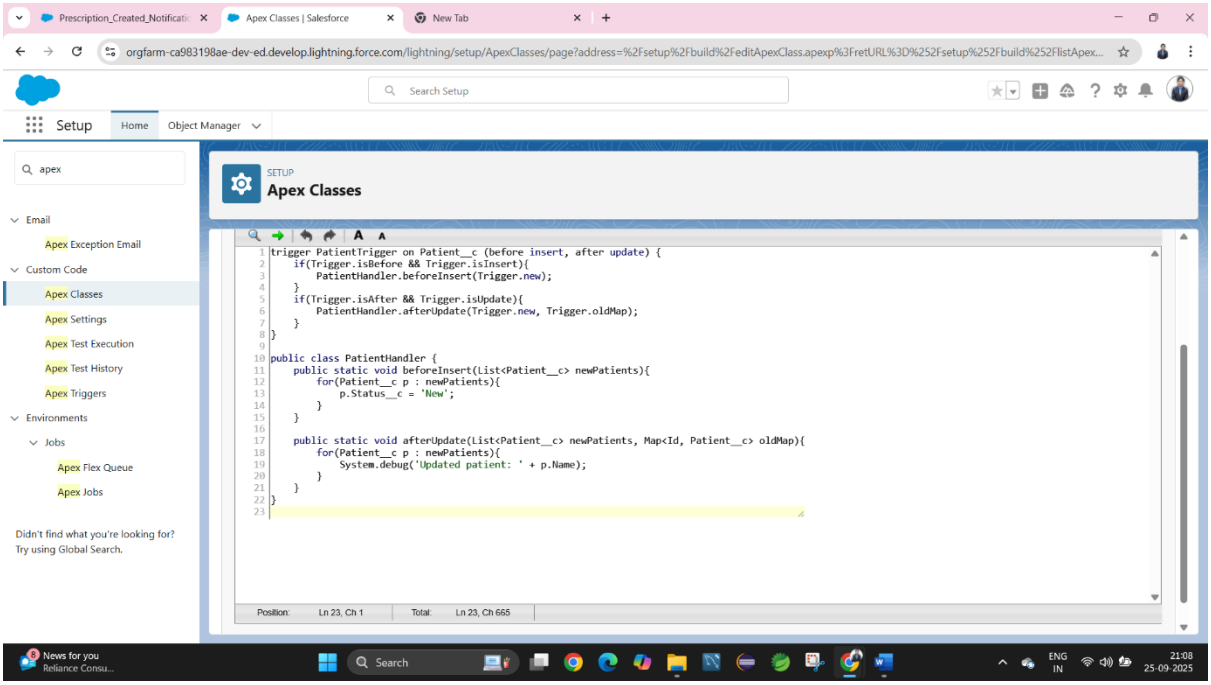
2. Apex Triggers (before/after insert/update/delete)

- Triggers execute **before or after** DML operations.



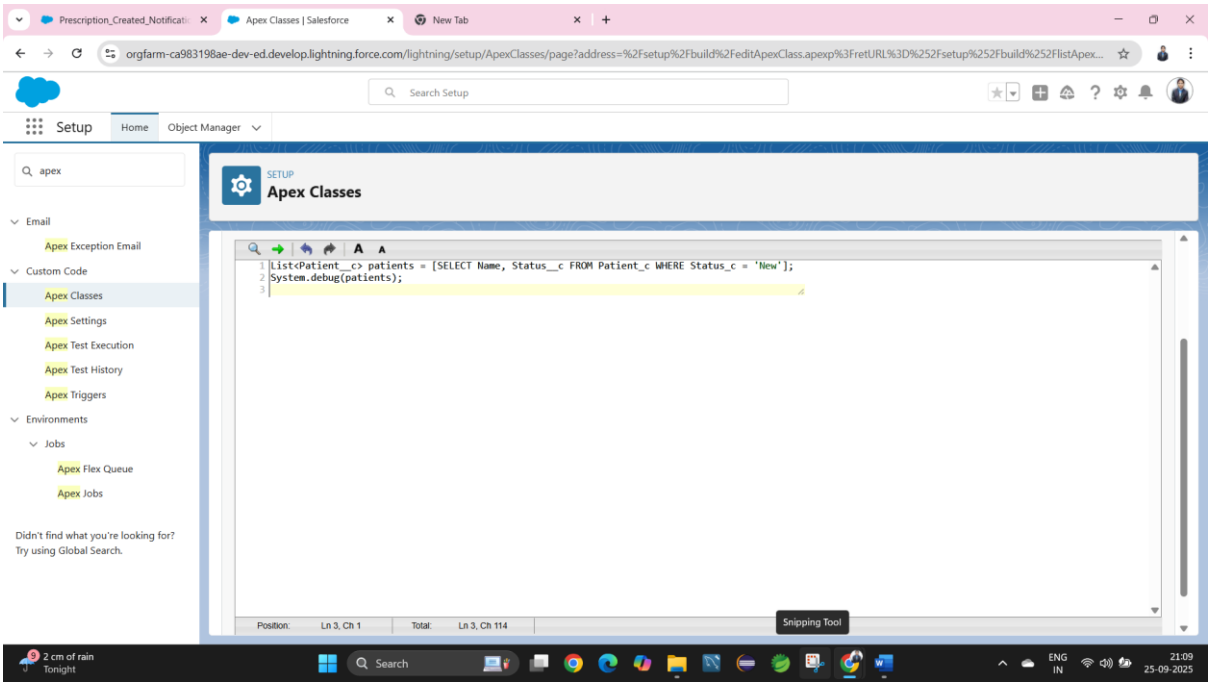
3. Trigger Design Pattern

- Separates logic from trigger into a **handler class**.
- Makes code **clean, reusable, and maintainable**.



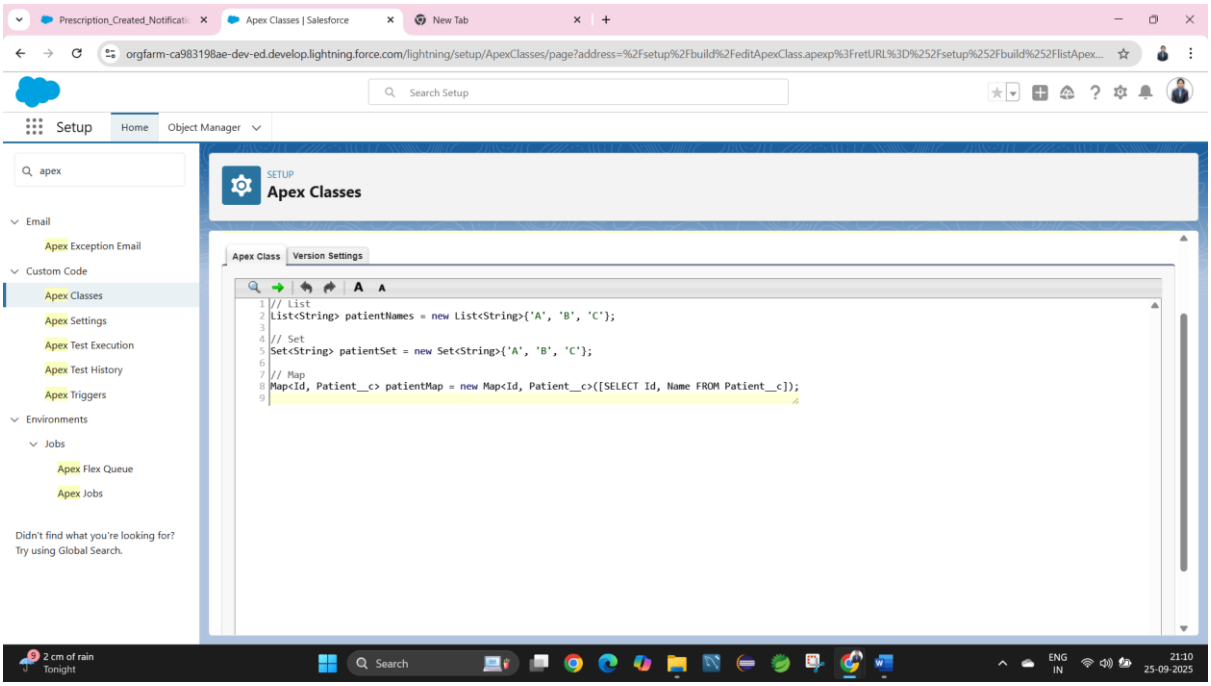
4. SOQL & SOSL

- **SOQL (Salesforce Object Query Language)** → Query records.
- **SOSL (Salesforce Object Search Language)** → Search across multiple objects.



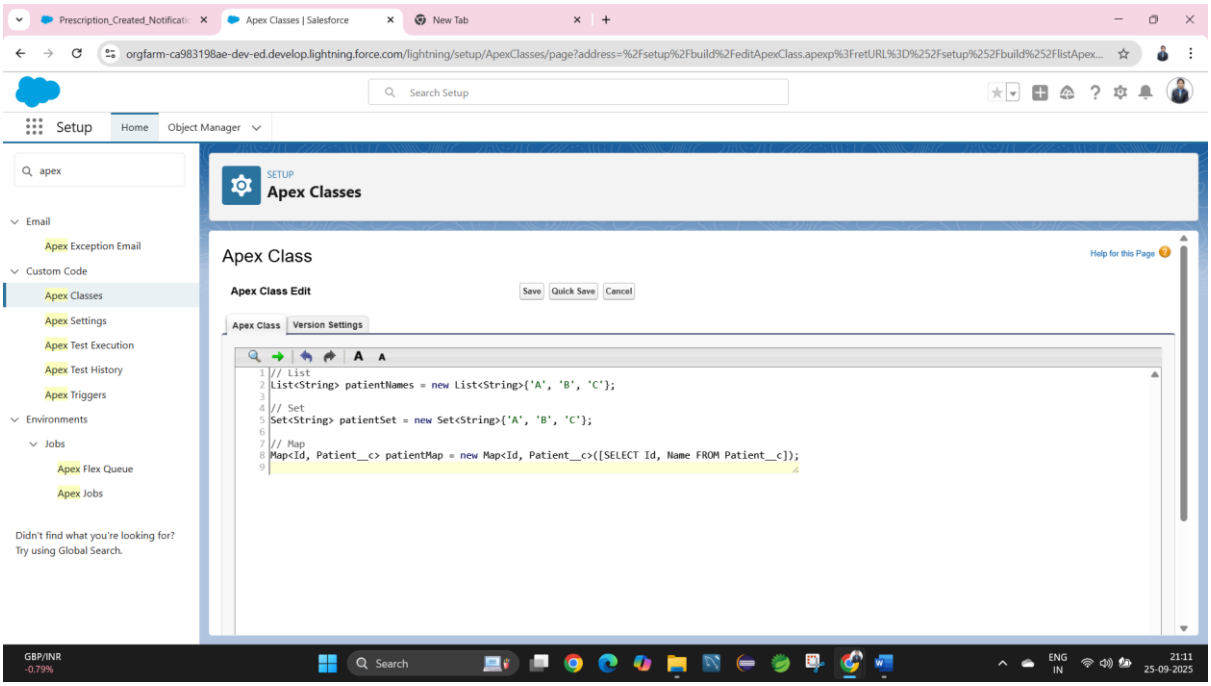
5. Collections (List, Set, Map)

- Apex supports **List, Set, and Map** for bulk operations.



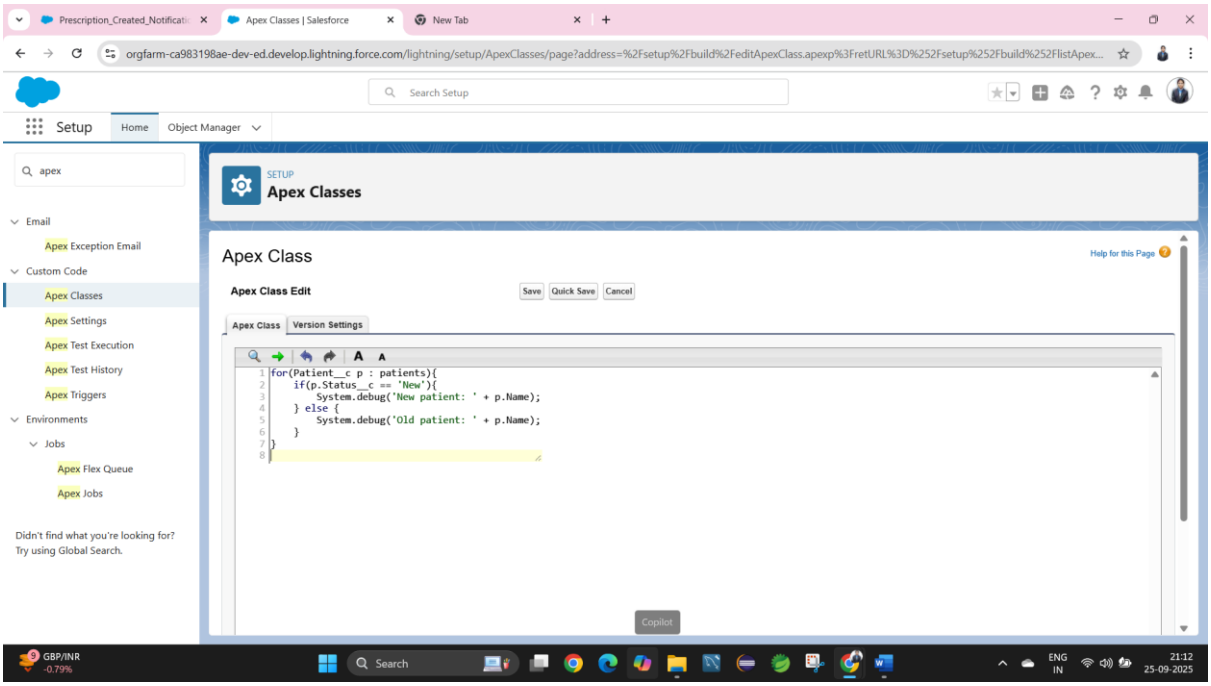
6. Control Statements

- Use **if-else, for, while, switch** for conditional logic.



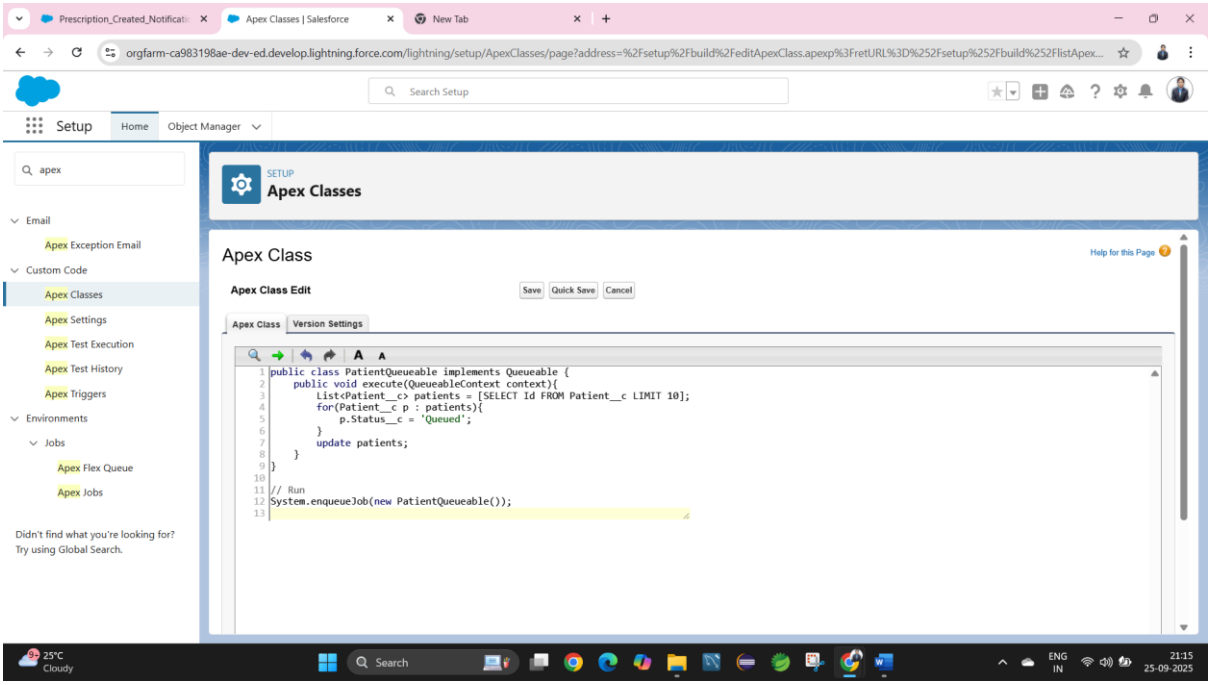
7. Batch Apex

- Processes **large records asynchronously**.
- Implement Database.Batchable<SObject> interface.



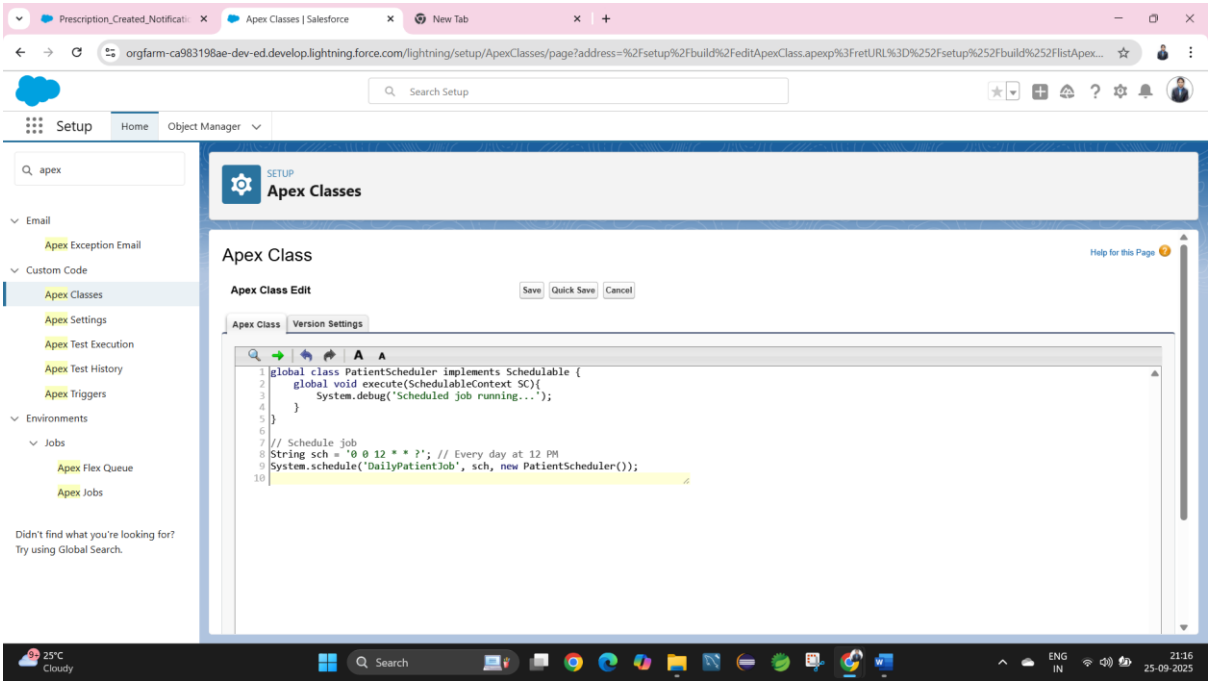
8. Queueable Apex

- Similar to Batch Apex but simpler for **asynchronous jobs**.



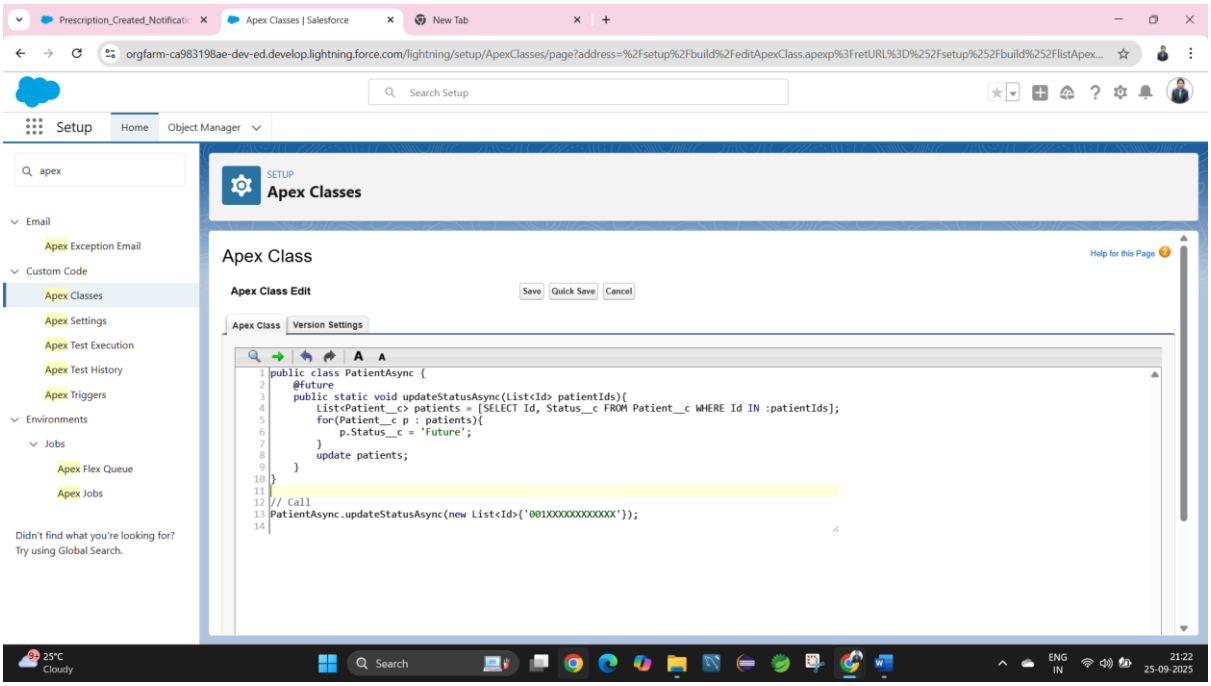
9. Scheduled Apex

- Schedule Apex classes to run at a specific **time or interval**.



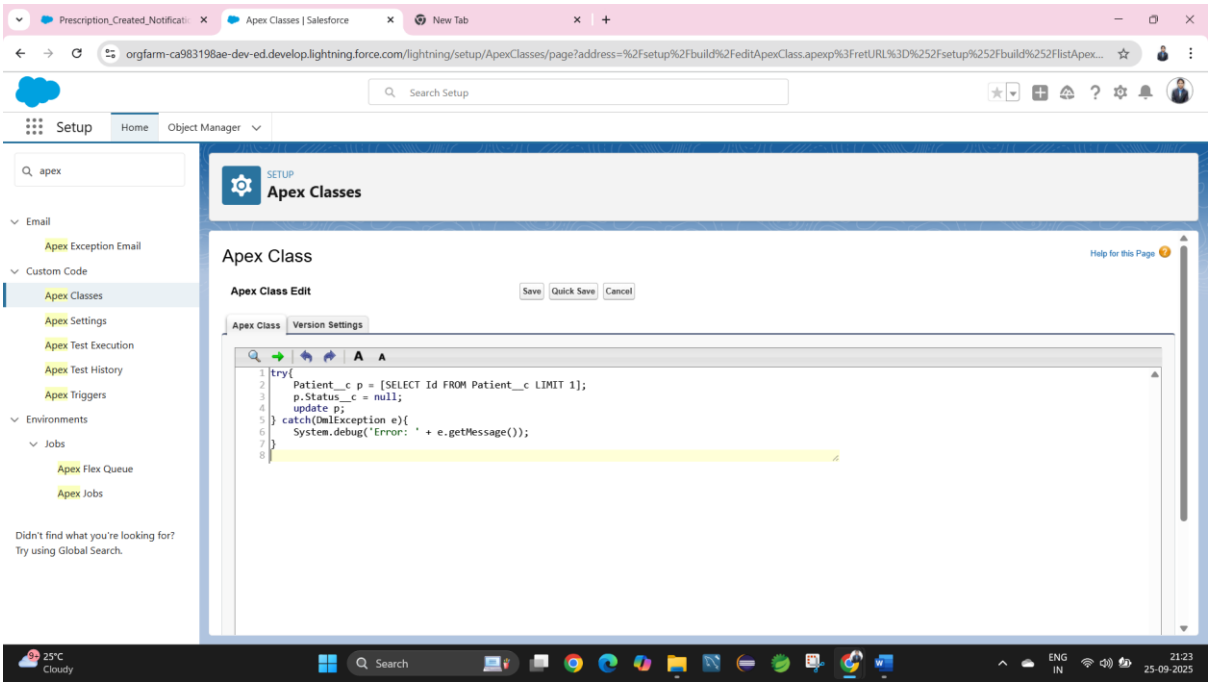
10. Future Methods

- Executes **methods asynchronously**.



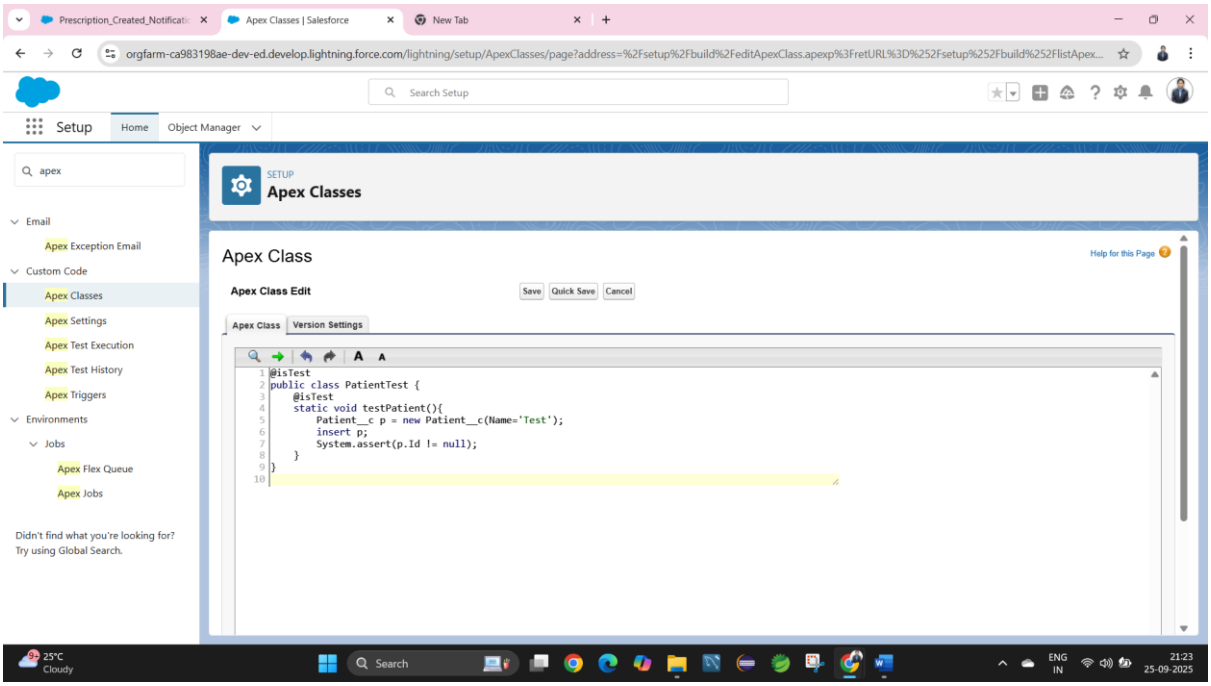
11. Exception Handling

- Use try-catch blocks to handle **errors gracefully**.



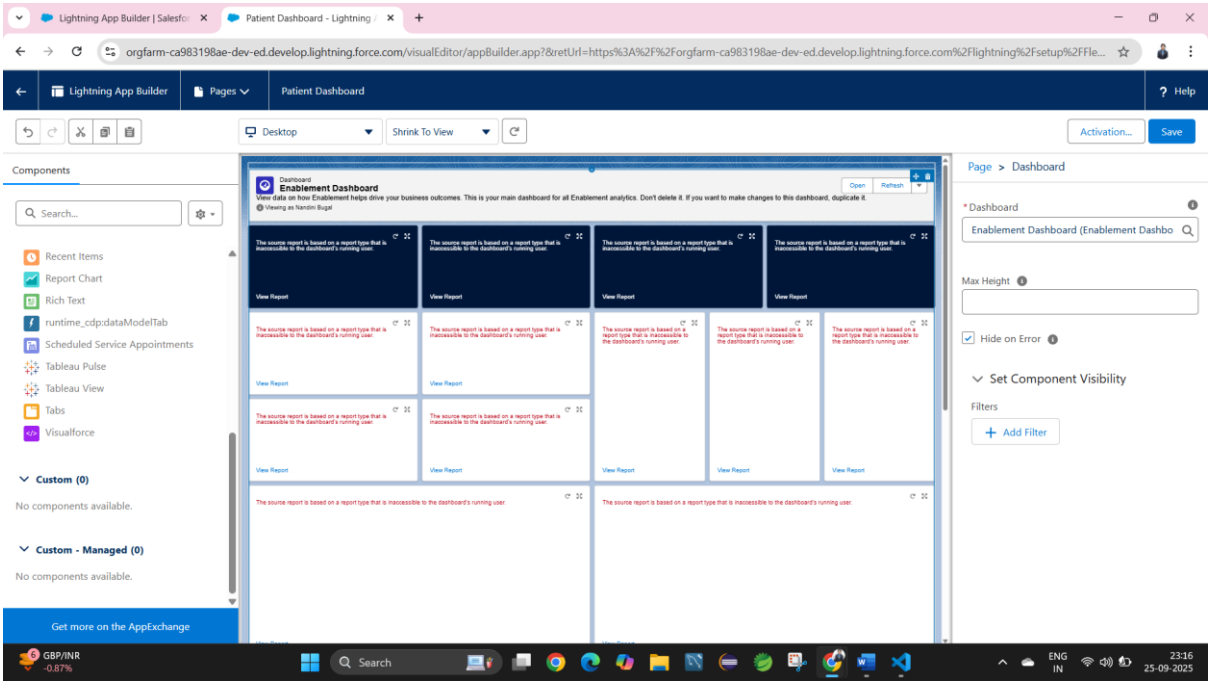
12. Test Classes

- Mandatory for **deployment**; must cover ≥75% of Apex code.


Prescription_Created_Notificati
Apex Classes | Salesforce
New Tab
orgfarm-ca983198ae-dev-ed.develop.lightning.force.com/lightning/setup/ApexClasses/page?address=%2Fsetup%2Fbuild%2FeditApexClass.apexp%3FretURL%3D%252Fsetup%252Fbuild%252FlistApex...
Search Setup
Setup
Home
Object Manager
apex
Email
Apex Exception Email
Custom Code
Apex Classes
Apex Settings
Apex Test Execution
Apex Test History
Apex Triggers
Environments
Jobs
Apex Flex Queue
Apex Jobs
Didn't find what you're looking for?
Try using Global Search.
SETUP
Apex Classes
Apex Class
Help for this Page
Apex Class Edit
Save
Quick Save
Cancel
Apex Class
Version Settings
@isTest
public class PatientTest {
@isTest
static void testPatient(){
Patient__c p = new Patient__c(Name='Test');
insert p;
System.assert(p.Id != null);
}
}
25°C
Cloudy
Search
ENG
IN
21:23
25-09-2025

Phase 6: User Interface Development

1. Lightning App Builder

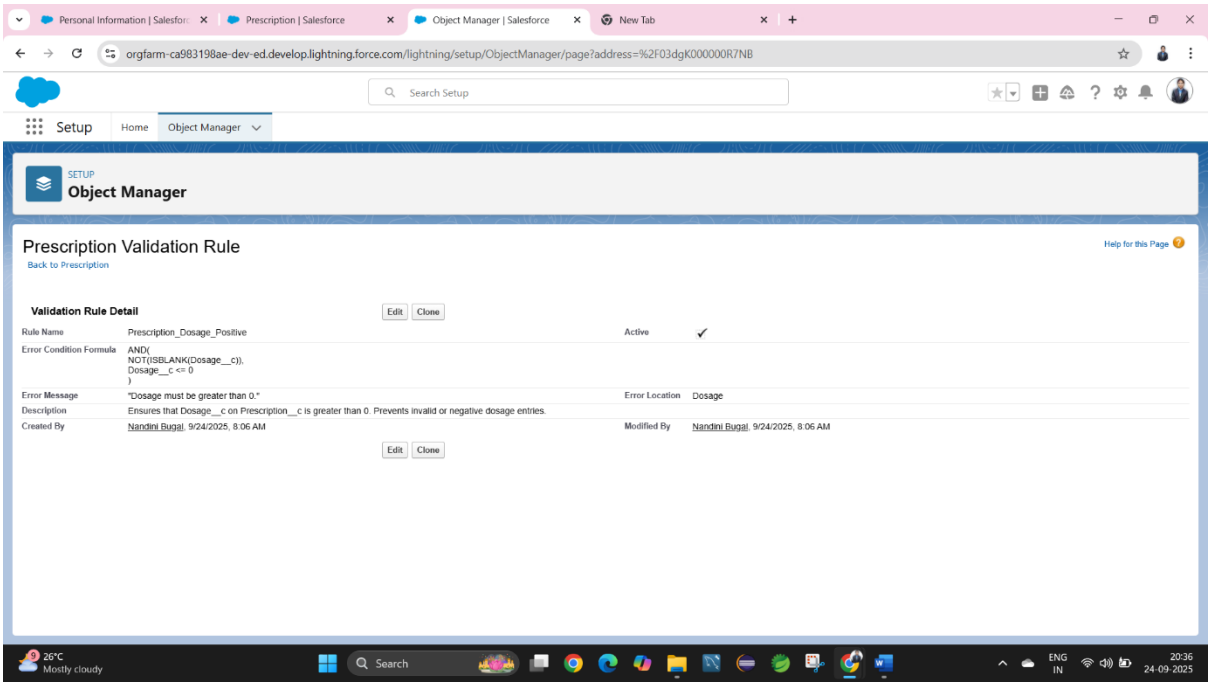
- Drag-and-drop tool to create custom pages for Salesforce Lightning Experience and Mobile.
- Combine standard, custom, and third-party components.
- 

2. Record Pages

- Customize how a record (e.g., Patient__c, Appointment__c) appears.
- Arrange tabs, fields, and components to match user needs.

3. Tabs

- Provide easy navigation to objects, apps, or web pages.
- Add custom object tabs (e.g., Prescription__c tab).

- 

4. Home Page Layouts

- Customize Home Page with dashboards, tasks, events, and custom components.
- Role-based Home Pages for different user profiles.

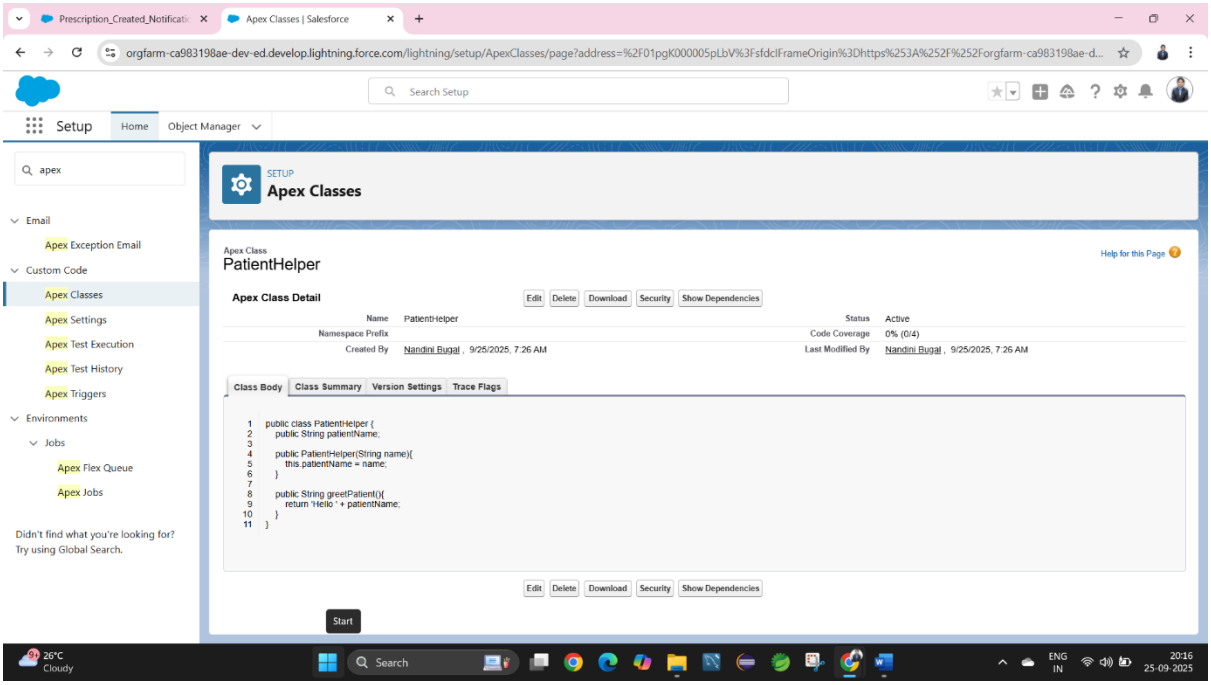
5. Utility Bar

- Persistent tools at the bottom of Lightning apps.
- Examples: Notes, Recent Items, Softphone, or a custom “Quick Actions” component.

6. LWC (Lightning Web Components)

- Modern UI framework for building fast, reusable web components.

- Built on standard web technologies (HTML, CSS, JavaScript).



7. Apex with LWC

- Use Apex to fetch, insert, or update Salesforce data not available via standard wire adapters.
- Example: Fetching filtered patient reports from Apex and showing them in an LWC table.

8. Events in LWC

- Communication mechanism between components.
- Types:
  - Child-to-Parent (Custom Events).
  - Parent-to-Child (Public properties).
  - Pub/Sub (unrelated components).

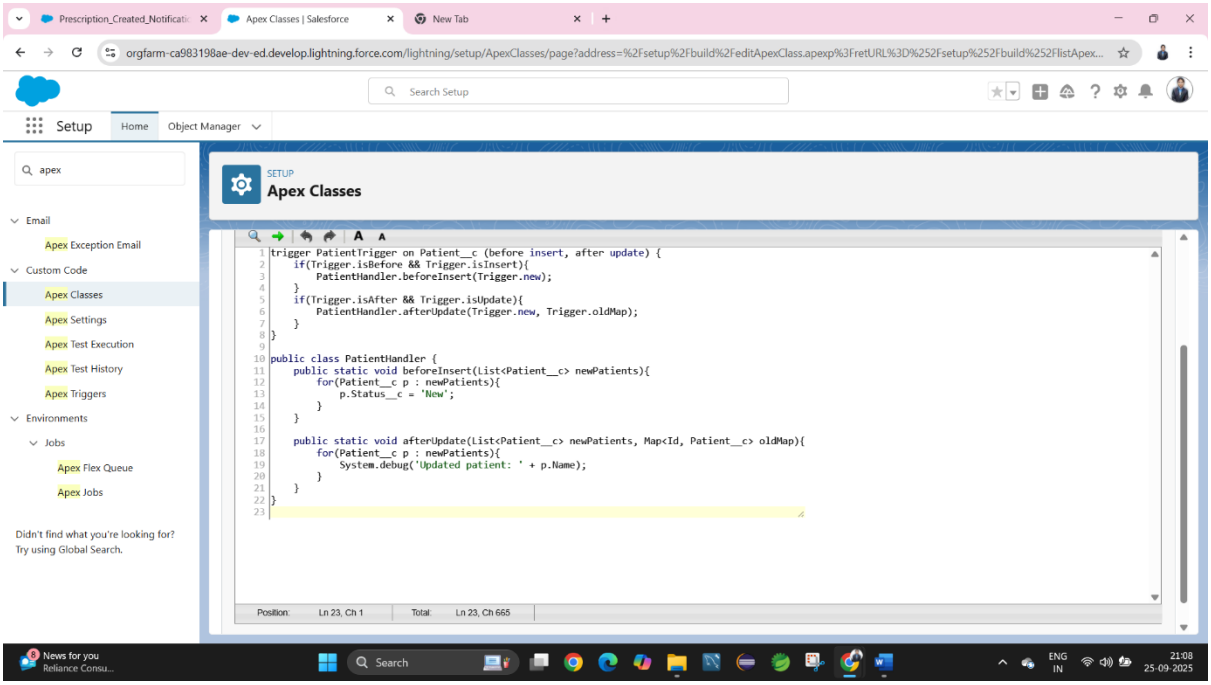
9. Wire Adapters

- Reactive way to fetch Salesforce data without explicit Apex calls.
- Example: @wire(getRecord, { recordId: '$recordId', fields }).

10. Imperative Apex Calls

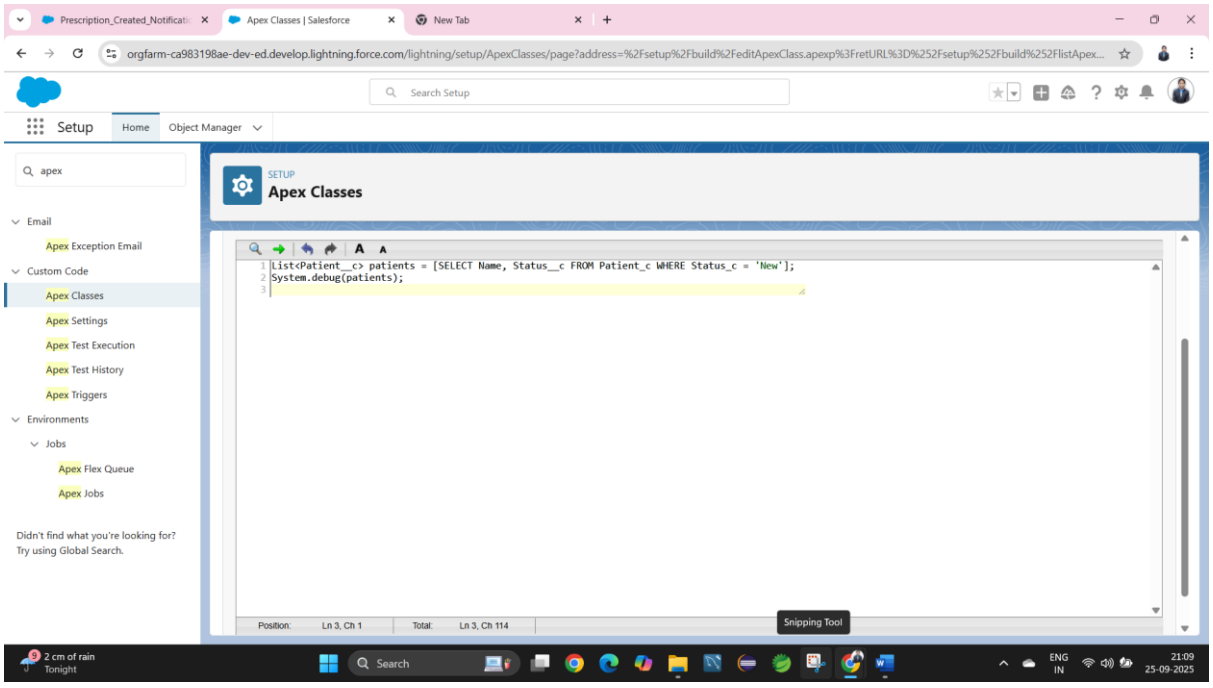
- Used when wire adapters are not sufficient (e.g., dynamic parameters, conditional calls).

- Example: Calling getPatientHistory Apex method on button click.



11. Navigation Service

- Used to navigate to standard or custom pages programmatically.
- Example: Redirecting to a Patient__c record after creation.



Phase 7: Integration & External Access – MediAid Project

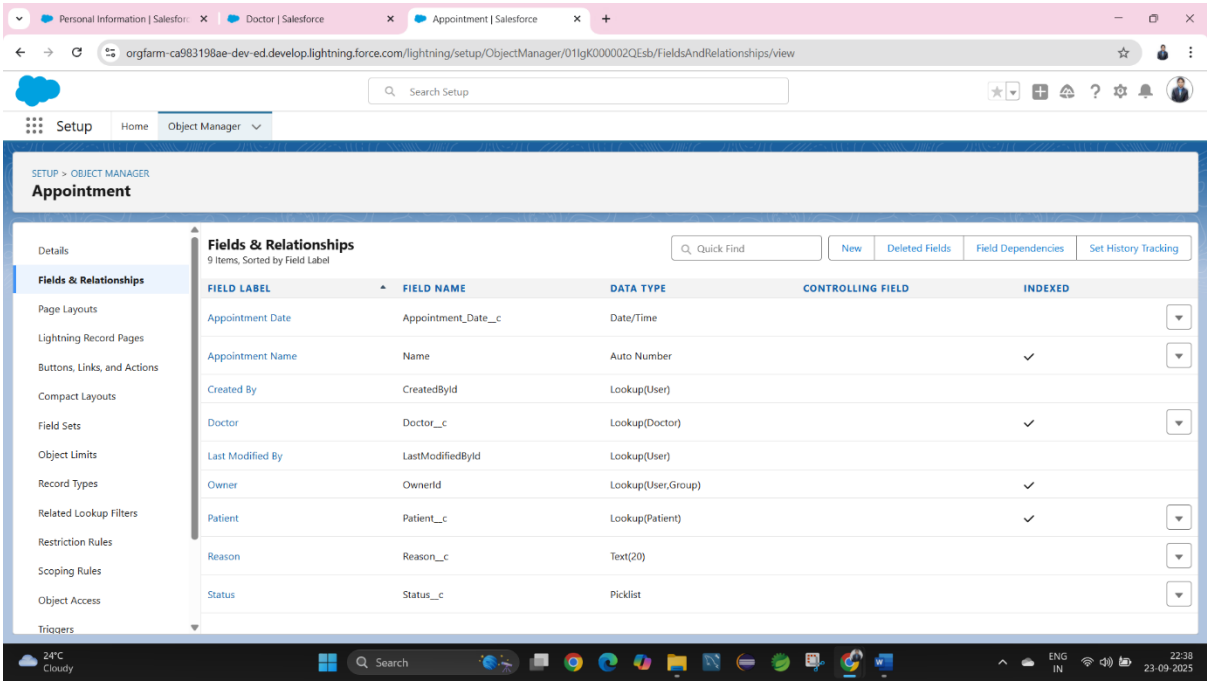
1. Named Credentials

Purpose

Used to securely store authentication details (like endpoint URL, username, password, or token) for external services — for example, connecting MediAid with an **External Pharmacy API** or **Lab Management System**.

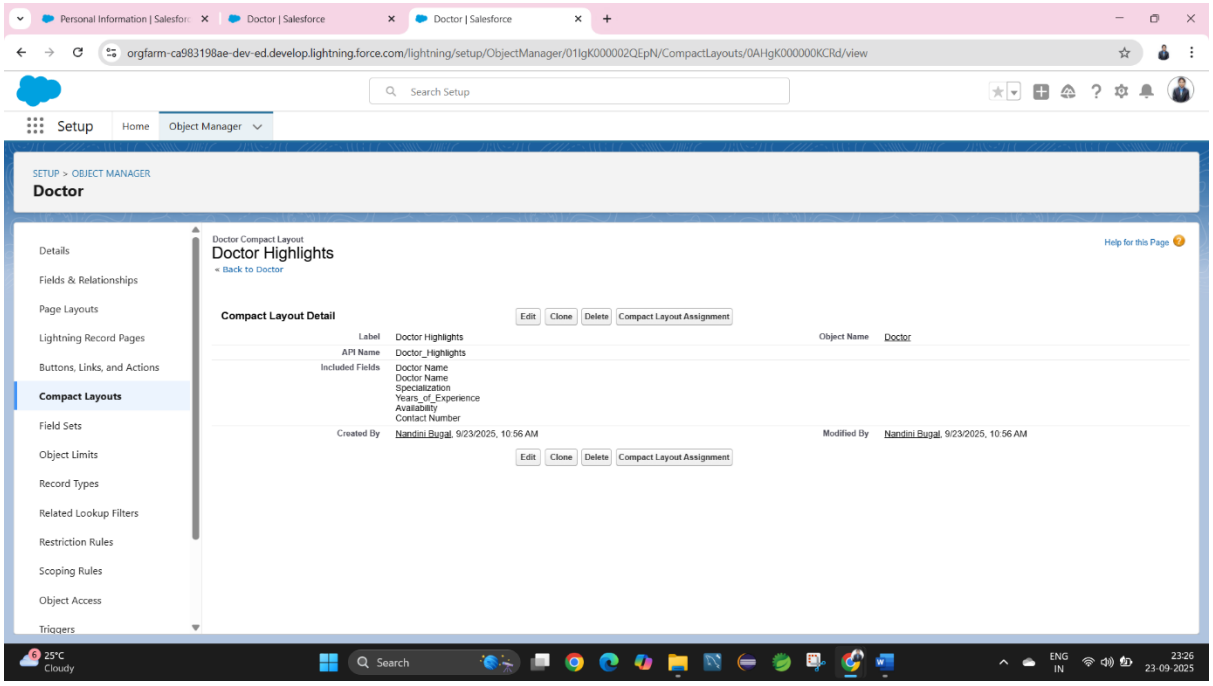
Steps

1. Go to **Setup → Named Credentials → New Named Credential**.
2. Enter:
    - **Label**: *PharmacyAPI_Credential*
    - **URL**: https://api.pharmacydata.com
    - **Identity Type**: Named Principal
    - **Authentication Protocol**: Password Authentication / OAuth 2.0



3. Click **Save**.
4. Use this Named Credential in Apex callouts without exposing credentials in code:
5. HttpRequest req = new HttpRequest();
6. req.setEndpoint('callout:PharmacyAPI_Credential/prescriptions');
7. req.setMethod('GET');

8. HttpResponse res = new Http().send(req);



2. External Services

Purpose

To connect MediAid with external REST APIs **without writing Apex code**, using declarative tools.

Example Use Case

Connecting with a **Lab Test Booking API** to fetch test results.

Steps

1. Go to **Setup → External Services → New External Service**.
2. Upload or paste the **OpenAPI (Swagger) specification** of the Lab API.
3. Salesforce automatically generates **Apex actions** that can be used in **Flows** or **LWC**.
4. Add it to a **Flow** in MediAid that automatically retrieves lab results for a patient.

3. Web Services (REST/SOAP)

Purpose

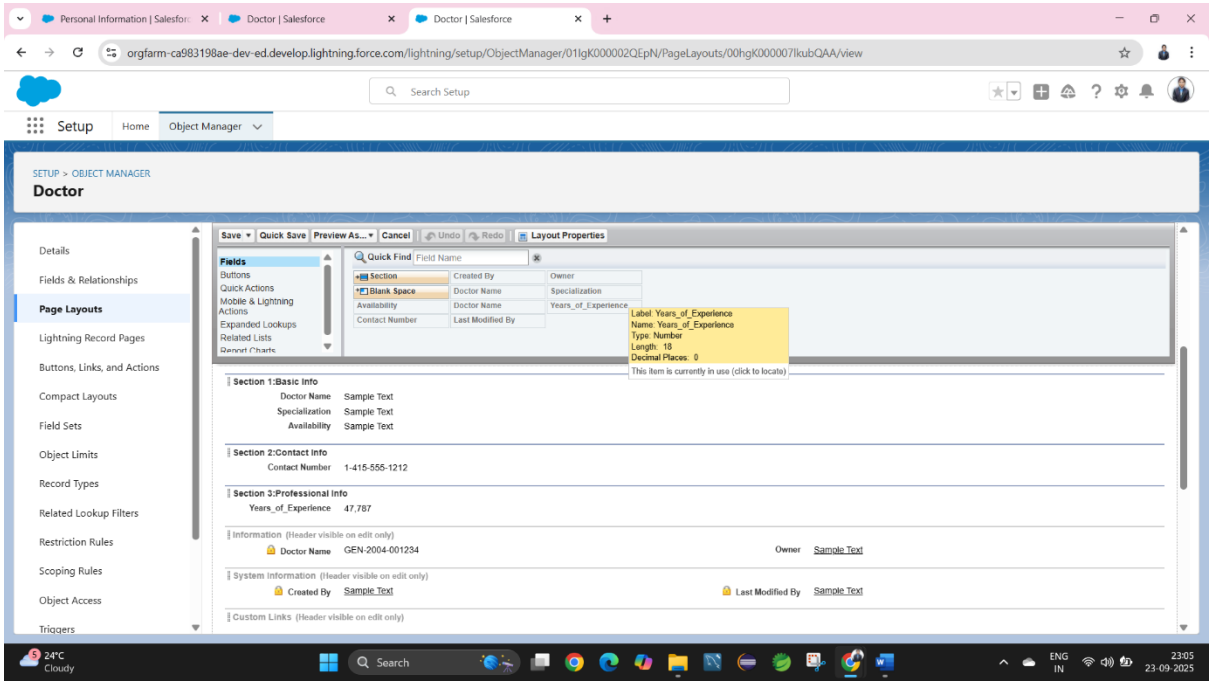
Expose or consume data between Salesforce and external systems using **REST or SOAP web services**.

Use Cases

- **REST**: Sending patient registration details from MediAid to an external government health registry.
- **SOAP**: Receiving medical insurance verification data.

Steps to Create REST Service

1. Create an Apex class annotated with @RestResource:
2. @RestResource(urlMapping='/PatientAPI/*')
3. global with sharing class PatientWebService {
4. @HttpGet
5. global static Patient__c getPatient() {
6. return [SELECT Name, Age__c FROM Patient__c LIMIT 1];
7. }
8. }
9. Access via URL: https://yourInstance.salesforce.com/services/apexrest/PatientAPI/



4. Callouts

Purpose

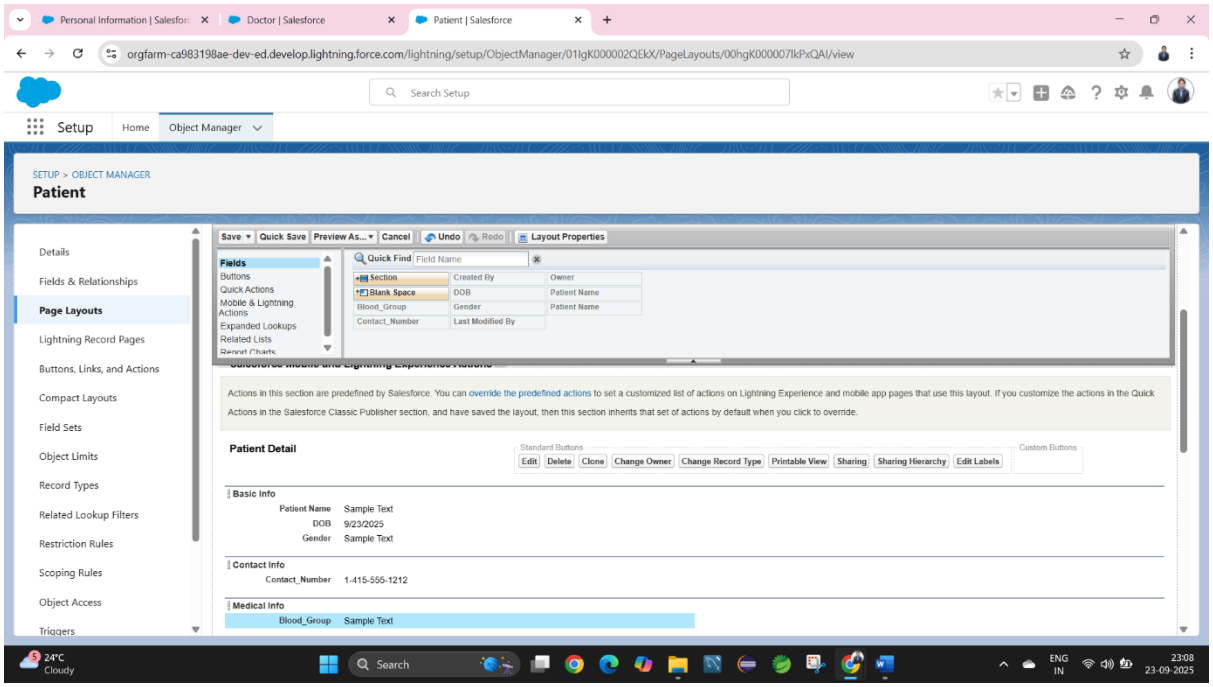
Allow MediAid to **send HTTP requests** to external systems (APIs).

Use Case

Sending **appointment details** to an external hospital booking system.

Steps

1. Add the API domain in **Setup → Remote Site Settings → New Remote Site**.
2. Create an Apex callout:
3. HttpRequest req = new HttpRequest();
4. req.setEndpoint('callout:HospitalAPI/appointments');
5. req.setMethod('POST');
6. req.setBody(JSON.serialize(appointmentData));
7. HttpResponse res = new Http().send(req);
8. Handle response for success or failure notifications in the UI.



5. Platform Events

Purpose

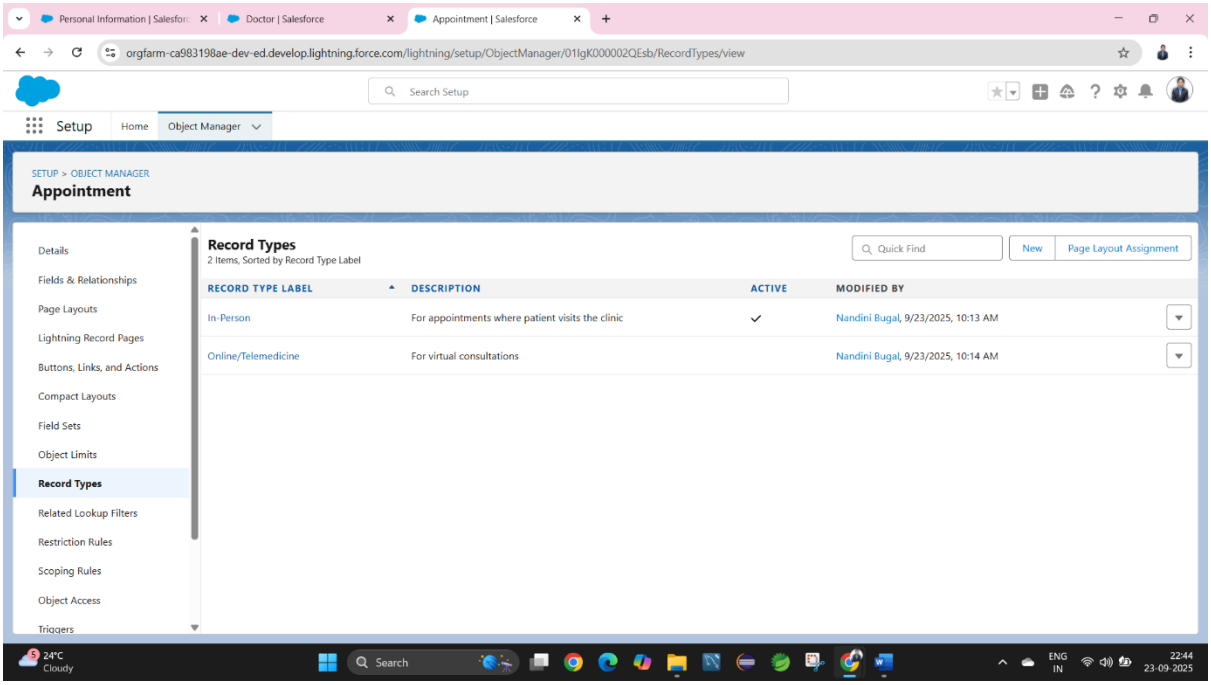
Enable **real-time communication** between MediAid and other systems (like alerting a pharmacy or lab when a new prescription is added).

Steps

1. Go to **Setup → Platform Events → New Platform Event**.
2. Create **Prescription_Alert__e** with fields:
    - Patient__c (Lookup)
    - Medicine__c (Text)

- Status__c (Text)

3. Publish event in Apex:
4. Prescription_Alert__e event = new Prescription_Alert__e(Patient__c = p.Id, Status__c = 'New');
5. Database.SaveResult sr = EventBus.publish(event);
6. Subscribe via **Trigger or External App** to act when the event fires.



6. Change Data Capture (CDC)

Purpose

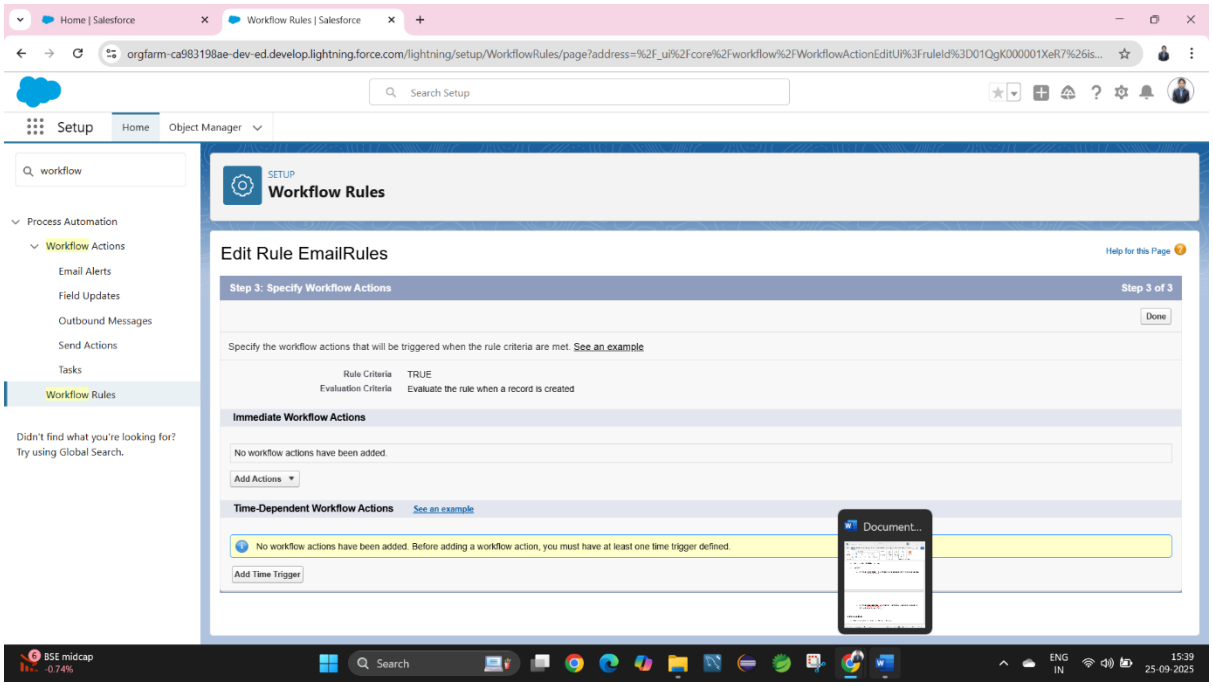
Track real-time data changes (Create, Update, Delete) in Salesforce objects and notify external systems.

Use Case

When a **Patient record** is updated in MediAid, notify an external **Analytics System**.

Steps

1. Go to **Setup → Change Data Capture**.
2. Select objects like **Patient__c**, **Appointment__c**, **Prescription__c**.
3. Subscribe using **CometD client** or **External App** (e.g., Java/Node.js) to listen for updates.
4. The app receives data changes automatically as events.



7. Salesforce Connect

Purpose

Access external data **in real-time without storing it in Salesforce** — useful for connecting MediAid with external hospital or lab databases.

Steps

1. Go to **Setup → External Data Source → New External Data Source**.
2. Choose **OData 4.0** as type.
3. Enter **Endpoint URL** (e.g., https://external-lab.com/odata).
4. Validate and sync tables → Creates **External Objects** (e.g., Lab_Reports__x).
5. View and use them like normal Salesforce objects.

8. API Limits

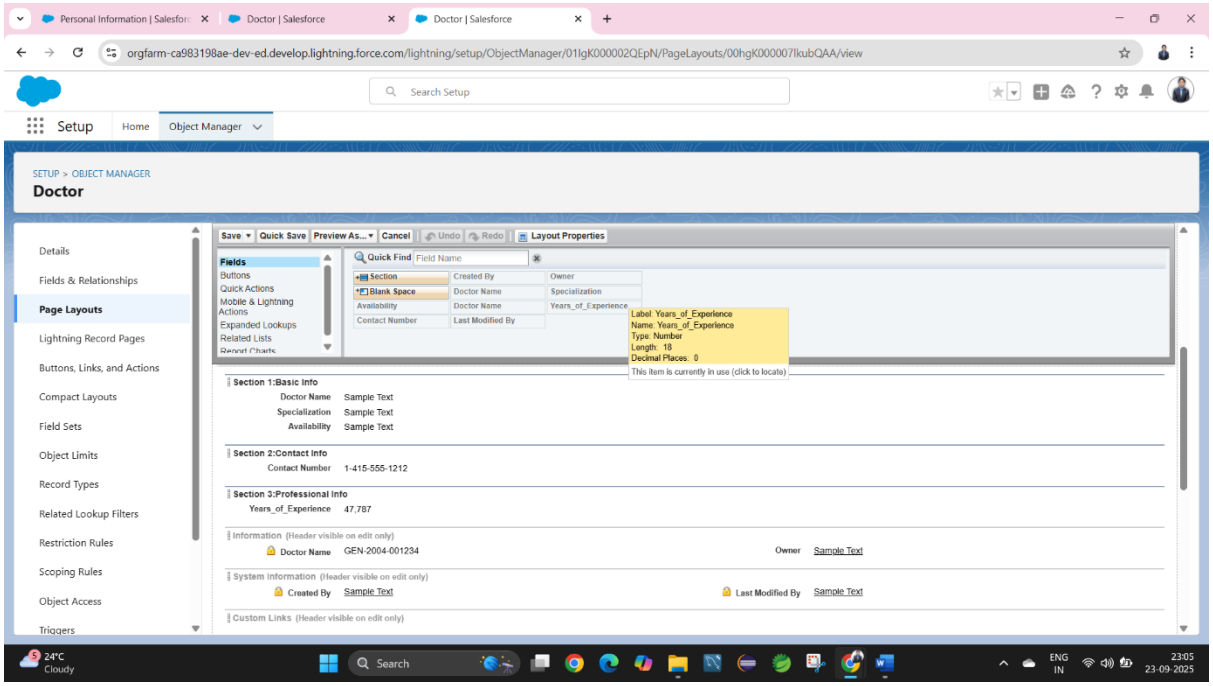
Purpose

Monitor and manage API usage to prevent exceeding Salesforce limits during integrations.

Steps

1. Go to **Setup → System Overview** → view **API Requests (Last 24 Hours)**.
2. Use **Developer Console → Query:**

3. SELECT Name, TotalRequests, Remaining FROM OrganizationLimit WHERE Name = 'DailyApiRequests'
4. Optimize integrations using **batch requests**, **caching**, and **asynchronous Apex**.



9. OAuth & Authentication

Purpose

Secure external integrations with proper authentication (login via OAuth 2.0 tokens).

Use Case

Allow patients to log into MediAid using **Google or Health ID** securely.

Steps

1. Go to **Setup → Auth. Providers → New**.
2. Select **Provider Type = Google**.
3. Fill Client ID and Secret from Google Cloud Console.
4. Create **Connected App** in Salesforce → Enable OAuth Scopes:
    - Access and manage your data (api)
    - Perform requests on your behalf (refresh_token, offline_access)
5. Test via generated login URL.

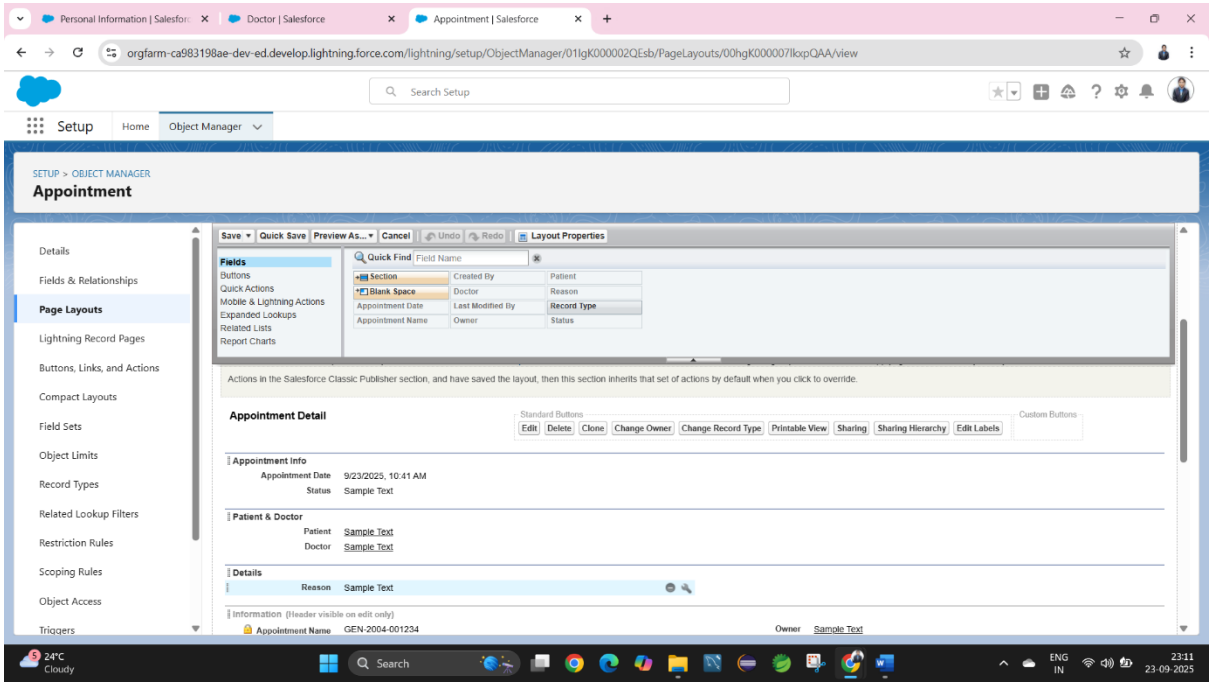
10. Remote Site Settings

Purpose

Authorize external endpoints used for callouts.

Steps

1. Go to **Setup → Remote Site Settings → New Remote Site**.
2. Enter:
    - **Label:** Pharmacy_API
    - **URL:** https://api.pharmacydata.com
    - **Active:** ☑
3. Save — this allows Apex callouts to that domain.



Phase 8: Data Management & Deployment – MediAid Project

Objective:

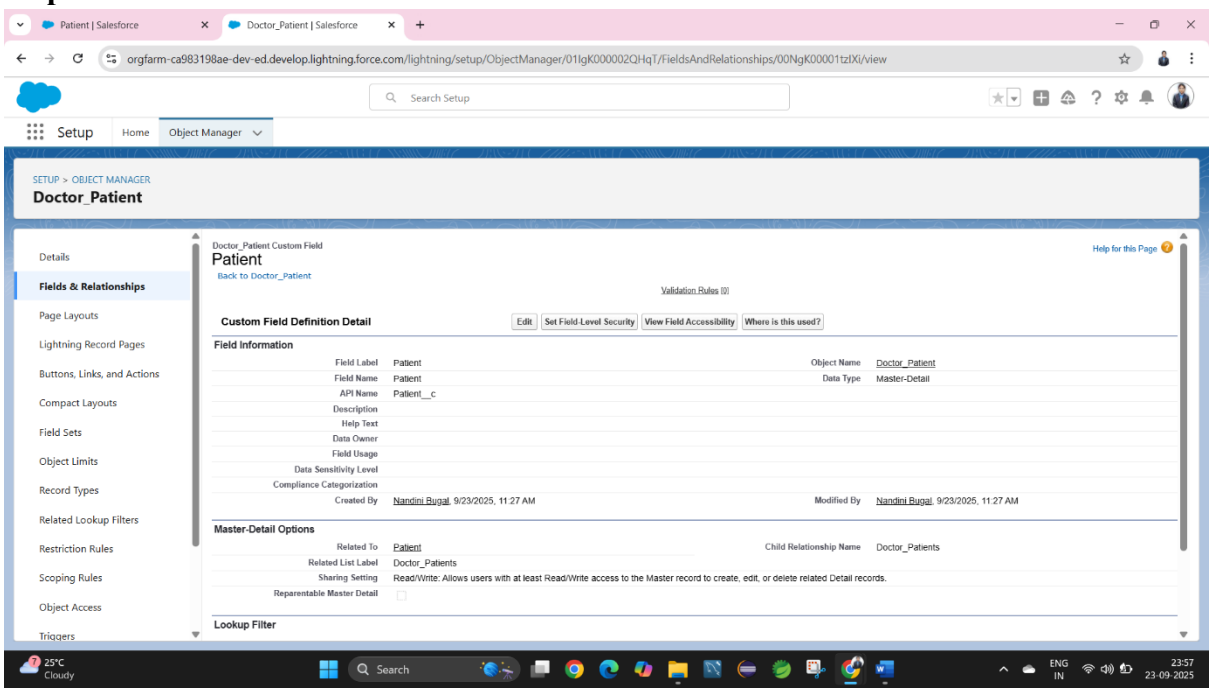
To effectively manage MediAid's healthcare data and ensure smooth deployment of customizations across different Salesforce environments while maintaining data security, consistency, and integrity.

1. Data Import Wizard

Purpose:

Used to quickly import data like **Patient records, Doctor details, and Appointments** into Salesforce using a simple interface.

Steps:



1. Go to **Setup → Data Import Wizard**.
2. Choose the object (e.g., **Patient__c**, **Doctor__c**, or **Appointment__c**).
3. Upload the **CSV file** containing data.
4. Map the CSV fields to Salesforce fields.
5. Click **Start Import**.
6. View import status in the background process.

Result:

Patient and doctor records are successfully imported into the MediAid database.

2. Data Loader

Purpose:

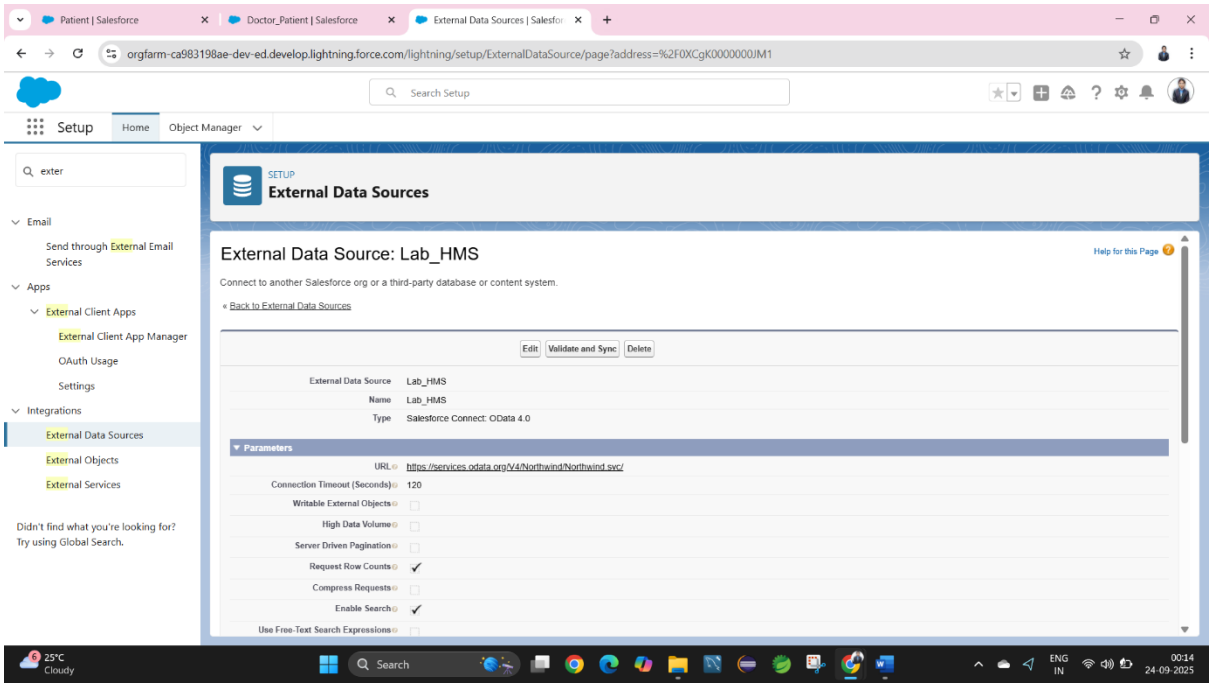
Used to perform **bulk data operations** like insert, update, delete, or export for large datasets.

Steps:

1. Download and open the **Salesforce Data Loader** tool.
2. Log in with Salesforce credentials.
3. Choose an operation (e.g., **Insert**, **Update**, **Delete**, **Export**).
4. Select the **object** and upload the CSV file.
5. Map fields and click **Finish** to process.
6. Review the **success and error logs**.

Result:

Bulk data like appointments and prescriptions are efficiently updated or exported.



3. Duplicate Rules

Purpose:

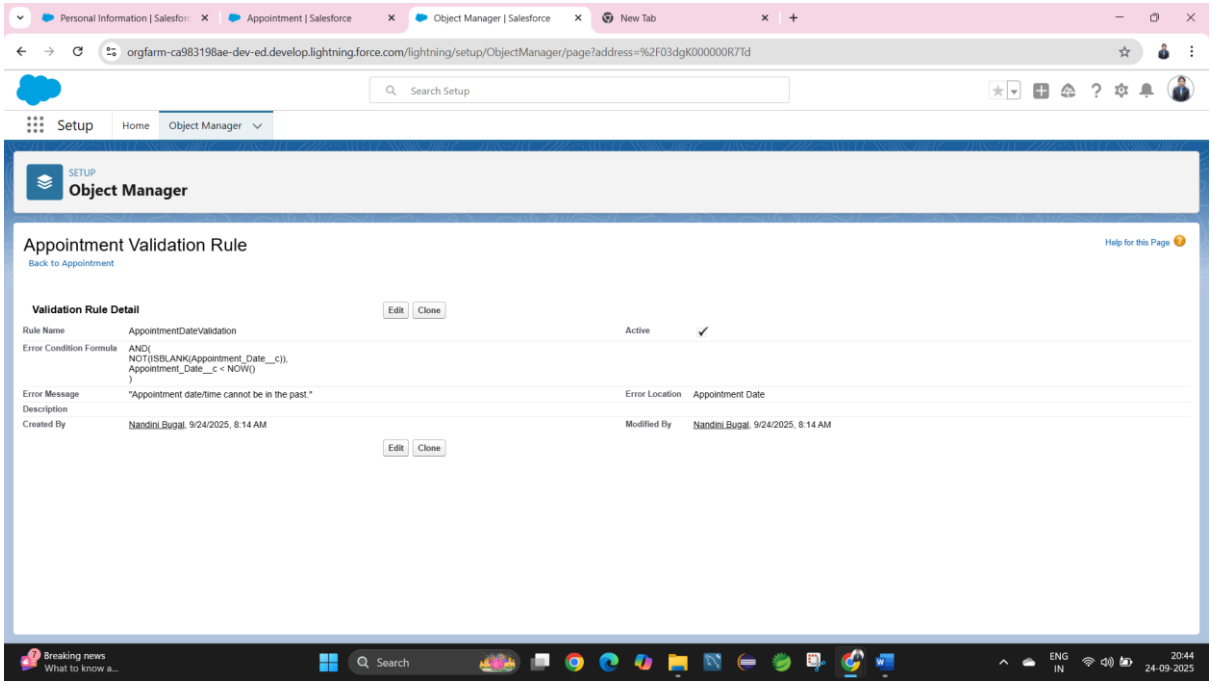
Prevent duplicate records for patients, doctors, or appointments to maintain clean and accurate data.

Steps:

1. Go to **Setup → Duplicate Rules**.
2. Click **New Rule** → Select **Patient__c** object.
3. Define matching criteria (e.g., Name, Phone, Aadhar).
4. Choose action (**Block** or **Allow with Alert**).
5. Activate the rule.

Result:

Duplicate patient or doctor records are prevented during data entry or import.



4. Data Export & Backup

Purpose:

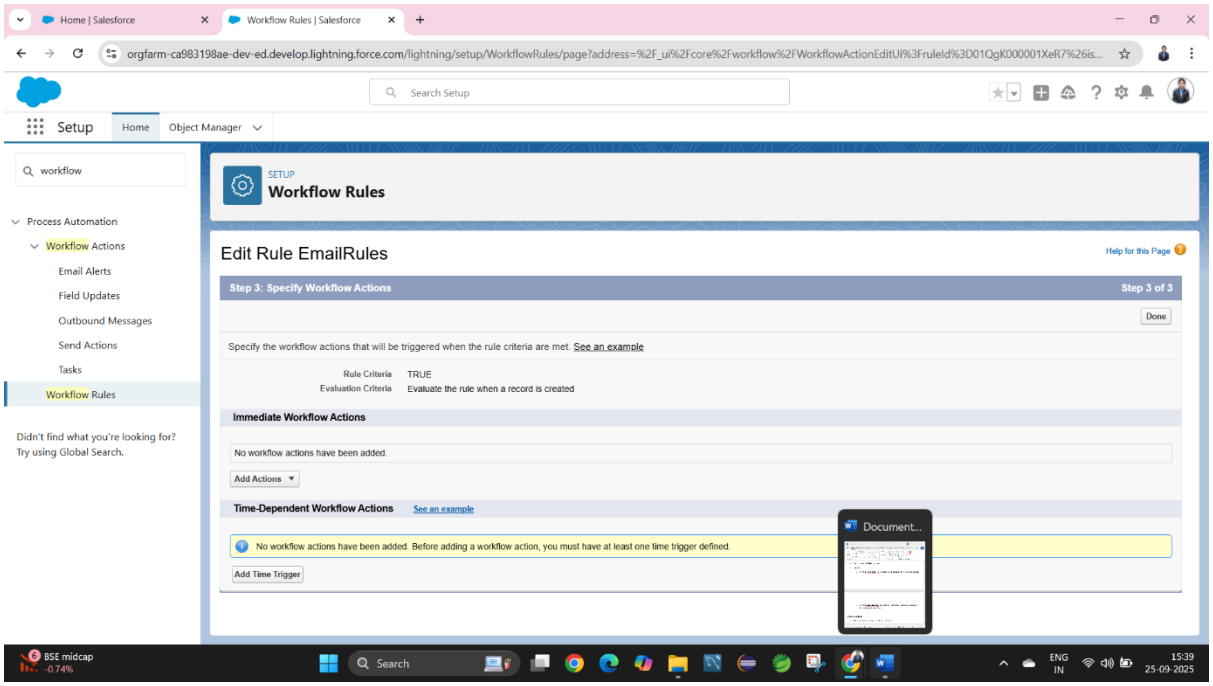
To regularly back up all MediAid data for recovery and security.

Steps:

1. Go to **Setup → Data Export**.
2. Click **Schedule Export**.
3. Choose the objects (e.g., Patient, Appointment, Prescription).
4. Select export frequency (Weekly/Monthly).
5. Receive a downloadable backup link via email.

Result:

A complete data backup of MediAid is generated for safe storage.



5. Change Sets

Purpose:

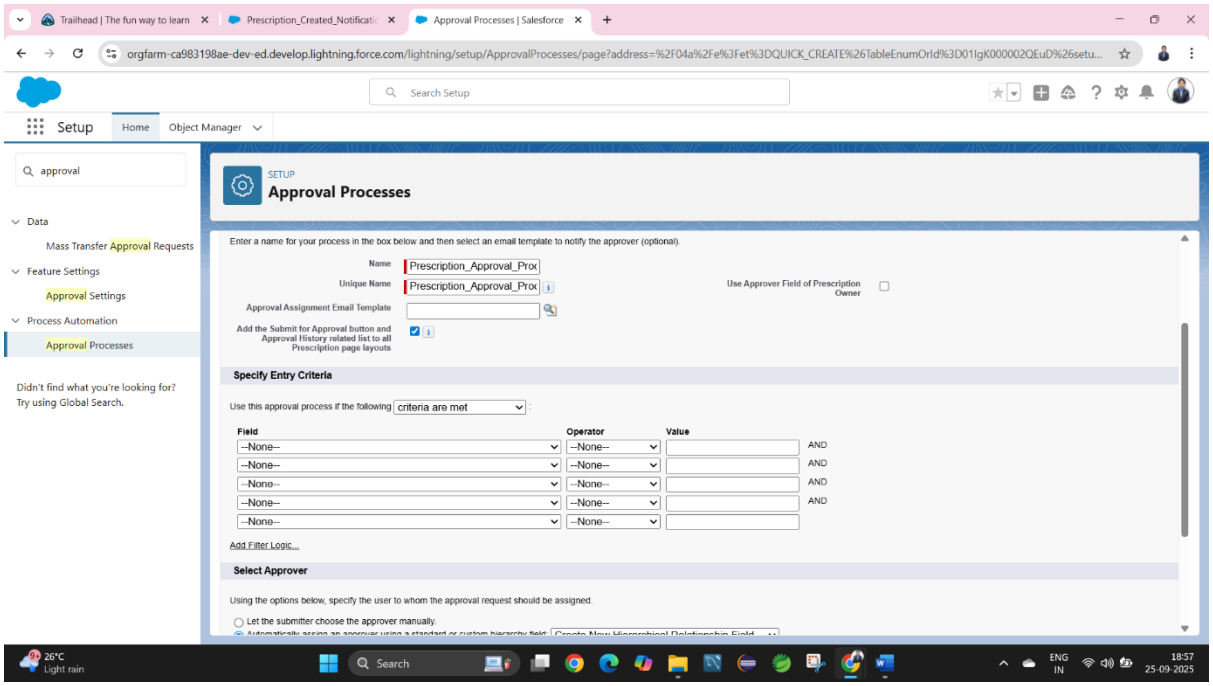
Used for deploying tested MediAid components from **Sandbox to Production**.

Steps:

1. Go to **Setup → Outbound Change Sets → New**.
2. Add components like **Apex Classes, Objects, LWC, and Triggers**.
3. Upload the change set to the **Production org**.
4. In Production, go to **Inbound Change Sets → Deploy**.
5. Validate and deploy successfully.

Result:

All new features and updates of MediAid are safely moved to the live environment.



6. Unmanaged vs Managed Packages

Purpose:

Used for packaging and distributing MediAid components.

Explanation:

- **Unmanaged Packages:** Used for internal development and testing; allows component modification.
- **Managed Packages:** Used for final deployment or distribution to other healthcare orgs; components are locked and versioned.

Result:

MediAid components are properly packaged and shared according to deployment needs.

7. ANT Migration Tool

Purpose:

Automates the deployment of metadata between MediAid's orgs using command-line scripts.

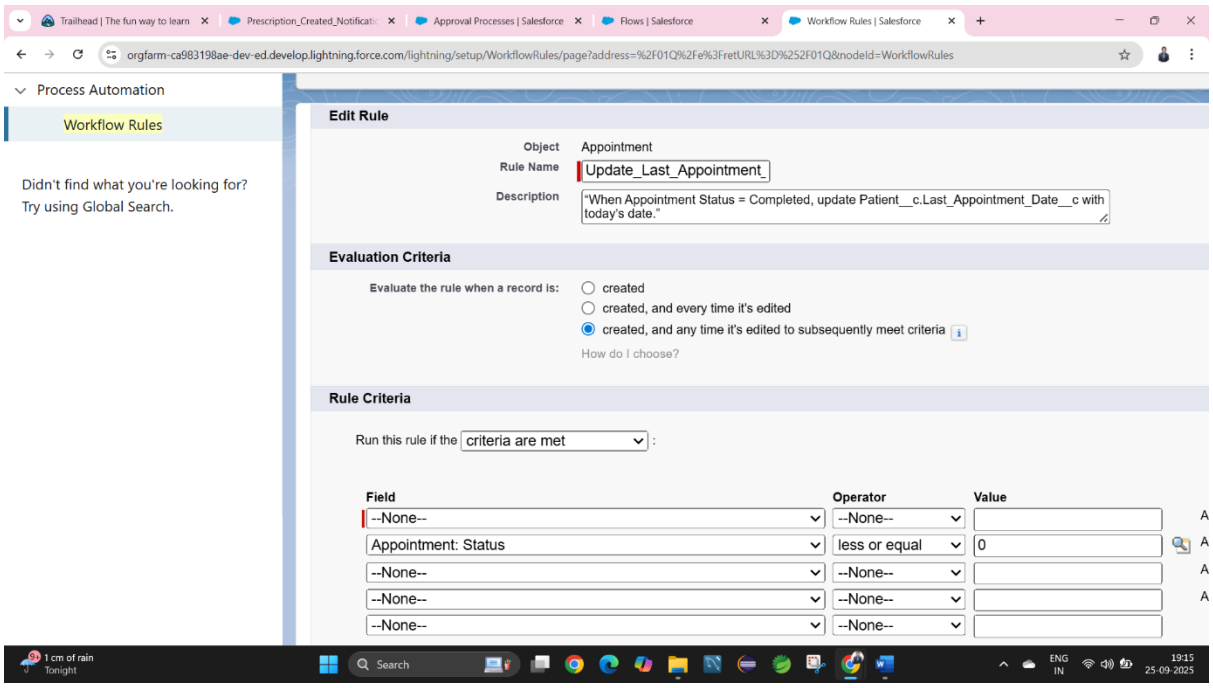
Steps:

1. Install the **Salesforce ANT Migration Tool**.
2. Create a **build.xml** and **package.xml** file listing the components.
3. Run the deployment command:
4. ant deployCode

5. Monitor console output for success or errors.

Result:

All metadata and components of MediAid are deployed efficiently and automatically.



8. VS Code & SFDX

Purpose:

Used as the main development and deployment environment for MediAid.

Steps:

1. Install **VS Code** and **Salesforce CLI (SFDX)**.
2. Connect to the Salesforce org:
3. sfdx force:auth:web:login -a MediAidOrg
4. Pull metadata from org:
5. sfdx force:source:retrieve -p force-app
6. Deploy changes:

Result:

MediAid's codebase is synchronized, version-controlled, and deployed through a modern development workflow.

Phase 9: Reporting, Dashboards & Security Review – MediAid Project

Objective:

To create insightful **reports and dashboards** for MediAid's healthcare data and perform a **security review** to ensure that all sensitive information such as patient and doctor records are well protected and accessible only to authorized users.

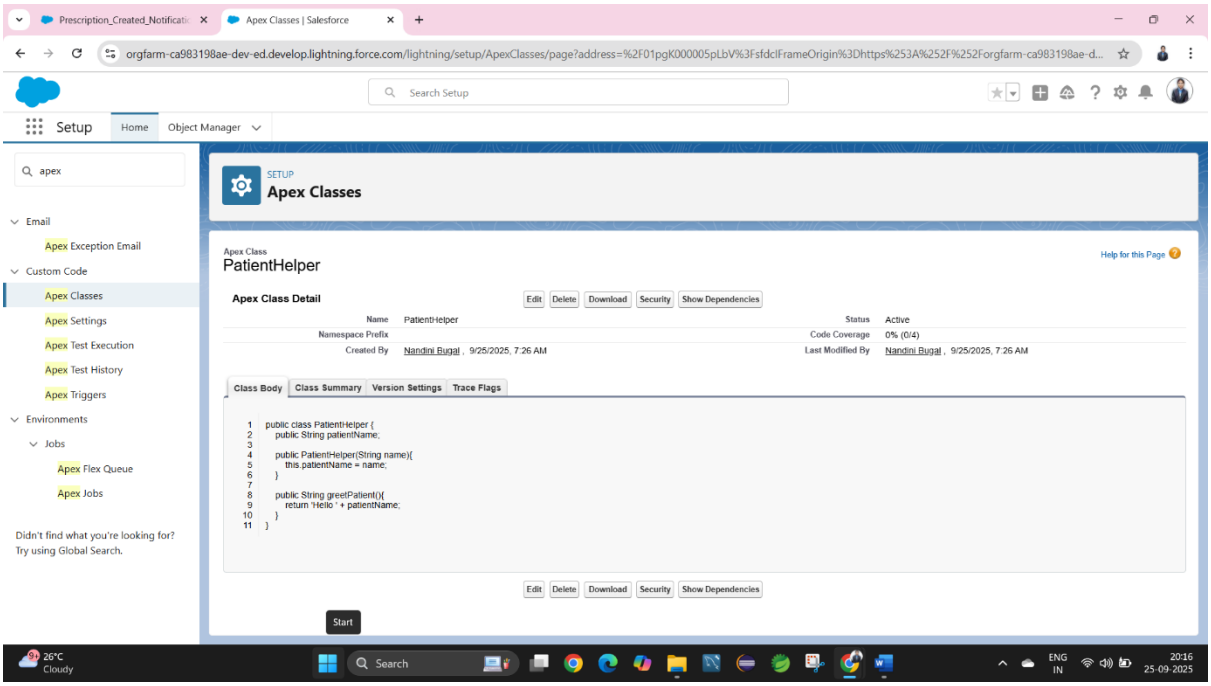
1. Reports (Tabular, Summary, Matrix, Joined)

Purpose:

To analyze key MediAid data like patient registrations, appointment trends, and doctor activity using different report formats.

Steps:

1. Navigate to **App Launcher → Reports → New Report**.
2. Select the report type (e.g., **Patients with Appointments**).
3. Choose the **format**:



   - **Tabular:** Simple list of patient details.
   - **Summary:** Grouped by doctor or department.
   - **Matrix:** Compare data (e.g., doctor vs. appointment count).
   - **Joined:** Combine patient and billing reports.
4. Add filters and fields.

5. Click **Save & Run Report**.

Result:

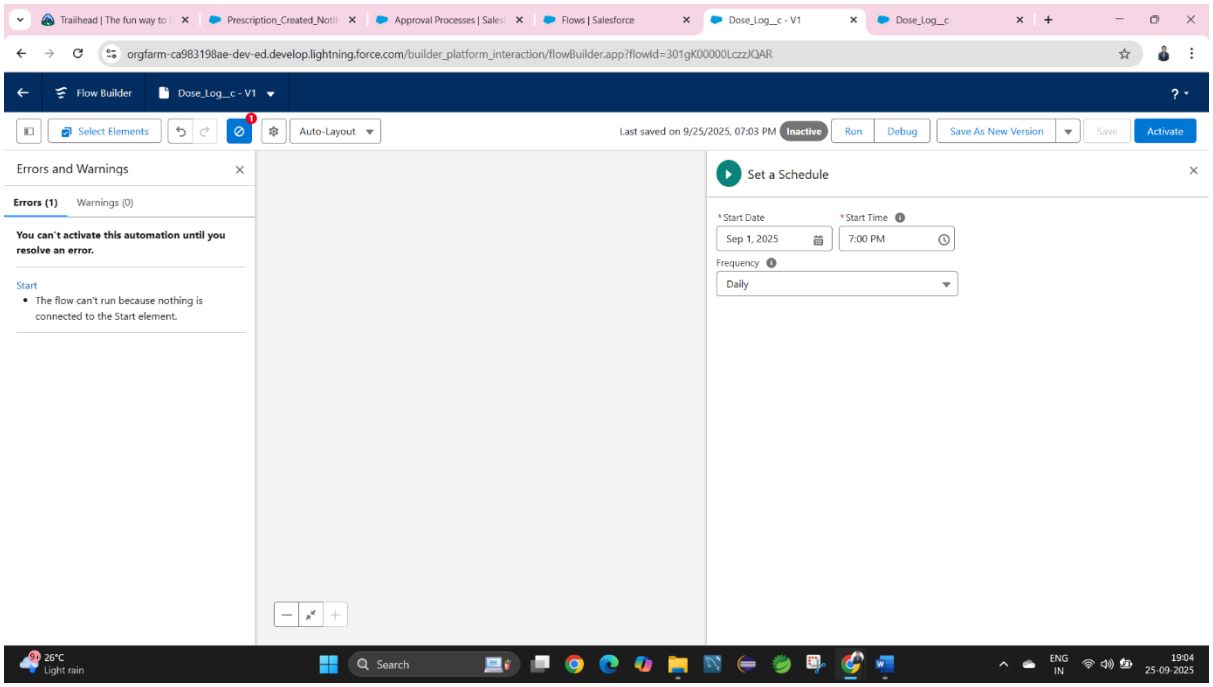
Different types of reports display MediAid's operational metrics clearly for analysis and decision-making.

2. Report Types

Purpose:

To define custom relationships between MediAid objects for generating combined reports (e.g., Patients with their Prescriptions).

Steps:



1. Go to **Setup → Report Types → New Custom Report Type**.
2. Choose **Primary Object:** Patient__c.
3. Choose **Related Object:** Appointment__c or Prescription__c.
4. Set relationship and deployment status.
5. Save and use it in the report builder.

Result:

Custom reports can now show linked data, such as which patient received which treatment or doctor's visit.

3. Dashboards

Purpose:

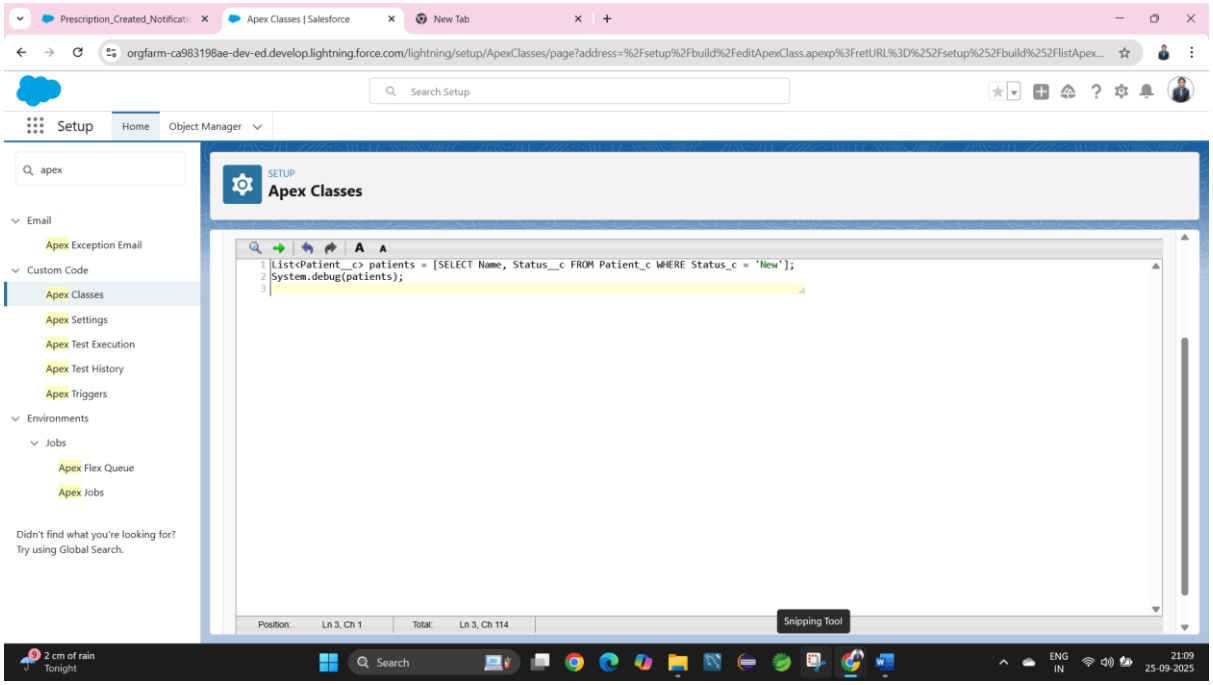
To visualize MediAid's reports using charts, metrics, and graphs for better decision support.

Steps:

1. Navigate to **App Launcher → Dashboards → New Dashboard**.
2. Add components (Charts, Gauges, Tables).
3. Link each component to a **report** (e.g., Appointment Count by Doctor).
4. Choose display options and save.

Result:

An interactive MediAid dashboard displays hospital KPIs like total patients, daily appointments, and feedback trends.



4. Dynamic Dashboards

Purpose:

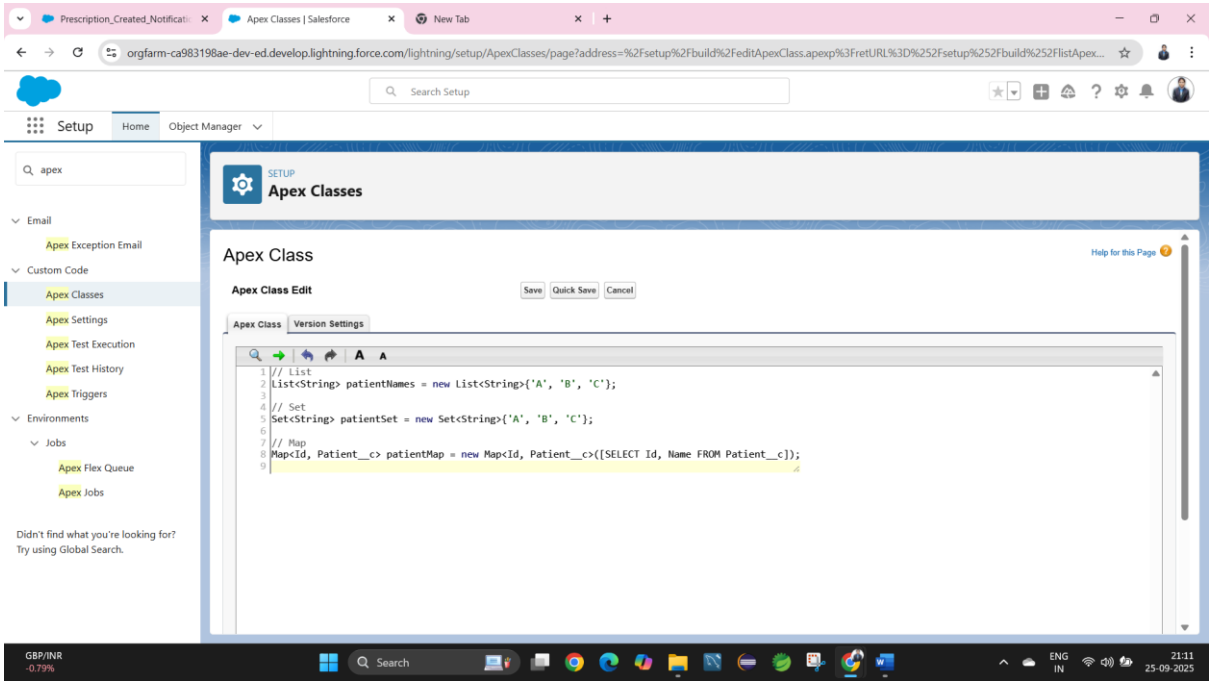
To allow dashboards to display data specific to the logged-in user (e.g., a doctor sees only their appointments).

Steps:

1. Edit an existing dashboard.
2. Under **View Dashboard As**, select **"The logged-in user."**
3. Save and assign dashboard permissions.

Result:

Dynamic dashboards give each MediAid user a personalized data view, maintaining privacy and role-based access.



5. Sharing Settings

Purpose:

To control record-level access across MediAid users (e.g., Admins, Doctors, Receptionists).

Steps:

1. Go to **Setup → Sharing Settings**.
2. Set **Organization-Wide Defaults (OWD)**:
    - Patients: Private
    - Appointments: Controlled by Parent
3. Define **Sharing Rules** (e.g., share Patient records with assigned Doctor).
4. Save the configuration.

Result:

Only authorized users can view or edit sensitive MediAid data, ensuring privacy and compliance.

6. Field Level Security (FLS)

Purpose:

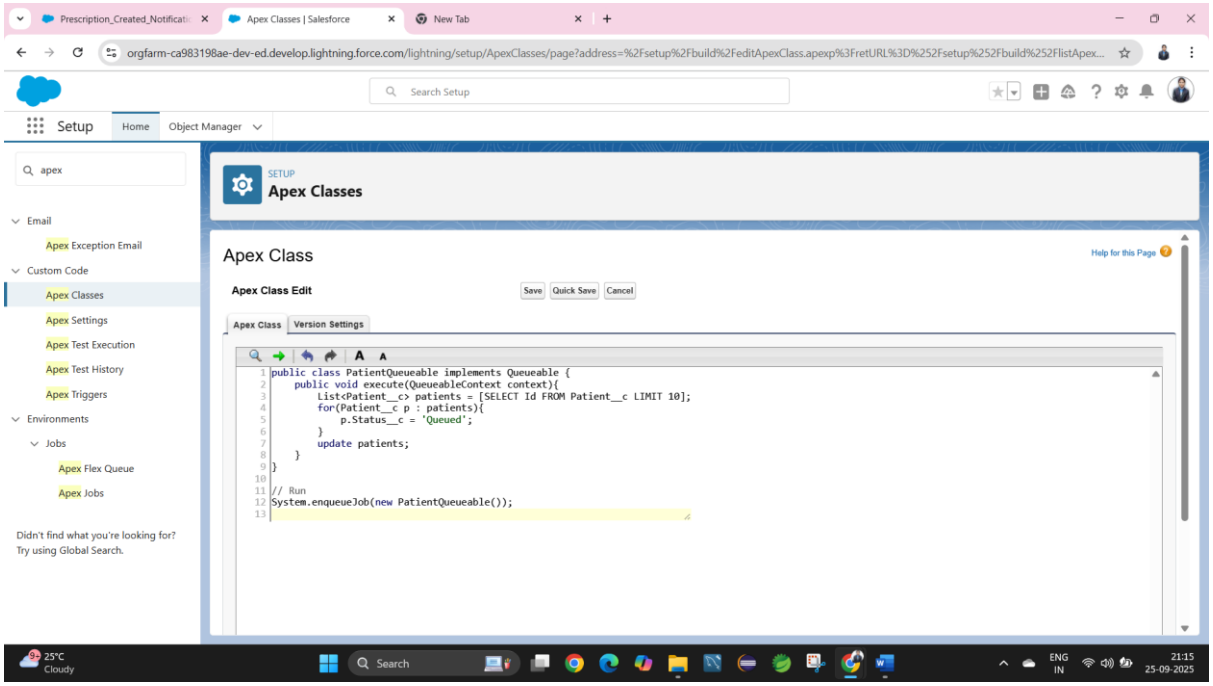
To restrict visibility of specific fields (e.g., Aadhar, Medical History) for non-admin users.

Steps:

1. Go to **Setup → Object Manager → Patient__c → Fields & Relationships.**
2. Click a field (e.g., Aadhar_Number).
3. Set **Field-Level Security** → Uncheck visibility for unauthorized profiles.
4. Save.

Result:

Sensitive fields remain hidden from non-privileged users such as receptionists.



7. Session Settings

Purpose:

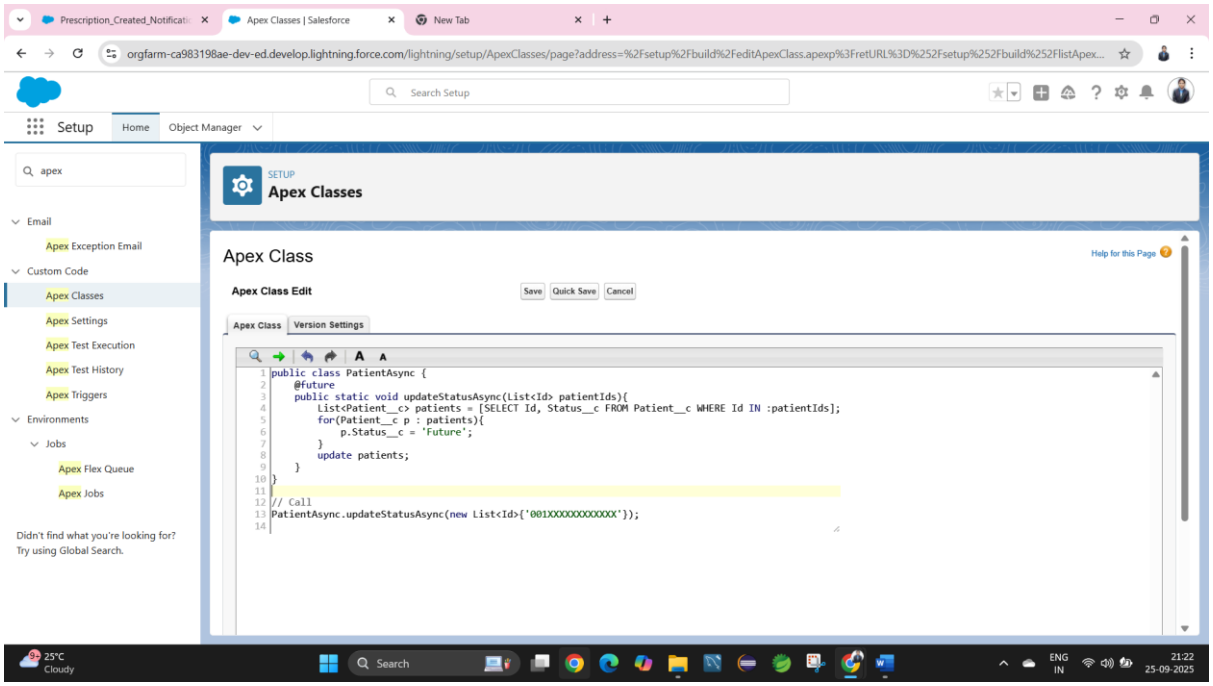
To manage user session timeouts and security for MediAid users.

Steps:

1. Go to **Setup → Session Settings**.
2. Define session timeout duration (e.g., 30 minutes).
3. Enable **HTTPS** and **IP Restriction** options.
4. Save the settings.

Result:

Inactive sessions automatically expire, preventing unauthorized access to MediAid data.



8. Login IP Ranges

Purpose:

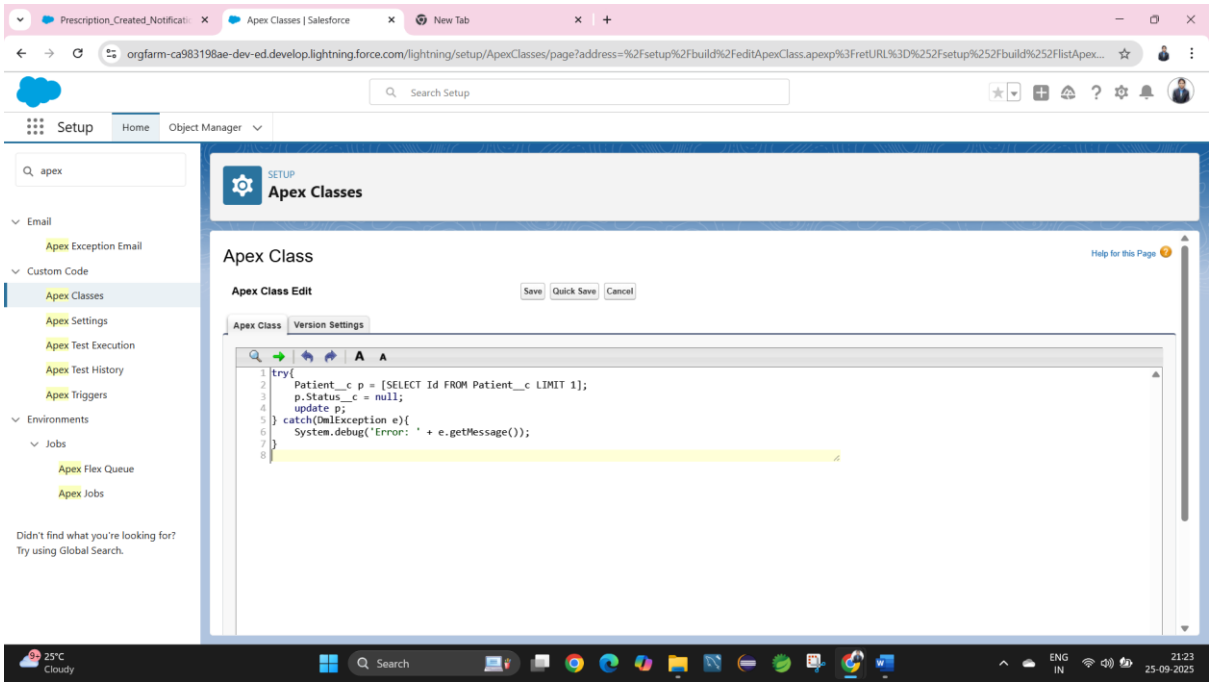
To allow user logins only from authorized hospital or clinic networks.

Steps:

1. Go to **Setup → Profiles → Select a Profile (e.g., Doctor Profile).**
2. In the **Login IP Ranges** section, click **New**.
3. Enter the IP range for the clinic or hospital network.
4. Save.

Result:

Unauthorized access attempts from unknown networks are blocked.



9. Audit Trail

Purpose:

To track all changes made to configuration or data for accountability.

Steps:

1. Go to **Setup → View Setup Audit Trail**.
2. Review the list of changes made (user, action, and date).
3. Download the log if needed.

Result:

All configuration and user actions in MediAid are logged for security

Phase 10: Final Presentation & Demo Day

1. Pitch Presentation

Goal: Convince evaluators and audience that MediAid solves a real healthcare problem effectively.

Slide-by-Slide Flow

Slide	Title	Content to Cover
1	**Title Slide**	*MediAid – Smart Medicine Reminder App* Team Name, Members, Institution, Guide Name
2	**Problem Statement**	Talk about the real-world issue of **missed medication doses**, especially for elderly or chronically ill patients.
3	**Proposed Solution**	Introduce **MediAid**, explain how it automates reminders, tracks doses, and connects patients, caregivers, and doctors.
4	**Objectives**	List key goals: medication adherence, smart scheduling, health tracking, and notifications.
5	**System Architecture**	Show the **Salesforce-based architecture** (Custom Objects, Data Model, and Process Flow).
6	**Key Features**	• Patient Registration • Medicine Reminder Scheduler • Notification Alerts • Appointment Management • Doctor-Patient Connect
7	**Technology Stack**	Salesforce Platform, Lightning App Builder, Apex Classes, Flows, Process Builder, Reports, and Dashboards.
8	**Implementation Snapshots**	Screenshots of record pages, components, and sample reminders.
9	**Impact & Benefits**	Explain how MediAid helps patients stay healthy, reduces hospital visits, and supports caregivers.
10	**Future Enhancements**	• AI-based dosage predictions • Mobile App integration • Cloud-based analytics for hospitals.
11	**Conclusion**	Summarize the journey and success of MediAid. Thank audience and invite questions.

2. Demo Walkthrough (Live or Recorded)

Duration: ~7–8 minutes

Goal: Show the project's functionality in action.

Step-by-Step Demo Flow

1. **Start with Login / Home Page**
    - Show MediAid App's main interface (Salesforce Lightning App).
    - Introduce navigation tabs (Patients, Appointments, Medicines, Reminders).
2. **Patient Registration (Custom Object: Patient__c)**
    - Demonstrate adding a new patient with all details.
    - Explain validation rules or automated field fills (if used).
3. **Medicine Scheduling (Custom Object: Medicine__c / Reminder__c)**
    - Add a new medicine and set dosage time.
    - Show how reminders are triggered or listed.
4. **Appointment Management**
    - Create and view an appointment record.
    - Show status updates or relations with Doctor__c object.
5. **Notification Flow (Automation Demo)**
    - Explain how the **Process Builder or Flow** sends reminders or notifications.
6. **Reports & Dashboard**
    - Show charts for medicine adherence, active patients, or appointments overview.
7. **End Summary**
    - Briefly recap the workflow from registration → reminder → reporting.

Optional: Record this demo using screen recorder (OBS or Loom) and narrate each step.

3. Feedback Collection

Goal: Get constructive input from evaluators, mentors, or users.

Methods:

- Use a short **Google Form** or **Salesforce Survey**:
    - UI/UX rating
    - Feature usefulness
    - Improvement suggestions
- Collect qualitative feedback (verbal + written).

Example Form Sections:

- Name / Role
- What feature did you find most useful?
- Any additional features you'd suggest?
- Rate MediAid on usability (1–5)

4. Handoff Documentation

Goal: Make your project reusable or extendable by others.

Deliverables:

- **Project Summary Report**
- **Technical Documentation**
  - Custom Objects (Patient__c, Medicine__c, Reminder__c)
  - Relationships (Lookup/Master-Detail)
  - Flow / Process Builder Diagrams
- **Demo Script or User Manual**
  - Step-by-step usage guide
- **Login Credentials (if demo org used)**
  - For reviewer testing

5. LinkedIn / Portfolio Project Showcase

Goal: Highlight your work professionally.

Steps:

1. Create a **LinkedIn Post**:
   - Title: *MediAid – Smart Medicine Reminder App | Salesforce-based Healthcare Solution*
   - Describe the project briefly (Problem + Solution + Tech + Impact)
   - Add screenshots or your demo video link.
2. Add to **Portfolio / Resume**:
   - Project Name: MediAid – Smart Medicine Reminder App
   - Tools: Salesforce Lightning, Apex, Flow Builder
   - Description: Developed a healthcare app to manage medication reminders and appointments using Salesforce automation tools.
3. Tag teammates, mentors, and Salesforce-related hashtags:
   #Salesforce #HealthcareTech #Innovation #MediAid #DemoDay