

SALESFORCE PROJECT IMPLEMENTATION PHASES

Problem Statement: Telemedicine Access for Rural Healthcare in Nabha

Description: Nabha and its surrounding rural areas face significant healthcare challenges. The local Civil Hospital operates at less than 50% staff capacity, with only 11 doctors for 23 sanctioned posts. Patients from 173 villages travel long distances, often missing work, only to find that specialists are unavailable or medicines are out of stock. Poor road conditions and sanitation further hinder access. Many residents lack timely medical care, leading to worsened health outcomes and increased financial strain.

Impact / Why this problem needs to be solved:

This problem directly affects the health and livelihood of thousands of rural residents, especially daily-wage earners and farmers. Lack of accessible healthcare leads to preventable complications, financial losses, and overall decline in community well-being. Addressing this issue would improve healthcare delivery, reduce unnecessary travel, and enhance quality of life for a large, underserved population.

Phase 1: Problem Understanding & Industry Analysis

1. Requirement Gathering

Conducted requirement gathering based on the problem statement and rural healthcare needs.

Functional Requirements:

- Video consultations between patients and doctors.
- Digital health records accessible both online and offline.
- Real-time medicine availability updates from local pharmacies.
- AI-powered symptom checker optimized for low bandwidth.
- Appointment booking and notifications via SMS/WhatsApp.

Non-Functional Requirements:

- Multilingual support (Punjabi, Hindi, English).

- Offline-first mobile app for areas with poor connectivity.
- Secure data storage with role-based access.
- Scalable design for other rural regions.

2. Stakeholder Analysis

Stakeholder	Role in Project	Benefits
Patients (Rural Villagers)	Access healthcare without traveling long distances.	Saves time, money, ensures timely treatment.
Doctors	Provide consultations remotely.	Reach more patients, efficient scheduling.
Pharmacy Staff	Manage medicine availability and inventory.	Real-time updates for patients and doctors.
Civil Hospital Staff	Support patient registration and manage referrals.	Reduce hospital crowding, track patient flow.
Punjab Health Dept.	Oversee rural healthcare delivery.	Data-driven insights, performance monitoring.
Daily Wage Workers	Patients who lose income due to travel for healthcare.	Can continue earning without frequent hospital visits.

3. Business Process Mapping

High-level flow of the telemedicine process:

1. Patient logs into the community portal/mobile app.
2. Patient books appointment with available doctor.
3. System sends confirmation + reminders (SMS/Email).
4. Doctor conducts video consultation.
5. Prescription generated and stored in patient's digital record.
6. Pharmacy updates stock & dispenses medicine.
7. Notifications sent to patient for medicine pickup.
8. Health Dept. accesses dashboards for analytics & monitoring.

FlowChart--

[Patient Login / Registration --> Book Appointment --> System Sends Confirmation + Reminder --> Doctor Conducts Video Consultation --> Prescription Generated &

**Saved --> Pharmacy Updates Inventory --> Medicine Dispensed to Patient -->
Notifications Sent to Patient --> Health Dept. Reviews Analytics Dashboard]**

4. Industry-specific Use Case Analysis

Healthcare Gaps in Rural Areas:

- Lack of specialists and infrastructure.
- High travel cost for patients.
- Poor medicine availability tracking.

How Salesforce Solves This:

- **Health Cloud:** Manage patient data, history, and care plans.
- **Community Cloud:** Patients access portal in their language.

- **Service Cloud:** Appointment booking, reminders, case management.
- **Einstein Bots & AI:** Symptom checker for quick triage.
- **Integration:** Pharmacy systems, government health databases.

5. AppExchange Exploration

To avoid reinventing the wheel, I explored Salesforce **AppExchange** solutions relevant to healthcare:

Health Cloud Packages → Prebuilt modules for patient management.

Telemedicine Connectors → Zoom/Teams integration for video calls.

Pharmacy Management Apps → Medicine inventory tracking.

SMS/WhatsApp Notification Apps → Twilio, ValueText.

SALESFORCE PROJECT IMPLEMENTATION PHASES

Phase 2: Org Setup & Configuration

In this phase, we configure the Salesforce Org to align with the Telemedicine Access for Rural Healthcare in Nabha project. Proper Org setup ensures smooth functionality, data security, role-based access, and scalability for future needs.

1. Salesforce Editions

Salesforce has different editions: Essentials, Professional, Enterprise, Unlimited, and Developer.

- For this project, I am using “Salesforce Developer Edition”.

1. It gives us customization features.
2. It supports Apex, LWC, Integrations, and APIs.

- Developer Edition provides advanced customization, role hierarchy, profiles, API access, and integration support.

- It allows us to build custom apps for “Patients, Doctors, Pharmacy Staff, and Health Department Officers.”

2. Company Profile Setup

Path: Setup → Company Information

- Company Name: Telemedicine Rural Healthcare.

- Default Locale: English (India)

- Default Time Zone: Asia/Kolkata (GMT +5:30)

- Default Currency: INR ₹ (Indian Rupees)

- Corporate Currency: INR ₹ (All records display INR as default)

The screenshot shows the 'Company Information' page in the Salesforce setup. The top navigation bar includes 'SETUP' and 'Company Information'. The main title is 'Company Information' followed by the organization name 'Telemedicine Rural Healthcare – Nabha'. A note says 'The organization's profile is below.' Below the title, there are tabs for 'User Licenses [10+]', 'Permission Set Licenses [10+]', 'Feature Licenses [11]', and 'Usage-based Entitlements [10+]'. The 'Organization Detail' section contains fields for Organization Name (Telemedicine Rural Healthcare – Nabha), Primary Contact (OrgFarm EPIC), Division (United States), Address (January), Fiscal Year Starts In (January), Activate Multiple Currencies (unchecked), Enable Data Translation (unchecked), Newsletter (checked), Admin Newsletter (checked), Hide Notices About System Maintenance (unchecked), Hide Notices About System Downtime (unchecked), and Locale Formats (ICU). To the right, the 'Phone' section lists fields for Phone, Fax, Default Locale (English (United States)), Default Language (English), Default Time Zone (GMT-07:00 Pacific Daylight Time (America/Los_Angeles)), Currency Locale (Hindi (India) - INR), Used Data Space (462 KB (9%) [View]), Used File Space (17 KB (0%) [View]), API Requests, Last 24 Hours (0 (15,000 max)), Streaming API Events, Last 24 Hours (0 (10,000 max)), Restricted Logins, Current Month (0 (0 max)), Salesforce.com Organization ID (00DgLO00007V3AT), Organization Edition (Developer Edition), and Instance (CAN98). At the bottom, it shows 'Created By' (OrgFarm EPIC, 7/18/2025, 12:19 AM) and 'Modified By' (Anjali Raikwar, 9/12/2025, 5:17 AM).

This setup ensures consistency across all users in the Org.

3. Business Hours & Holidays

Path: Setup → Company Settings → Business Hours

- Default Business Hours: 9 AM – 6 PM (Monday – Saturday)

- Custom Business Hours:

1. Teleconsultation Service Window: 7 AM – 9 PM
2. Pharmacy Support: 9 AM – 8 PM

Holidays Configured:

1. 15 August – Independence Day
2. 26 January – Republic Day
3. 2 October – Gandhi Jayanti
4. 1 January – New Year

The screenshot shows the 'Business Hours' page under 'Setup'. The title is 'Business Hours'. Below it, there's a section titled 'Organization Business Hours' with a note about selecting days and hours for support teams. A 'Holidays [0]' link is present. The main table shows 'Business Hours Detail' with columns for 'Business Hours Name' (set to 'Hospital Working Hours'), 'Time Zone' (set to '(GMT-07:00) Pacific Daylight Time (America/Los_Angeles)'), and 'Default Business Hours' (checkbox). The table lists days from Sunday to Saturday with their respective working hours. At the bottom, there are fields for 'Active' status, 'Created By' (Anjali Raikwar), and 'Last Modified By' (Anjali Raikwar).

This ensures that SLA calculations, case escalations, and automated workflows respect working/non-working hours.

5. Fiscal Year Settings

Path: Setup → Company Profile → Fiscal Year

- Fiscal Year set to “January” (Indian standard financial year).

- Standard fiscal year used (not custom).

The screenshot shows the 'Change Fiscal Year Period' page. It has fields for 'Name' (Telemedicine Rural Healthcare – Nabha), 'Fiscal Year Start Month' (set to 'January'), and 'Fiscal Year is Based On' (radio button selected for 'The ending month'). There are 'Save' and 'Cancel' buttons at the bottom.

6. User Setup & Licenses

We created multiple users to represent project stakeholders.

Users Created:

1. Doctor User

- Profile: Standard User
- License: Salesforce
- Role: Doctor
- Access: Can create/view patient records, schedule teleconsultations.

- **Name:** Dr. Amit Sharma
- **Email:** dr.amitsharma@telemedicine.com
- **Username:** doctor1@telemedicine.com
- **Role:** Doctor
- **User License:** Salesforce
- **Profile:** Doctor Profile
- **Business Hours:** Hospital Working Hours

The screenshot shows the User Detail page for 'Dr. Amit Sharma Sharma'. At the top, there are tabs for 'Edit', 'Sharing', 'Reset Password', 'Freeze', and 'View Summary'. Below the tabs, the user's name is displayed as 'Dr. Amit Sharma Sharma'. The page includes a navigation bar with links like 'Permission Set Assignments', 'Activation Required', 'Permission Set Group Assignments', 'Permission Set License Assignments', 'Personal Groups', 'Public Group Membership', 'Queue Membership', 'Team', 'Managers in the Role Hierarchy', 'OAuth Apps', 'Third-Party Account Links', 'Built-in Authenticators', 'Installed Mobile Apps', 'Authentication Settings for External Systems', 'Login History', and 'User Provisioning Accounts'. The main content area is titled 'User Detail' and contains sections for 'Name', 'Alias', 'Email', 'Username', 'Nickname', 'Title', 'Company', 'Department', 'Division', 'Address', 'Time Zone', 'Locale', 'Language', 'Delegated Approver', 'Manager', 'Receive Approval Request Emails', and 'Federation ID'. To the right of each field, there are 'Role' and 'User License' columns with checkboxes for various profiles like 'Doctor', 'Salesforce', 'Doctor Profile', etc. Some checkboxes are checked (e.g., 'Doctor' for Name, 'Active' for Username). Other sections include 'Mobile Push Registrations', 'Data.com User Type', 'Accessibility Mode (Classic Only)', 'Debug Mode', and 'High-Contrast Palette on Charts'.

2. Pharmacy Staff User

- Profile: Standard User
- License: Salesforce Platform
- Role: Pharmacy Staff
- Access: Manage medicine inventory, update stock, issue medicines.

- **Name:** Neha Gupta
- **Email:** neha.pharmacy@telemedicine.com
- **Username:** pharmacy1@telemedicine.com
- **Role:** Pharmacy Staff
- **User License:** Salesforce Platform
- **Profile:** Pharmacy Profile

User
Neha Gupta Gupta

[User Profile](#)[Help for this Page](#)

[Permission Set Assignments \(0\)](#) | [Permission Set Assignments: Activation Required \(0\)](#) | [Permission Set Group Assignments \(0\)](#) | [Permission Set License Assignments \(0\)](#) | [Personal Groups \(0\)](#) | [Public Group Membership \(0\)](#) | [Queue Membership \(0\)](#)
 | [Team \(0\)](#) | [Managers in the Role Hierarchy \(1\)](#) | [OAuth Apps \(0\)](#) | [Third-Party Account Links \(0\)](#) | [Built-In Authenticators \(0\)](#) | [Installed Mobile Apps \(0\)](#) | [Authentication Settings for External Systems \(0\)](#) | [Login History \(0+\)](#) | [User Provisioning Accounts \(0\)](#)

User Detail		Edit	Sharing	Reset Password	Freeze	View Summary
Name	Neha Gupta Gupta					Role Pharmacy_Staff
Alias	ngupt					User License Salesforce Platform
Email	neha.pharmacy@telemedicine.com [Verify]					Profile Standard Platform User
Username	pharmacy1@telemedicine.com					Active <input checked="" type="checkbox"/>
Nickname	User17576792249357097440					Marketing User <input type="checkbox"/>
Title						Offline User <input type="checkbox"/>
Company						Knowledge User <input type="checkbox"/>
Department						Flow User <input type="checkbox"/>
Division						Service Cloud User <input type="checkbox"/>
Address						Site.com Contributor User <input type="checkbox"/>
Time Zone	(GMT-07:00) Pacific Daylight Time (America/Los_Angeles)					Site.com Publisher User <input type="checkbox"/>
Locale	English (United States)					WDC User <input type="checkbox"/>
Language	English					Mobile Push Registrations View
Delegated Approver						Data.com User Type View
Manager						Accessibility Mode (Classic Only) <input type="checkbox"/> View
Receive Approval Request Emails	Only if I am an approver					Debug Mode <input type="checkbox"/> View
Federation ID						High-Contrast Palette on Charts <input type="checkbox"/> View

3. Patient User

- Profile: Customer Community User
- License: Customer Community
- Role: Patient
- Access: Limited portal access → view prescriptions, book consultations.

- **Name:** Rajesh Kumar
- **Email:** rajesh.kumar@patient.com
- **Username:** patient1@telemedicine.com
- **Role:** Patient
- **User License:** Customer Community
- **Profile:** Patient Profile (restricted, only self-records visible)

4. Admin User

- Profile: System Administrator
 - License: Salesforce
 - Role: Admin
 - Access: Full control of the Org, configurations, and monitoring.
- **Name:** Priya Singh
 - **Email:** admin.priya@telemedicine.com
 - **Username:** admin1@telemedicine.com
 - **Role:** Hospital Admin
 - **User License:** Salesforce
 - **Profile:** System Administrator (full access)

User Priya Singh Singh

User Profile Help for this Page

Permission Set Assignments (0) | Permission Set Assignments Activation Required (0) | Permission Set Group Assignments (0) | Permission Set License Assignments (0) | Personal Groups (0) | Public Group Membership (0) | Queue Membership (0) | Team (0) | Managers in the Role Hierarchy (0) | OAuth Apps (0) | Third-Party Account Links (0) | Built-in Authenticators (0) | Installed Mobile Apps (0) | Authentication Settings for External Systems (0) | Login History (0) | User Provisioning Accounts (0)

User Detail		Edit	Sharing	Reset Password	Freeze	View Summary
Name	Priya Singh Singh	Role	Hospital Admin			
Alias	psingh	User License	Salesforce			
Email	admin.priya@telemedicine.com [Verify]	Profile	System Administrator			
Username	admin1@telemedicine.com	Active	<input checked="" type="checkbox"/>			
Nickname	User17576780822576745734	Marketing User	<input type="checkbox"/>			
Title		Offline User	<input type="checkbox"/>			
Company		Knowledge User	<input type="checkbox"/>			
Department		Flow User	<input type="checkbox"/>			
Division		Service Cloud User	<input type="checkbox"/>			
Address		Site.com Contributor User	<input type="checkbox"/>			
Time Zone	(GMT-07:00) Pacific Daylight Time (America/Los_Angeles)	Site.com Published User	<input type="checkbox"/>			
Locale	English (United States)	WDC User	<input type="checkbox"/>			
Language	English	Mobile Push Registrations	View			
Delegated Approver		Data.com User Type	View			
Manager		Accessibility Mode (Classic Only)	<input type="checkbox"/>			
Receive Approval Request Emails	Only if I am an approver	Debug Mode	<input type="checkbox"/>			
Federation ID		High-Contrast Palette on Charts	<input type="checkbox"/>			

7. Profiles

Profiles define baseline permissions for each type of user.

- Doctor Profile → Access to Patient, Appointment, Teleconsultation objects.
- Pharmacy Staff Profile → Access to Medicine Inventory, Order objects.
- Patient Profile → Limited access via community, can view only self-records.
- System Admin Profile → Full access.

8. Roles

Roles help in data visibility (hierarchy-based).

- Admin / Hospital Admin (Top of hierarchy)
- Full org visibility.
 - Doctor
- Can access assigned patient records.
 - Pharmacy Staff
 - o Can access medicine inventory and assigned orders.
 - Patient
 - o Only their own data (no hierarchy beyond).



9. Permission Sets

Permission sets give additional access without modifying profiles.

- Teleconsultation Access → Grants doctors permission to conduct video consultations.
- Analytics Access → Allows Health Dept Officers to view dashboards.
- Pharmacy Inventory Manager → Special permission to update stock levels.

The screenshot shows the 'Permission Set Overview' page for a permission set named 'Teleconsultation Access'. The page includes a toolbar with 'Find Settings...', 'Clone', 'Delete', 'Edit Properties', 'Manage Assignments', and 'View Summary'. The main table displays the following details:

Description	API Name	Namespace Prefix
License	Teleconsultation_Access	
Session Activation Required	Anjali.Raiwar	9/12/2025, 5:58 AM
Permission Set Groups Added To	0	Anjali.Raiwar 9/12/2025, 5:58 AM

10. OWD

Path: Setup → Sharing Settings

- Patient Records: Private (only Doctor + Patient can access).
- Teleconsultation Records: Private (only involved Doctor & Patient).
- Medicine Inventory: Public Read-Only (so staff can view, but only Pharmacy can edit).
- Hospital Announcements: Public Read/Write (all users can view/edit as needed).

11. Sharing Rules

Created to extend access beyond OWD.

- Rule 1: Share all Patient Records with Doctor Role.
- Rule 2: Share Pharmacy Inventory (read-only) with Doctor Role so doctors can check stock before prescribing.
- Rule 3: Share Health Reports with Health Dept Officers for analysis.

12. Login Access Policies

Admins can log in as any user for troubleshooting.

- Session Timeout: 2 hours.
- Trusted IP Ranges: Limited to India-based ranges to enhance security.

13. Dev Org Setup

- A Developer Org is used for testing, LWC development, and configuration.
- Enabled Dev Hub for Salesforce DX (SFDX).
- Source-tracked development with VS Code + GitHub integration.

14. Sandbox Usage

- Developer Sandbox: For coding and testing LWC components.
- Partial Copy Sandbox: For testing with sample patient & medicine records.
- Deployment between Sandbox and Production done using Change Sets.

15. Deployment Basics

- Code & metadata stored in GitHub Repository.
- Deployment via:
 - o Change Sets (Admin-friendly way).
 - o SFDX CLI (Salesforce DX) for CI/CD pipeline simulation.
- Pre-deployment validations: Run unit tests, check profiles/permissions.

Outcome of Phase 2

With this setup:

- Users, roles, and permissions are clearly defined.
- Data access is secured via OWD & Sharing Rules.
- Business hours, holidays, and fiscal year ensure correct SLAs.
- Developer workflow is ready with Sandboxes, GitHub, and SFDX.

SALESFORCE PROJECT IMPLEMENTATION PHASES

Phase 3: Data Modeling & Relationships

In this phase, I design the Salesforce data model for the Telemedicine Access for Rural Healthcare project. Proper data modeling ensures that all stakeholders (patients, doctors, pharmacy staff, and health officers) can interact with the system smoothly while maintaining security and scalability.

16. Standard & Custom Objects

I used both standard Salesforce objects and custom objects.

Standard Objects Used:

- User → Doctors, Admins, Pharmacy Staff.
- Contact → Patients (extended with custom fields).
- Account → Civil Hospital / Clinics.

Custom Objects Created:

- Appointment__c – Stores teleconsultation appointment details.
- Prescription__c – Holds doctor-issued prescriptions.
- Medicine_Inventory__c – Tracks stock availability at local pharmacies.
- Symptom_Checker__c – Logs AI-based symptom assessments.
- Notification_Log__c – Tracks reminders, SMS, and alerts.

17. Fields

Each custom object includes key fields to capture project requirements.

Appointment__c

- Patient (Lookup → Contact)
- Doctor (Lookup → User)
- Date/Time (DateTime)
- Status (Picklist: Scheduled, Completed, Cancelled, No-Show)
- Mode (Picklist: Video, In-person)

Details

Description	
API Name	Enable Reports
Appointment_c	✓
Custom	Track Activities
✓	✓
Singular Label	Track Field History
Appointment	✓
Plural Label	Deployment Status
Appointment	Deployed
	Help Settings
	Standard salesforce.com Help Window

Prescription__c

- Appointment (Lookup → Appointment__c)
- Doctor Notes (Long Text)
- Medicine List (Text Area)
- Next Review Date (Date)

Details

Description	
API Name	Enable Reports
Prescription__c	✓
Custom	Track Activities
✓	✓
Singular Label	Track Field History
Prescription	✓
Plural Label	Deployment Status
Prescriptions	Deployed
	Help Settings
	Standard salesforce.com Help Window

Medicine_Inventory__c

- Medicine Name (Text)
- Stock Quantity (Number)
- Expiry Date (Date)
- Pharmacy Location (Text)

Details

Description

API Name

Medicine_Inventory__c

Custom

✓

Singular Label

Medicine Inventory

Plural Label

Medicine Inventories

Enable Reports

✓

Track Activities

✓

Track Field History

✓

Deployment Status

Deployed

Help Settings

Standard salesforce.com Help Window

Symptom_Checker__c

- Patient (Lookup → Contact)
- Entered Symptoms (Text Area)
- Suggested Action (Picklist: Home Care, Doctor Visit, Emergency)

Details

Description

API Name

Symptom_Checker__c

Custom

✓

Singular Label

Symptom Checker

Plural Label

Symptom Checkers

Enable Reports

✓

Track Activities

✓

Track Field History

✓

Deployment Status

Deployed

Help Settings

Standard salesforce.com Help Window

Notification_Log__c

- Patient (Lookup → Contact)
- Type (Picklist: SMS, WhatsApp, Email)
- Message Content (Long Text)
- Status (Picklist: Sent, Failed, Pending)

Details

Description

API Name

Notification_Log__c

Custom



Singular Label

Notification Log

Plural Label

Notification Logs

Enable Reports



Track Activities



Track Field History



Deployment Status

Deployed

Help Settings

Standard salesforce.com Help Window

Prescription_Medicine__c

Details

Description

API Name

Prescription_Medicine__c

Custom



Singular Label

Prescription Medicine

Plural Label

Prescription Medicines

Enable Reports



Track Activities



Track Field History



Deployment Status

Deployed

Help Settings

Standard salesforce.com Help Window

18. Record Types

I used record types to manage different processes within the same object.

Appointment__c Record Types:

- General Consultation
- Specialist Consultation
- Follow-Up

Record Types		Quick Find	New	Page Layout Assignment
Record Type Label	Description	Active	Modified By	
General Consultation		✓	Anjali Raikwar, 9/16/2025, 6:37 AM	▼
Specialist Consultation		✓	Anjali Raikwar, 9/16/2025, 6:38 AM	▼

Prescription__c Record Types:

- Regular Prescription
- Emergency Prescription

Record Types			
2 Items, Sorted by Record Type Label			
RECORD TYPE LABEL	DESCRIPTION	ACTIVE	MODIFIED BY
Emergency Prescription		✓	Anjali Raikwar, 9/16/2025, 9:01 AM
Regular Prescription		✓	Anjali Raikwar, 9/16/2025, 9:00 AM

This allows customized page layouts and business processes.

19. Page Layouts

I created different layouts for different users:

- Doctor Layout (Appointment): Shows patient details, symptoms, prescription fields.
- Pharmacy Layout (Medicine): Shows medicine name, stock, expiry.
- Patient Layout (Community): Only summary fields visible (doctor, date, notes).

Page Layouts			
4 Items, Sorted by Page Layout Name			
PAGE LAYOUT NAME	CREATED BY	MODIFIED BY	
Admin Appointment Layout	Anjali Raikwar, 9/16/2025, 6:35 AM	Anjali Raikwar, 9/16/2025, 6:35 AM	▼
Appointment Layout	Anjali Raikwar, 9/15/2025, 7:24 AM	Anjali Raikwar, 9/16/2025, 6:19 AM	▼
Doctor Appointment Layout	Anjali Raikwar, 9/16/2025, 6:35 AM	Anjali Raikwar, 9/16/2025, 6:35 AM	▼
Patient Booking Layout	Anjali Raikwar, 9/16/2025, 6:34 AM	Anjali Raikwar, 9/16/2025, 6:34 AM	▼

20. Compact Layouts

Compact layouts improve mobile/Lightning view.

1. Appointment Compact Layout:

- Patient
- Doctor
- Date/Time
- Status

Compact Layouts				
2 Items, Sorted by Label				
LABEL	API NAME	PRIMARY	MODIFIED BY	LAST MODIFIED
Appointment Highlights	Appointment_Highlights	✓	Anjali Raikwar	9/16/2025, 6:59 AM

2. Medicine Inventory Compact Layout:

- Medicine Name
- Stock Quantity
- Expiry Date

Compact Layouts		Quick Find	New	Compact Layout Assignment
LABEL	API NAME	PRIMARY	MODIFIED BY	LAST MODIFIED
Medicine Inventory	Medicine_Inventory	✓	Anjali Raikwar	9/16/2025, 7:06 AM

21. Schema Builder

I used “Schema Builder” in Salesforce to:

- Visualize relationships between objects.
- Drag and drop new fields.
- Ensure field-level security for Patients vs Doctors.

Schema Snapshot:

- Contact → Appointment__c (Lookup)
- User → Appointment__c (Lookup)
- Appointment__c → Prescription__c (Master-Detail)
- Appointment__c → Notification_Log__c (Lookup)
- Medicine_Inventory__c → Prescription__c (Junction for many-to-many)

22. Lookup vs Master-Detail vs Hierarchical Relationships

Lookup:

- Appointment__c → Patient (Contact)
- Appointment__c → Doctor (User)

Master-Detail:

- Prescription__c → Appointment__c (if Appointment is deleted, prescription also deleted).

Hierarchical:

- Only for Users (not used in our model, but Admin can be superior to Staff).

23. Junction Objects

I created a “ Prescription_Medicine__c (Junction Object) ” to handle many-to-many between Prescription and Medicine_Inventory.

Fields:

- Prescription (Master-Detail → Prescription_c)
- Medicine (Master-Detail → Medicine_Inventory_c)
- Dosage (Text)
- Duration (Number of Days)

This ensures a single prescription can include multiple medicines, and one medicine can belong to multiple prescriptions.

24. External Objects

To integrate with “ Government Health Databases ” or “ Pharmacy ERP systems ”, we define “ External Objects.”

Examples:

- Govt_Health_Stats_x → External object for village-level healthcare reports.
- Pharma_Supplier_Inventory_x → External object to fetch stock availability from pharmacy ERP.

These external objects use Salesforce Connect (OData) to avoid data duplication).

External Data Sources

Help for this Page 

Access data in other Salesforce orgs as well as third-party databases and content systems.

View:	All	Create New View	New External Data Source	Connect to Amazon DynamoDB	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z Other All
Action	Name 	External Data Source	Type	URL	
Edit Del	Pharma_Supplier	Pharma_Supplier	Salesforce Connect: OData 4.0	https://services.odata.org/V4/Northwind/Northwind.svc/	

External Objects					
Action	Label	Namespace Prefix	Description	Table Name	
Edit Erase Validate	Alphabetical_list_of_product		Alphabetical_list_of_products	Alphabetical_list_of_products	
Edit Erase Validate	Category		Categories	Categories	
Edit Erase Validate	Customer		Customers	Customers	
Edit Erase Validate	Employee		Employees	Employees	
Edit Erase Validate	Order_Detail		Order_Details	Order_Details	
Edit Erase Validate	Order		Orders	Orders	
Edit Erase Validate	Product		Products	Products	
Edit Erase Validate	Pharma_Category_x		Categories	Categories	
Edit Erase Validate	Pharma_Customer_x		Customers	Customers	
Edit Erase Validate	Pharma_Employee_x		Employees	Employees	
Edit Erase Validate	Pharma_OrderDetail_x		Order_Details	Order_Details	
Edit Erase Validate	Pharma_Order_x		Orders	Orders	
Edit Erase Validate	Pharma_Product_x		Products	Products	

Field-Level Security (FLS)

Even if a field exists, not everyone should see it (privacy).

Example: Aadhaar Number should only be visible to Doctors and Admin, not to Pharmacy Staff.

Contact Custom Field
Aadhaar Number
[Back to Contact Fields](#)

[Validation Rules \[0\]](#)

Custom Field Definition Detail		Edit	Set Field-Level Security	View Field Accessibility	Where is this used?
Field Information					
Field Label	Aadhaar Number	Object Name	Contact		
Field Name	Aadhaar_Number	Data Type	Text		
API Name	Aadhaar_Number__c				
Description					
Help Text	Enter 12-digit Aadhaar number				
Data Owner					
Field Usage					
Data Sensitivity Level					
Compliance Categorization					
Created By	Anjali Raikwar, 9/16/2025, 8:50 AM	Modified By	Anjali Raikwar, 9/16/2025, 8:50 AM		
General Options					
Required	<input type="checkbox"/>				
Unique	<input checked="" type="checkbox"/>				
Case Sensitive	<input type="checkbox"/>				
External ID	<input type="checkbox"/>				
Default Value					

Visible for only:

- Doctor Profile
 - System Administrator Profile
- Now, only doctors and admins can see Aadhaar, others can't.

Outcome of Phase 3

- Standard & Custom objects defined for Telemedicine use case.
- Record types, page layouts, and compact layouts implemented for role-specific UI.
- Relationships established (Lookup, Master-Detail, Junction Objects).
- Schema visualized using Schema Builder.
- External objects planned for ERP/Govt integrations.

With Phase 3 complete, our Salesforce org now has a Solid Data Foundation for building LWC components, automation, and analytics.

SALESFORCE PROJECT IMPLEMENTATION PHASES

Phase 4: Process Automation (Admin)

In this phase, I automate business processes in Salesforce to reduce manual work, ensure data quality, and improve patient care efficiency. We use **Validation Rules, Workflow Rules, Process Builder, Approval Processes, Flow Builder, and Notifications** tailored for Telemedicine CRM.

➤ Validation Rules

Validation Rules ensure data accuracy before records are saved.

➤ Patient Aadhaar Validation

- Object: Patient__c
- Rule: Aadhaar_Number__c must be 12 digits.
- Formula: NOT(REGEX(Aadhaar_Number__c, "[0-9]{12}"))
- Error: "Please enter a valid 12-digit Aadhaar number."

➤ Appointment Date Check

- Object: Appointment__c
- Rule: Appointment_Date__c must be greater than TODAY().
- Error: "Appointment date cannot be in the past."

Patient Validation Rule

[Back to Patient](#)

Validation Rule Detail		Edit	Clone	
Rule Name	Aadhaar_Validation	Active	<input checked="" type="checkbox"/>	
Error Condition Formula	NOT(REGEX(Aadhaar_Number__c, "[0-9]{12}"))			
Error Message	Please enter a valid 12-digit Aadhaar number.	Error Location	Aadhaar Number	
Description		Created By	Anjali Raikwar, 9/17/2025, 6:24 AM	Modified By
				Anjali Raikwar, 9/17/2025, 6:24 AM
		Edit	Clone	

Appointment Validation Rule

[Back to Appointment](#)

Validation Rule Detail		Edit	Clone	
Rule Name	Appointment_Date_Check	Active	<input checked="" type="checkbox"/>	
Error Condition Formula	Appointment_Date__c < TODAY()			
Error Message	Appointment date cannot be in the past.	Error Location	Top of Page	
Description		Created By	Anjali Raikwar, 9/17/2025, 6:31 AM	Modified By
				Anjali Raikwar, 9/17/2025, 6:31 AM
		Edit	Clone	

➤ Workflow Rules

Workflow Rules help automate simple actions.

Send Email Reminder

- Object: Appointment__c
- Condition: Appointment__Date__c = Tomorrow.
- Action: Send email alert to Patient__c.Email and Doctor__c.Email.

Workflow Rule Help for this Page

Appointment Reminder

Go with the flow! With Flow Builder, the future of low-code automation, you can do everything you do with workflow rules — and more! Salesforce plans to retire workflow rules and recommends building automation in Flow Builder. [Tell Me More](#) | [Migrate your workflow rules to flows](#)

Workflow Rule Detail

Rule Name	Appointment Reminder	Object	Appointment
Active	<input type="checkbox"/>	Evaluation Criteria	Evaluate the rule when a record is created, and any time it's edited to subsequently meet criteria
Description			
Rule Criteria	Appointment: Appointment Date EQUALS TOMORROW		
Created By	Anjali Raikwar 9/17/2025, 6:40 AM	Modified By	Anjali Raikwar 9/19/2025, 6:07 AM

Medicine Stock Low

- Object: Medicine__c
- Condition: Quantity__c < 10.
- Action: Send notification to Pharmacy Staff.

Workflow Rule Help for this Page

Low Medicine Stock Alert

Go with the flow! With Flow Builder, the future of low-code automation, you can do everything you do with workflow rules — and more! Salesforce plans to retire workflow rules and recommends building automation in Flow Builder. [Tell Me More](#) | [Migrate your workflow rules to flows](#)

Workflow Rule Detail

Rule Name	Low Medicine Stock Alert	Object	Medicine
Active	<input type="checkbox"/>	Evaluation Criteria	Evaluate the rule when a record is created, and every time it's edited
Description			
Rule Criteria	Medicine: Quantity LESS THAN 10		
Created By	Anjali Raikwar 9/19/2025, 6:30 AM	Modified By	Anjali Raikwar 9/19/2025, 6:30 AM

SETUP Classic Email Templates

Custom Email Template Help for this Page

Medicine_Running_Low

Preview your email template below.

Email Template Detail

Email Templates from Salesforce	Unfiled Public Classic Email Templates	Available For Use	<input checked="" type="checkbox"/>
Email Template Name	Medicine_Running_Low	Last Used Date	
Template Unique Name	Medicine_Running_Low	Times Used	
Encoding	Unicode (UTF-8)		
Author	Anjali Raikwar [Change]		
Description			
Created By	Anjali Raikwar 9/19/2025, 6:22 AM	Modified By	Anjali Raikwar 9/19/2025, 6:23 AM

Email Template Send Test and Verify Merge Fields

Subject | Low Stock Alert: Medicine Inventory

HTML Preview

Low Stock Alert: Medicine Inventory

Dear Pharmacy Team,

This is an automated notification regarding low stock for the following medicine:

Medicine Name:	{!Medicine__c.Name}
Quantity Available:	{!Medicine__c.Quantity__c}
Created On:	{!Medicine__c.CreatedDate}
Last Updated:	{!Medicine__c.LastModifiedDate}

Please review the inventory and restock as needed to avoid disruption in supply.

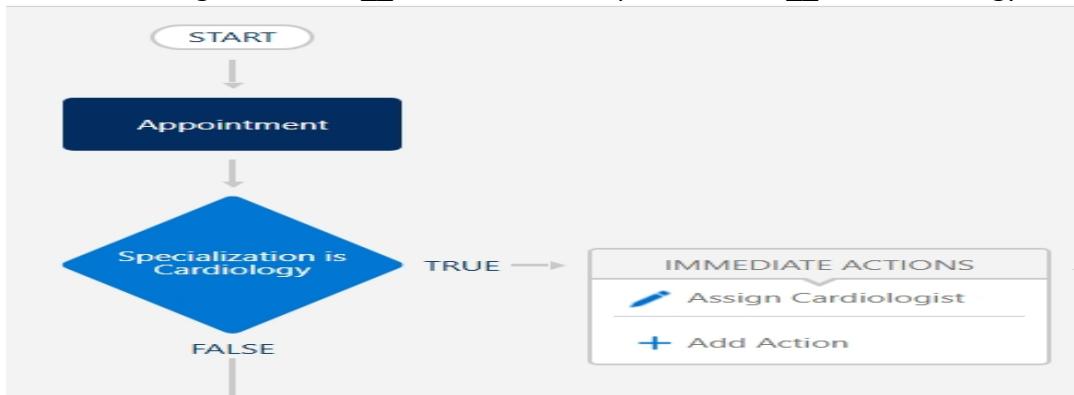
Thank you,
Inventory Management System

➤ Process Builder

Process Builder is used for slightly complex automation with if/else logic.

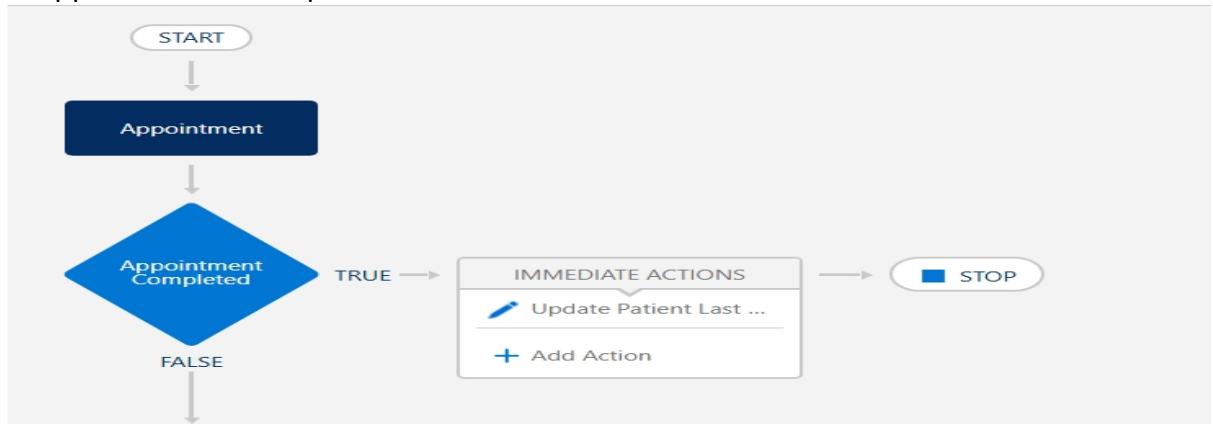
Auto-Assign Doctor

- Object: Appointment__c
- Criteria: Specialization__c = "Cardiology"
- Action: Assign to Doctor__c where Doctor.Specialization__c = "Cardiology".



Update Patient Last Visit

- Object: Appointment__c
- Action: Update Patient__c.Last_Visit_Date__c with Appointment__c.Appointment_Date__c when appointment is completed.



➤ Approval Process

Approval processes help manage requests that need authorization.

Medicine Restock Request

- Object: Medicine__c
- Criteria: Quantity__c < 5
- Step 1: Pharmacy Staff submits a restock request.
- Step 2: Approved by Hospital Admin.
- Step 3: Once approved → Purchase order is created (custom object: Purchase_Order__c).

Approval Processes
Medicine: Medicine Restock Approval
[Help for this Page](#)

[Back to Approval Process List](#)

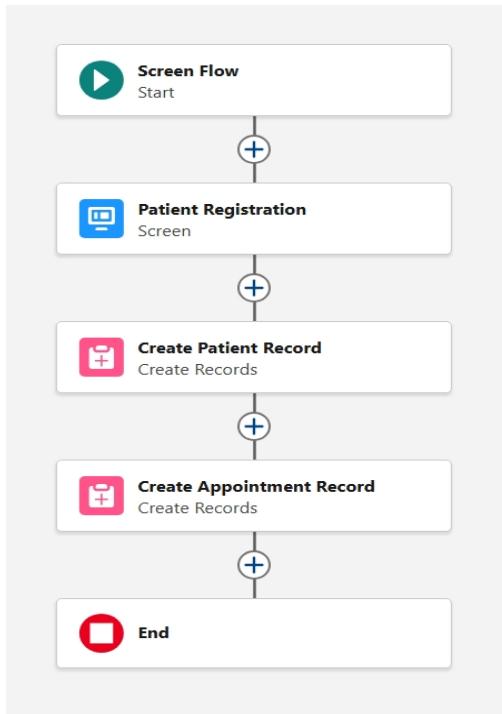
Process Definition Detail		Action
Process Name	Medicine Restock Approval	Active <input checked="" type="checkbox"/>
Unique Name	Medicine_Restock_Approval	Next Automated Approver Determined By
Description		
Entry Criteria	Medicine: Quantity LESS THAN 5	
Record Editability	Administrator ONLY	Allow Submitters to Recall Approval Requests <input type="checkbox"/>
Approval Assignment Email Template		
Initial Submitters	Medicine Owner	
Created By	Anjali Raikwar, 9/19/2025, 8:06 AM	Modified By Anjali Raikwar, 9/19/2025, 8:07 AM

➤ Flow Builder

Flows provide advanced automation (Screen Flows, Record-Triggered, Auto-launched).

Screen Flow: New Patient Registration

- Collect Aadhaar, Name, Age, Symptoms.
- Auto-create Patient__c record + Initial Appointment__c.



Record-Triggered Flow: Appointment Reminder

- Trigger: 1 day before Appointment__c.Appointment_Date__c
- Action: Send WhatsApp/SMS/Email to patient.

Scheduled Flow: Expiring Medicine Notification

- Run daily at 10 AM.
- Check Medicine__c.Expiry_Date__c = TODAY() + 2.
- Send alert to Pharmacy Staff.

➤ Email Alerts

Email templates + alerts linked with rules and flows.

- Appointment Confirmation Email → Patient & Doctor.
- Low Inventory Email → Hospital Admin.

Text Email Template
Appointment Confirmation

Preview your email template below.

Email Template Detail		Edit	Delete	Clone
Email Templates from Salesforce	Unfiled Public Classic Email Templates			
Email Template Name	Appointment Confirmation			
Template Unique Name	Appointment_Confirmation			
Encoding	Unicode (UTF-8)			
Author	Anjali Raikwar [Change]			
Description				
Created By	Anjali Raikwar, 9/19/2025, 10:47 AM			
Modified By	Anjali Raikwar, 9/19/2025, 10:47 AM			

Help for this Page

Text Email Template
Medicine Expiry Alert

Preview your email template below.

Email Template Detail		Edit	Delete	Clone
Email Templates from Salesforce	Unfiled Public Classic Email Templates			
Email Template Name	Medicine_Expiry_Alert			
Template Unique Name	Medicine_Expiry_Alert			
Encoding	Unicode (UTF-8)			
Author	Anjali Raikwar [Change]			
Description				
Created By	Anjali Raikwar, 9/19/2025, 10:48 AM			
Modified By	Anjali Raikwar, 9/19/2025, 10:48 AM			

Help for this Page

➤ Field Updates

Automatically update fields.

- Appointment__c.Status__c → "Completed" when Doctor marks Consultation Complete.
- Patient__c.Last_Consulted_Doctor__c → Auto-updated on new appointment.

Mark Appointment Completed



Go with the flow! With Flow Builder, the future of low-code automation, you can do everything you do with workflow rules — and more! Salesforce plans to retire workflow rules and recommends building automation in Flow Builder. [Tell Me More](#) | [Migrate your workflow rules to flows](#)

Workflow Rule Detail

[Edit](#) [Clone](#) [Deactivate](#)

Rule Name	Mark Appointment Completed	Object	Appointment
Active	<input checked="" type="checkbox"/>	Evaluation Criteria	Evaluate the rule when a record is created, and every time it's edited
Description	Your Appointment is marked as Completed.		
Rule Criteria	Appointment: Doctor EQUALS Done		
Created By	Anjali Raikwar, 9/19/2025, 10:54 AM	Modified By	Anjali Raikwar, 9/19/2025, 10:55 AM

➤ Tasks

Assign tasks automatically.

- If Appointment is missed → Auto-create Task for Doctor to follow up with patient.

➤ Custom Notifications

Use in-app and mobile notifications.

- Notify Doctor: "New appointment assigned."
- Notify Pharmacy: "Paracetamol stock below 10 units."
- Notify Patient: "Your appointment with Dr. Sharma is tomorrow at 10:30 AM."

New Appointment Assigned	New_Appointment_Assigned	✓	✓	▼
Stock Running Low	Stock_Running_Low	✓	✓	▼

SALESFORCE PROJECT IMPLEMENTATION PHASES

Phase 5: Apex Programming (Developer)

In this phase, I have extended Salesforce point-and-click features with **Apex programming** to implement advanced business logic for our Telemedicine CRM. Since healthcare use-cases often require automation, bulk processing, and integrations, Apex gives us flexibility and scalability.

➤ Classes & Objects

I have created “Apex classes” to encapsulate business logic.

- TelemedicineService

```
► TelemedicineUtils.cls U X
nabha > force-app > main > default > classes > ► TelemedicineUtils.cls > 📁 TelemedicineUtils > ⚡ sendSMS(List<String>, String) : void
1  public with sharing class TelemedicineUtils {
2      // Validate Aadhaar (12 digits)
3      public static Boolean isAadhaarValid(String aadhaar){
4          if(String.isBlank(aadhaar)) return false;
5          return Pattern.matches('^\d{12}', aadhaar);
6      }
7
8      // Future method to simulate SMS sending
9      @future
10     public static void sendSMS(List<String> phoneNumbers, String message) {
11         if(phoneNumbers == null || phoneNumbers.isEmpty()) return;
12         for(String ph : phoneNumbers){
13             // For demo: only log. Replace with HTTP callout to SMS provider when needed.
14             System.debug('[Telemedicine SMS] to: ' + ph + ' message: ' + message);
15         }
16     }
17
18     // Enqueue a Queueable job (wrapper)
19     public static Id enqueue(Queueable job) {
20         return System.enqueueJob(job);
21     }
22 }
```

- PurchaseOrderService

```
► PurchaseOrderService.cls U X
nabha > force-app > main > default > classes > ► PurchaseOrderService.cls > ...
1  public with sharing class PurchaseOrderService {
2      @InvocableMethod(label='Create Purchase Order')
3      public static List<Id> createPurchaseOrders(List<PORequest> requests){
4          List<Purchase_Order__c> pos = new List<Purchase_Order__c>();
5          for(PORequest r : requests){
6              Purchase_Order__c p = new Purchase_Order__c();
7              p.Supplier__c = r.supplierId;
8              p.Medicine__c = r.medicineId;
9              p.Quantity__c = r.quantity;
10             p.Status__c = 'Pending';
11             p.Order_Date__c = Date.today();
12             pos.add(p);
13         }
14         if(!pos.isEmpty()) insert pos;
15         List<Id> ids = new List<Id>();
16         for(Purchase_Order__c pRec : pos) ids.add(pRec.Id);
17         return ids;
18     }
19
20     public class PORequest {
21         @InvocableVariable(required=true) public Id supplierId;
22         @InvocableVariable(required=true) public Id medicineId;
23         @InvocableVariable(required=true) public Integer quantity;
24     }
25 }
```

- AppointmentNotificationQueueable

```
AppointmentNotificationQueueable.cls X
nabha > force-app > main > default > classes > AppointmentNotificationQueueable.cls > ...
1 public with sharing class AppointmentNotificationQueueable implements Queueable {
2     private List<Id> appointmentIds;
3
4     // Constructor takes list of Appointment IDs
5     public AppointmentNotificationQueueable(List<Id> appointmentIds){
6         this.appointmentIds = appointmentIds;
7     }
8
9     public void execute(QueueableContext qc){
10        try {
11            if(appointmentIds == null || appointmentIds.isEmpty()) return;
12
13            // Fetch appointments with patient details
14            List<Appointment__c> appts = [
15                SELECT Id, Appointment_DateTime__c, Patient__c,
16                | Patient__r.Name, Patient__r.Phone__c, Patient__r.Email__c
17                | FROM Appointment__c
18                | WHERE Id IN :appointmentIds
19            ];
20
21            List<String> phones = new List<String>();
22
23            for(Appointment__c a : appts){
24                String msg = 'Dear ' + (a.Patient__r != null ? a.Patient__r.Name : 'Patient') +
25                | ', your appointment is scheduled on ' +
26                | String.valueOf(a.Appointment_DateTime__c) + '.';
27
28                // Collect phones for SMS
29                if(a.Patient__r != null && a.Patient__r.Phone__c != null) {
30                    phones.add(a.Patient__r.Phone__c);
31                }
32
33                // Send Email
34                if(a.Patient__r != null && a.Patient__r.Email__c != null){
35                    Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
36                    mail.setToAddresses(new String[]{a.Patient__r.Email__c});
37                    mail.setSubject('Appointment Scheduled');
```

- AppointmentArchiveBatch

```
AppointmentArchiveBatch.cls X
nabha > force-app > main > default > classes > AppointmentArchiveBatch.cls > ...
1 global class AppointmentArchiveBatch implements Database.Batchable<sObject>, Database.Stateful {
2     global Database.QueryLocator start(Database.BatchableContext bc) {
3         Datetime cutoff = System.now().addYears(-1);
4         return Database.getQueryLocator([SELECT Id FROM Appointment__c WHERE Appointment_DateTime__c < :cutoff AND (Archived__c = false OR Archived__c = null)]);
5     }
6     global void execute(Database.BatchableContext bc, List<sObject> scope){
7         List<Appointment__c> toUpdate = new List<Appointment__c>();
8         for(sObject s : scope){
9             Appointment__c a = (Appointment__c)s;
10            a.Archived__c = true;
11            toUpdate.add(a);
12        }
13        if(!toUpdate.isEmpty()) update toUpdate;
14    }
15    global void finish(Database.BatchableContext bc){
16        System.debug('AppointmentArchiveBatch finished.');
17    }
18 }
```

- SendPrescriptionQueueable

```
SendPrescriptionQueueable.cls U X
nabha > force-app > main > default > classes > SendPrescriptionQueueable.cls > ...
1 public with sharing class SendPrescriptionQueueable implements Queueable {
2     private Id prescId;
3     public SendPrescriptionQueueable(Id prescId){
4         this.prescId = prescId;
5     }
6     public void execute(QueueableContext qc){
7         try {
8             Prescription__c p = [SELECT Id, Notes__c, Appointment__c, Appointment__r.Patient__r.Email__c FROM Prescription__c WHERE Id = :prescId LIMIT 1];
9             if(p != null && p.Appointment__r != null && p.Appointment__r.Patient__r.Email__c != null){
10                 Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
11                 mail.setToAddresses(new String[]{p.Appointment__r.Patient__r.Email__c});
12                 mail.setSubject('Your Prescription from Telemedicine Portal');
13                 mail.setPlainTextBody('Dear Patient,\nPlease find your prescription notes:\n' + p.Notes__c + '\n\nRegards');
14                 Messaging.sendEmail(new Messaging.SingleEmailMessage[]{mail});
15             }
16         } catch(Exception e){
17             ErrorHandler.logException(e, 'SendPrescriptionQueueable.execute');
18         }
19     }
20 }
```

- ErrorHandler

```
ErrorHandler.cls U X
nabha > force-app > main > default > classes > ErrorHandler.cls > ...
1 public with sharing class ErrorHandler {
2     public static void logException(Exception e, String contextMessage) {
3         try {
4             Error_Log__c logRec = new Error_Log__c();
5             logRec.Message__c = e.getMessage();
6             logRec.Stack_Trace__c = e.getStackTraceString();
7             logRec.Context__c = contextMessage;
8             insert logRec;
9         } catch (Exception ex) {
10             System.debug('ErrorHandler failed to log: ' + ex.getMessage());
11         }
12     }
13 }
```

- CustomExceptions

```
CustomExceptions.cls U X
nabha > force-app > main > default > classes > CustomExceptions.cls > ...
1 public class CustomExceptions {
2     public class AppointmentConflictException extends Exception {}
3     public class MedicineOutOfStockException extends Exception {}
4 }
```

➤ Apex Triggers

Triggers automate logic that cannot be handled via declarative tools.

We created:

- AppointmentTrigger (delegates to AppointmentTriggerHandler)

```
≡ AppointmentTrigger.trigger U X
nabha > force-app > main > default > triggers > ≡ AppointmentTrigger.trigger > ⚒ AppointmentTrigger
1 trigger AppointmentTrigger on Appointment__c (before insert, before update, after insert, after update) {
2     if(Trigger.isBefore){
3         if(Trigger.isInsert) AppointmentTriggerHandler.beforeInsert(Trigger.new);
4         if(Trigger.isUpdate) AppointmentTriggerHandler.beforeUpdate(Trigger.new, Trigger.oldMap);
5     }
6     if(Trigger.isAfter){
7         if(Trigger.isInsert) AppointmentTriggerHandler.afterInsert(Trigger.new);
8         if(Trigger.isUpdate) AppointmentTriggerHandler.afterUpdate(Trigger.new, Trigger.oldMap);
9     }
10 }
```

```
≡ AppointmentTriggerHandler.cls U X
nabha > force-app > main > default > classes > ≡ AppointmentTriggerHandler.cls > ⚒ AppointmentTriggerHandler > ⚒ preventDoubleBooking
1 public with sharing class AppointmentTriggerHandler {
2     // BEFORE INSERT
3     public static void beforeInsert(List<Appointment__c> newList) {
4         for(Appointment__c a : newList){
5             if(String.isBlank(a.Status__c)) a.Status__c = 'Scheduled';
6             if(a.Appointment_DateTime__c == null) {
7                 a.addError('Appointment date/time is required.');
8             }
9         }
10     preventDoubleBooking(newList, null);
11 }
12
13 // BEFORE UPDATE
14 public static void beforeUpdate(List<Appointment__c> newList, Map<Id, Appointment__c> oldMap) {
15     for(Appointment__c a : newList){
16         if(a.Appointment_DateTime__c != null && a.Appointment_DateTime__c < System.now()){
17             a.addError('Appointment date cannot be in the past.');
18         }
19     }
20     preventDoubleBooking(newList, oldMap);
21 }
22
23 // AFTER INSERT
24 public static void afterInsert(List<Appointment__c> newList) {
25     try {
26         // collect ids and enqueue notifications
27         List<Id> ids = new List<Id>();
28         for(Appointment__c a : newList) ids.add(a.Id);
29         if(!ids.isEmpty()){
30             TelemedicineUtils.enqueue(new AppointmentNotificationQueueable(ids));
31         }
32     } catch(Exception e){
33         ErrorHandler.logException(e, 'AppointmentTriggerHandler.afterInsert');
34     }
35 }
36
37 // AFTER UPDATE (placeholder: can be expanded for status changes)
```

- PrescriptionTrigger (delegates to PrescriptionTriggerHandler)

```
≡ PrescriptionTrigger.trigger U X
nabha > force-app > main > default > triggers > ≡ PrescriptionTrigger.trigger > ⚒ PrescriptionTrigger
1 trigger PrescriptionTrigger on Prescription__c (before insert, before update, after insert, after update) {
2     PrescriptionTriggerHandler.handleTrigger(Trigger.isBefore, Trigger.isAfter, Trigger.isInsert, Trigger.isUpdate, Trigger.new, Trigger.oldMap);
3 }
```

```

● PrescriptionTriggerHandler.cls U
nabha > force-app > main > default > classes > ● PrescriptionTriggerHandler.cls > ...
1  public with sharing class PrescriptionTriggerHandler {
2    public static void handleTrigger(Boolean isBefore, Boolean isAfter, Boolean isInsert, Boolean isUpdate, List<Prescription__c> newList, Map<Id, ...
3      try {
4        if (isBefore && (isInsert || isUpdate)) {
5          for (Prescription__c pres : newList) {
6            if (pres.Medicine__c != null && pres.Quantity__c != null) {
7              Medicine_Inventory__c med = [
8                SELECT Id, Quantity__c
9                FROM Medicine_Inventory__c
10               WHERE Id = :pres.Medicine__c
11               LIMIT 1
12             ];
13
14             if (med.Quantity__c < pres.Quantity__c) {
15               throw new CustomExceptions.MedicineOutOfStockException();
16             }
17           }
18         }
19       } catch (CustomExceptions.MedicineOutOfStockException mex) {
20         ErrorHandler.logException(mex, 'Prescription insert/update error - Medicine Out of Stock');
21         throw mex; // re-throw so user sees the error
22     } catch (Exception ex) {
23       ErrorHandler.logException(ex, 'PrescriptionTriggerHandler unexpected error');
24       throw ex;
25     }
26   }
27 }
28

```

➤ Trigger Design Pattern

Implemented “Trigger Handler Pattern” to avoid hardcoding logic directly inside triggers.

Trigger: AppointmentTrigger

- Handler Class: AppointmentTriggerHandler
- Methods: beforeInsert, beforeUpdate, afterInsert, afterUpdate.

```

● AppointmentTriggerHandler.cls U
nabha > force-app > main > default > classes > ● AppointmentTriggerHandler.cls > ✎ AppointmentTriggerHandler > ⚡ preventDoubleBooking
1  public with sharing class AppointmentTriggerHandler {
2    // BEFORE INSERT
3    public static void beforeInsert(List<Appointment__c> newList) {
4      for(Appointment__c a : newList){
5        if(String.isBlank(a.Status__c)) a.Status__c = 'Scheduled';
6        if(a.Appointment_DateTime__c == null) {
7          aaddError('Appointment date/time is required.');
8        }
9      }
10     preventDoubleBooking(newList, null);
11   }
12
13   // BEFORE UPDATE
14   public static void beforeUpdate(List<Appointment__c> newList, Map<Id, Appointment__c> oldMap) {
15     for(Appointment__c a : newList){
16       if(a.Appointment_DateTime__c != null && a.Appointment_DateTime__c < system.now()){
17         aaddError('Appointment date cannot be in the past.');
18       }
19     }
20     preventDoubleBooking(newList, oldMap);
21   }
22
23   // AFTER INSERT
24   public static void afterInsert(List<Appointment__c> newList) {
25     try {
26       // collect ids and enqueue notifications
27       List<Id> ids = new List<Id>();
28       for(Appointment__c a : newList) ids.add(a.Id);
29       if(!ids.isEmpty()){
30         Telemedicineutils.enqueue(new AppointmentNotificationQueueable(ids));
31       }
32     } catch(Exception e){
33       ErrorHandler.logException(e, 'AppointmentTriggerHandler.afterInsert');
34     }
35   }
36
37   // AFTER UPDATE (placeholder: can be expanded for status changes)

```

➤ SOQL & SOSL

- “SOQL” used for retrieving patient and appointment details.
- “SOSL” implemented for global patient search by Aadhaar, phone, or email.

Already used in:

- AppointmentNotificationQueueable (queries Patient & Contact)

```
► AppointmentNotificationQueueable.cls U X
nabha > force-app > main > default > classes > ► AppointmentNotificationQueueable.cls > ...
1  public with sharing class AppointmentNotificationQueueable implements Queueable {
2      private List<Id> appointmentIds;
3
4      // Constructor takes list of Appointment IDs
5      public AppointmentNotificationQueueable(List<Id> appointmentIds){
6          this.appointmentIds = appointmentIds;
7      }
8
9      public void execute(QueueableContext qc){
10         try {
11             if(appointmentIds == null || appointmentIds.isEmpty()) return;
12
13             // Fetch appointments with patient details
14             List<Appointment__c> appts = [
15                 SELECT Id, Appointment_DateTime__c, Patient__c,
16                 | Patient__r.Name, Patient__r.Phone__c, Patient__r.Email__c
17                 FROM Appointment__c
18                 WHERE Id IN :appointmentIds
19             ];
20
21             List<String> phones = new List<String>();
22
23             for(Appointment__c a : appts){
24                 String msg = 'Dear ' + (a.Patient__r != null ? a.Patient__r.Name : 'Patient') +
25                 ', your appointment is scheduled on ' +
26                 String.valueOf(a.Appointment_DateTime__c) + '.';
27
28                 // Collect phones for SMS
29                 if(a.Patient__r != null && a.Patient__r.Phone__c != null) {
30                     phones.add(a.Patient__r.Phone__c);
31                 }
32
33                 // Send Email
34                 if(a.Patient__r != null && a.Patient__r.Email__c != null){
35                     Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
36                     mail.setToAddresses(new String[]{a.Patient__r.Email__c});
37                     mail.setSubject('Appointment Scheduled');
38
39             }
40
41         }
42     }
43
44     public void handleTrigger(Boolean isBefore, Boolean isAfter, Boolean isInsert, Boolean isUpdate, List<Prescription__c> newList, Map<Id, Prescription__c> oldList) {
45         try {
46             if (isBefore && (isInsert || isUpdate)) {
47                 for (Prescription__c pres : newList) {
48                     if (pres.Medicine__c != null && pres.Quantity__c != null) {
49                         Medicine_Inventory__c med = [
50                             SELECT Id, Quantity__c
51                             FROM Medicine_Inventory__c
52                             WHERE Id = :pres.Medicine__c
53                             LIMIT 1
54                         ];
55
56                         if (med.Quantity__c < pres.Quantity__c) {
57                             throw new CustomExceptions.MedicineOutOfStockException();
58                         }
59                     }
60                 }
61             }
62         }
63     }
64
65     public void handleDelete(List<Prescription__c> newList) {
66         try {
67             for (Prescription__c pres : newList) {
68                 if (pres.Medicine__c != null) {
69                     Medicine_Inventory__c med = [
70                         SELECT Id, Quantity__c
71                         FROM Medicine_Inventory__c
72                         WHERE Id = :pres.Medicine__c
73                         LIMIT 1
74                     ];
75
76                     if (med.Quantity__c < pres.Quantity__c) {
77                         throw new CustomExceptions.MedicineOutOfStockException();
78                     }
79                 }
80             }
81         }
82     }
83
84     public void handleUpdate(List<Prescription__c> newList, Map<Id, Prescription__c> oldList) {
85         try {
86             for (Prescription__c pres : newList) {
87                 if (pres.Medicine__c != null && pres.Quantity__c != null) {
88                     Medicine_Inventory__c med = [
89                         SELECT Id, Quantity__c
90                         FROM Medicine_Inventory__c
91                         WHERE Id = :pres.Medicine__c
92                         LIMIT 1
93                     ];
94
95                     if (med.Quantity__c < pres.Quantity__c) {
96                         throw new CustomExceptions.MedicineOutOfStockException();
97                     }
98                 }
99             }
100        }
101    }
102
103    public void handleInsert(List<Prescription__c> newList) {
104        try {
105            for (Prescription__c pres : newList) {
106                if (pres.Medicine__c != null) {
107                    Medicine_Inventory__c med = [
108                        SELECT Id, Quantity__c
109                        FROM Medicine_Inventory__c
110                        WHERE Id = :pres.Medicine__c
111                        LIMIT 1
112                    ];
113
114                    if (med.Quantity__c < pres.Quantity__c) {
115                        throw new CustomExceptions.MedicineOutOfStockException();
116                    }
117                }
118            }
119        }
120    }
121
122    public void handleDelete(List<Prescription__c> newList) {
123        try {
124            for (Prescription__c pres : newList) {
125                if (pres.Medicine__c != null) {
126                    Medicine_Inventory__c med = [
127                        SELECT Id, Quantity__c
128                        FROM Medicine_Inventory__c
129                        WHERE Id = :pres.Medicine__c
130                        LIMIT 1
131                    ];
132
133                    if (med.Quantity__c < pres.Quantity__c) {
134                        throw new CustomExceptions.MedicineOutOfStockException();
135                    }
136                }
137            }
138        }
139    }
140
141    public void handleUpdate(List<Prescription__c> newList, Map<Id, Prescription__c> oldList) {
142        try {
143            for (Prescription__c pres : newList) {
144                if (pres.Medicine__c != null && pres.Quantity__c != null) {
145                    Medicine_Inventory__c med = [
146                        SELECT Id, Quantity__c
147                        FROM Medicine_Inventory__c
148                        WHERE Id = :pres.Medicine__c
149                        LIMIT 1
150                    ];
151
152                    if (med.Quantity__c < pres.Quantity__c) {
153                        throw new CustomExceptions.MedicineOutOfStockException();
154                    }
155                }
156            }
157        }
158    }
159
160    public void handleInsert(List<Prescription__c> newList) {
161        try {
162            for (Prescription__c pres : newList) {
163                if (pres.Medicine__c != null) {
164                    Medicine_Inventory__c med = [
165                        SELECT Id, Quantity__c
166                        FROM Medicine_Inventory__c
167                        WHERE Id = :pres.Medicine__c
168                        LIMIT 1
169                    ];
170
171                    if (med.Quantity__c < pres.Quantity__c) {
172                        throw new CustomExceptions.MedicineOutOfStockException();
173                    }
174                }
175            }
176        }
177    }
178
179    public void handleDelete(List<Prescription__c> newList) {
180        try {
181            for (Prescription__c pres : newList) {
182                if (pres.Medicine__c != null) {
183                    Medicine_Inventory__c med = [
184                        SELECT Id, Quantity__c
185                        FROM Medicine_Inventory__c
186                        WHERE Id = :pres.Medicine__c
187                        LIMIT 1
188                    ];
189
190                    if (med.Quantity__c < pres.Quantity__c) {
191                        throw new CustomExceptions.MedicineOutOfStockException();
192                    }
193                }
194            }
195        }
196    }
197
198    public void handleUpdate(List<Prescription__c> newList, Map<Id, Prescription__c> oldList) {
199        try {
200            for (Prescription__c pres : newList) {
201                if (pres.Medicine__c != null && pres.Quantity__c != null) {
202                    Medicine_Inventory__c med = [
203                        SELECT Id, Quantity__c
204                        FROM Medicine_Inventory__c
205                        WHERE Id = :pres.Medicine__c
206                        LIMIT 1
207                    ];
208
209                    if (med.Quantity__c < pres.Quantity__c) {
210                        throw new CustomExceptions.MedicineOutOfStockException();
211                    }
212                }
213            }
214        }
215    }
216
217    public void handleInsert(List<Prescription__c> newList) {
218        try {
219            for (Prescription__c pres : newList) {
220                if (pres.Medicine__c != null) {
221                    Medicine_Inventory__c med = [
222                        SELECT Id, Quantity__c
223                        FROM Medicine_Inventory__c
224                        WHERE Id = :pres.Medicine__c
225                        LIMIT 1
226                    ];
227
228                    if (med.Quantity__c < pres.Quantity__c) {
229                        throw new CustomExceptions.MedicineOutOfStockException();
230                    }
231                }
232            }
233        }
234    }
235
236    public void handleDelete(List<Prescription__c> newList) {
237        try {
238            for (Prescription__c pres : newList) {
239                if (pres.Medicine__c != null) {
240                    Medicine_Inventory__c med = [
241                        SELECT Id, Quantity__c
242                        FROM Medicine_Inventory__c
243                        WHERE Id = :pres.Medicine__c
244                        LIMIT 1
245                    ];
246
247                    if (med.Quantity__c < pres.Quantity__c) {
248                        throw new CustomExceptions.MedicineOutOfStockException();
249                    }
250                }
251            }
252        }
253    }
254
255    public void handleUpdate(List<Prescription__c> newList, Map<Id, Prescription__c> oldList) {
256        try {
257            for (Prescription__c pres : newList) {
258                if (pres.Medicine__c != null && pres.Quantity__c != null) {
259                    Medicine_Inventory__c med = [
260                        SELECT Id, Quantity__c
261                        FROM Medicine_Inventory__c
262                        WHERE Id = :pres.Medicine__c
263                        LIMIT 1
264                    ];
265
266                    if (med.Quantity__c < pres.Quantity__c) {
267                        throw new CustomExceptions.MedicineOutOfStockException();
268                    }
269                }
270            }
271        }
272    }
273
274    public void handleInsert(List<Prescription__c> newList) {
275        try {
276            for (Prescription__c pres : newList) {
277                if (pres.Medicine__c != null) {
278                    Medicine_Inventory__c med = [
279                        SELECT Id, Quantity__c
280                        FROM Medicine_Inventory__c
281                        WHERE Id = :pres.Medicine__c
282                        LIMIT 1
283                    ];
284
285                    if (med.Quantity__c < pres.Quantity__c) {
286                        throw new CustomExceptions.MedicineOutOfStockException();
287                    }
288                }
289            }
290        }
291    }
292
293    public void handleDelete(List<Prescription__c> newList) {
294        try {
295            for (Prescription__c pres : newList) {
296                if (pres.Medicine__c != null) {
297                    Medicine_Inventory__c med = [
298                        SELECT Id, Quantity__c
299                        FROM Medicine_Inventory__c
300                        WHERE Id = :pres.Medicine__c
301                        LIMIT 1
302                    ];
303
304                    if (med.Quantity__c < pres.Quantity__c) {
305                        throw new CustomExceptions.MedicineOutOfStockException();
306                    }
307                }
308            }
309        }
310    }
311
312    public void handleUpdate(List<Prescription__c> newList, Map<Id, Prescription__c> oldList) {
313        try {
314            for (Prescription__c pres : newList) {
315                if (pres.Medicine__c != null && pres.Quantity__c != null) {
316                    Medicine_Inventory__c med = [
317                        SELECT Id, Quantity__c
318                        FROM Medicine_Inventory__c
319                        WHERE Id = :pres.Medicine__c
320                        LIMIT 1
321                    ];
322
323                    if (med.Quantity__c < pres.Quantity__c) {
324                        throw new CustomExceptions.MedicineOutOfStockException();
325                    }
326                }
327            }
328        }
329    }
330
331    public void handleInsert(List<Prescription__c> newList) {
332        try {
333            for (Prescription__c pres : newList) {
334                if (pres.Medicine__c != null) {
335                    Medicine_Inventory__c med = [
336                        SELECT Id, Quantity__c
337                        FROM Medicine_Inventory__c
338                        WHERE Id = :pres.Medicine__c
339                        LIMIT 1
340                    ];
341
342                    if (med.Quantity__c < pres.Quantity__c) {
343                        throw new CustomExceptions.MedicineOutOfStockException();
344                    }
345                }
346            }
347        }
348    }
349
350    public void handleDelete(List<Prescription__c> newList) {
351        try {
352            for (Prescription__c pres : newList) {
353                if (pres.Medicine__c != null) {
354                    Medicine_Inventory__c med = [
355                        SELECT Id, Quantity__c
356                        FROM Medicine_Inventory__c
357                        WHERE Id = :pres.Medicine__c
358                        LIMIT 1
359                    ];
360
361                    if (med.Quantity__c < pres.Quantity__c) {
362                        throw new CustomExceptions.MedicineOutOfStockException();
363                    }
364                }
365            }
366        }
367    }
368
369    public void handleUpdate(List<Prescription__c> newList, Map<Id, Prescription__c> oldList) {
370        try {
371            for (Prescription__c pres : newList) {
372                if (pres.Medicine__c != null && pres.Quantity__c != null) {
373                    Medicine_Inventory__c med = [
374                        SELECT Id, Quantity__c
375                        FROM Medicine_Inventory__c
376                        WHERE Id = :pres.Medicine__c
377                        LIMIT 1
378                    ];
379
380                    if (med.Quantity__c < pres.Quantity__c) {
381                        throw new CustomExceptions.MedicineOutOfStockException();
382                    }
383                }
384            }
385        }
386    }
387
388    public void handleInsert(List<Prescription__c> newList) {
389        try {
390            for (Prescription__c pres : newList) {
391                if (pres.Medicine__c != null) {
392                    Medicine_Inventory__c med = [
393                        SELECT Id, Quantity__c
394                        FROM Medicine_Inventory__c
395                        WHERE Id = :pres.Medicine__c
396                        LIMIT 1
397                    ];
398
399                    if (med.Quantity__c < pres.Quantity__c) {
400                        throw new CustomExceptions.MedicineOutOfStockException();
401                    }
402                }
403            }
404        }
405    }
406
407    public void handleDelete(List<Prescription__c> newList) {
408        try {
409            for (Prescription__c pres : newList) {
410                if (pres.Medicine__c != null) {
411                    Medicine_Inventory__c med = [
412                        SELECT Id, Quantity__c
413                        FROM Medicine_Inventory__c
414                        WHERE Id = :pres.Medicine__c
415                        LIMIT 1
416                    ];
417
418                    if (med.Quantity__c < pres.Quantity__c) {
419                        throw new CustomExceptions.MedicineOutOfStockException();
420                    }
421                }
422            }
423        }
424    }
425
426    public void handleUpdate(List<Prescription__c> newList, Map<Id, Prescription__c> oldList) {
427        try {
428            for (Prescription__c pres : newList) {
429                if (pres.Medicine__c != null && pres.Quantity__c != null) {
430                    Medicine_Inventory__c med = [
431                        SELECT Id, Quantity__c
432                        FROM Medicine_Inventory__c
433                        WHERE Id = :pres.Medicine__c
434                        LIMIT 1
435                    ];
436
437                    if (med.Quantity__c < pres.Quantity__c) {
438                        throw new CustomExceptions.MedicineOutOfStockException();
439                    }
440                }
441            }
442        }
443    }
444
445    public void handleInsert(List<Prescription__c> newList) {
446        try {
447            for (Prescription__c pres : newList) {
448                if (pres.Medicine__c != null) {
449                    Medicine_Inventory__c med = [
450                        SELECT Id, Quantity__c
451                        FROM Medicine_Inventory__c
452                        WHERE Id = :pres.Medicine__c
453                        LIMIT 1
454                    ];
455
456                    if (med.Quantity__c < pres.Quantity__c) {
457                        throw new CustomExceptions.MedicineOutOfStockException();
458                    }
459                }
460            }
461        }
462    }
463
464    public void handleDelete(List<Prescription__c> newList) {
465        try {
466            for (Prescription__c pres : newList) {
467                if (pres.Medicine__c != null) {
468                    Medicine_Inventory__c med = [
469                        SELECT Id, Quantity__c
470                        FROM Medicine_Inventory__c
471                        WHERE Id = :pres.Medicine__c
472                        LIMIT 1
473                    ];
474
475                    if (med.Quantity__c < pres.Quantity__c) {
476                        throw new CustomExceptions.MedicineOutOfStockException();
477                    }
478                }
479            }
480        }
481    }
482
483    public void handleUpdate(List<Prescription__c> newList, Map<Id, Prescription__c> oldList) {
484        try {
485            for (Prescription__c pres : newList) {
486                if (pres.Medicine__c != null && pres.Quantity__c != null) {
487                    Medicine_Inventory__c med = [
488                        SELECT Id, Quantity__c
489                        FROM Medicine_Inventory__c
490                        WHERE Id = :pres.Medicine__c
491                        LIMIT 1
492                    ];
493
494                    if (med.Quantity__c < pres.Quantity__c) {
495                        throw new CustomExceptions.MedicineOutOfStockException();
496                    }
497                }
498            }
499        }
500    }
501
502    public void handleInsert(List<Prescription__c> newList) {
503        try {
504            for (Prescription__c pres : newList) {
505                if (pres.Medicine__c != null) {
506                    Medicine_Inventory__c med = [
507                        SELECT Id, Quantity__c
508                        FROM Medicine_Inventory__c
509                        WHERE Id = :pres.Medicine__c
510                        LIMIT 1
511                    ];
512
513                    if (med.Quantity__c < pres.Quantity__c) {
514                        throw new CustomExceptions.MedicineOutOfStockException();
515                    }
516                }
517            }
518        }
519    }
520
521    public void handleDelete(List<Prescription__c> newList) {
522        try {
523            for (Prescription__c pres : newList) {
524                if (pres.Medicine__c != null) {
525                    Medicine_Inventory__c med = [
526                        SELECT Id, Quantity__c
527                        FROM Medicine_Inventory__c
528                        WHERE Id = :pres.Medicine__c
529                        LIMIT 1
530                    ];
531
532                    if (med.Quantity__c < pres.Quantity__c) {
533                        throw new CustomExceptions.MedicineOutOfStockException();
534                    }
535                }
536            }
537        }
538    }
539
540    public void handleUpdate(List<Prescription__c> newList, Map<Id, Prescription__c> oldList) {
541        try {
542            for (Prescription__c pres : newList) {
543                if (pres.Medicine__c != null && pres.Quantity__c != null) {
544                    Medicine_Inventory__c med = [
545                        SELECT Id, Quantity__c
546                        FROM Medicine_Inventory__c
547                        WHERE Id = :pres.Medicine__c
548                        LIMIT 1
549                    ];
550
551                    if (med.Quantity__c < pres.Quantity__c) {
552                        throw new CustomExceptions.MedicineOutOfStockException();
553                    }
554                }
555            }
556        }
557    }
558
559    public void handleInsert(List<Prescription__c> newList) {
560        try {
561            for (Prescription__c pres : newList) {
562                if (pres.Medicine__c != null) {
563                    Medicine_Inventory__c med = [
564                        SELECT Id, Quantity__c
565                        FROM Medicine_Inventory__c
566                        WHERE Id = :pres.Medicine__c
567                        LIMIT 1
568                    ];
569
570                    if (med.Quantity__c < pres.Quantity__c) {
571                        throw new CustomExceptions.MedicineOutOfStockException();
572                    }
573                }
574            }
575        }
576    }
577
578    public void handleDelete(List<Prescription__c> newList) {
579        try {
580            for (Prescription__c pres : newList) {
581                if (pres.Medicine__c != null) {
582                    Medicine_Inventory__c med = [
583                        SELECT Id, Quantity__c
584                        FROM Medicine_Inventory__c
585                        WHERE Id = :pres.Medicine__c
586                        LIMIT 1
587                    ];
588
589                    if (med.Quantity__c < pres.Quantity__c) {
590                        throw new CustomExceptions.MedicineOutOfStockException();
591                    }
592                }
593            }
594        }
595    }
596
597    public void handleUpdate(List<Prescription__c> newList, Map<Id, Prescription__c> oldList) {
598        try {
599            for (Prescription__c pres : newList) {
600                if (pres.Medicine__c != null && pres.Quantity__c != null) {
601                    Medicine_Inventory__c med = [
602                        SELECT Id, Quantity__c
603                        FROM Medicine_Inventory__c
604                        WHERE Id = :pres.Medicine__c
605                        LIMIT 1
606                    ];
607
608                    if (med.Quantity__c < pres.Quantity__c) {
609                        throw new CustomExceptions.MedicineOutOfStockException();
610                    }
611                }
612            }
613        }
614    }
615
616    public void handleInsert(List<Prescription__c> newList) {
617        try {
618            for (Prescription__c pres : newList) {
619                if (pres.Medicine__c != null) {
620                    Medicine_Inventory__c med = [
621                        SELECT Id, Quantity__c
622                        FROM Medicine_Inventory__c
623                        WHERE Id = :pres.Medicine__c
624                        LIMIT 1
625                    ];
626
627                    if (med.Quantity__c < pres.Quantity__c) {
628                        throw new CustomExceptions.MedicineOutOfStockException();
629                    }
630                }
631            }
632        }
633    }
634
635    public void handleDelete(List<Prescription__c> newList) {
636        try {
637            for (Prescription__c pres : newList) {
638                if (pres.Medicine__c != null) {
639                    Medicine_Inventory__c med = [
640                        SELECT Id, Quantity__c
641                        FROM Medicine_Inventory__c
642                        WHERE Id = :pres.Medicine__c
643                        LIMIT 1
644                    ];
645
646                    if (med.Quantity__c < pres.Quantity__c) {
647                        throw new CustomExceptions.MedicineOutOfStockException();
648                    }
649                }
650            }
651        }
652    }
653
654    public void handleUpdate(List<Prescription__c> newList, Map<Id, Prescription__c> oldList) {
655        try {
656            for (Prescription__c pres : newList) {
657                if (pres.Medicine__c != null && pres.Quantity__c != null) {
658                    Medicine_Inventory__c med = [
659                        SELECT Id, Quantity__c
660                        FROM Medicine_Inventory__c
661                        WHERE Id = :pres.Medicine__c
662                        LIMIT 1
663                    ];
664
665                    if (med.Quantity__c < pres.Quantity__c) {
666                        throw new CustomExceptions.MedicineOutOfStockException();
667                    }
668                }
669            }
670        }
671    }
672
673    public void handleInsert(List<Prescription__c> newList) {
674        try {
675            for (Prescription__c pres : newList) {
676                if (pres.Medicine__c != null) {
677                    Medicine_Inventory__c med = [
678                        SELECT Id, Quantity__c
679                        FROM Medicine_Inventory__c
680                        WHERE Id = :pres.Medicine__c
681                        LIMIT 1
682                    ];
683
684                    if (med.Quantity__c < pres.Quantity__c) {
685                        throw new CustomExceptions.MedicineOutOfStockException();
686                    }
687                }
688            }
689        }
690    }
691
692    public void handleDelete(List<Prescription__c> newList) {
693        try {
694            for (Prescription__c pres : newList) {
695                if (pres.Medicine__c != null) {
696                    Medicine_Inventory__c med = [
697                        SELECT Id, Quantity__c
698                        FROM Medicine_Inventory__c
699                        WHERE Id = :pres.Medicine__c
700                        LIMIT 1
701                    ];
702
703                    if (med.Quantity__c < pres.Quantity__c) {
704                        throw new CustomExceptions.MedicineOutOfStockException();
705                    }
706                }
707            }
708        }
709    }
710
711    public void handleUpdate(List<Prescription__c> newList, Map<Id, Prescription__c> oldList) {
712        try {
713            for (Prescription__c pres : newList) {
714                if (pres.Medicine__c != null && pres.Quantity__c != null) {
715                    Medicine_Inventory__c med = [
716                        SELECT Id, Quantity__c
717                        FROM Medicine_Inventory__c
718                        WHERE Id = :pres.Medicine__c
719                        LIMIT 1
720                    ];
721
722                    if (med.Quantity__c < pres.Quantity__c) {
723                        throw new CustomExceptions.MedicineOutOfStockException();
724                    }
725                }
726            }
727        }
728    }
729
730    public void handleInsert(List<Prescription__c> newList) {
731        try {
732            for (Prescription__c pres : newList) {
733                if (pres.Medicine__c != null) {
734                    Medicine_Inventory__c med = [
735                        SELECT Id, Quantity__c
736                        FROM Medicine_Inventory__c
737                        WHERE Id = :pres.Medicine__c
738                        LIMIT 1
739                    ];
740
741                    if (med.Quantity__c < pres.Quantity__c) {
742                        throw new CustomExceptions.MedicineOutOfStockException();
743                    }
744                }
745            }
746        }
747    }
748
749    public void handleDelete(List<Prescription__c> newList) {
750        try {
751            for (Prescription__c pres : newList) {
752                if (pres.Medicine__c != null) {
753                    Medicine_Inventory__c med = [
754                        SELECT Id, Quantity__c
755                        FROM Medicine_Inventory__c
756                        WHERE Id = :pres.Medicine__c
757                        LIMIT 1
758                    ];
759
760                    if (med.Quantity__c < pres.Quantity__c) {
761                        throw new CustomExceptions.MedicineOutOfStockException();
762                    }
763                }
764            }
765        }
766    }
767
768    public void handleUpdate(List<Prescription__c> newList, Map<Id, Prescription__c> oldList) {
769        try {
770            for (Prescription__c pres : newList) {
771                if (pres.Medicine__c != null && pres.Quantity__c != null) {
772                    Medicine_Inventory__c med = [
773                        SELECT Id, Quantity__c
774                        FROM Medicine_Inventory__c
775                        WHERE Id = :pres.Medicine__c
776                        LIMIT 1
777                    ];
778
779                    if (med.Quantity__c < pres.Quantity__c) {
780                        throw new CustomExceptions.MedicineOutOfStockException();
781                    }
782                }
783            }
784        }
785    }
786
787    public void handleInsert(List<Prescription__c> newList) {
788        try {
789            for (Prescription__c pres : newList) {
790                if (pres.Medicine__c != null) {
791                    Medicine_Inventory__c med = [
792                        SELECT Id, Quantity__c
793                        FROM Medicine_Inventory__c
794                        WHERE Id = :pres.Medicine__c
795                        LIMIT 1
796                    ];
797
798                    if (med.Quantity__c < pres.Quantity__c) {
799                        throw new CustomExceptions.MedicineOutOfStockException();
800                    }
801                }
802            }
803        }
804    }
805
806    public void handleDelete(List<Prescription__c> newList) {
807        try {
808            for (Prescription__c pres : newList) {
809                if (pres.Medicine__c != null) {
810                    Medicine_Inventory__c med = [
811                        SELECT Id, Quantity__c
812                        FROM Medicine_Inventory__c
813                        WHERE Id = :pres.Medicine__c
814                        LIMIT 1
815                    ];
816
817                    if (med.Quantity__c < pres.Quantity__c) {
818                        throw new CustomExceptions.MedicineOutOfStockException();
819                    }
820                }
821            }
822        }
823    }
824
825    public void handleUpdate(List<Prescription__c> newList, Map<Id, Prescription__c> oldList) {
826        try {
827            for (Prescription__c pres : newList) {
828                if (pres.Medicine__c != null && pres.Quantity__c != null) {
829                    Medicine_Inventory__c med = [
830                        SELECT Id, Quantity__c
831                        FROM Medicine_Inventory__c
832                        WHERE Id = :pres.Medicine__c
833                        LIMIT 1
834                    ];
835
836                    if (med.Quantity__c < pres.Quantity__c) {
837                        throw new CustomExceptions.MedicineOutOfStockException();
838                    }
839                }
840            }
841        }
842    }
843
844    public void handleInsert(List<Prescription__c> newList) {
845        try {
846            for (Prescription__c pres : newList) {
847                if (pres.Medicine__c != null) {
848                    Medicine_Inventory__c med = [
849                        SELECT Id, Quantity__c
850                        FROM Medicine_Inventory__c
851                        WHERE Id = :pres.Medicine__c
852                        LIMIT 1
853                    ];
854
855                    if (med.Quantity__c < pres.Quantity__c) {
856                        throw new CustomExceptions.MedicineOutOfStockException();
857                    }
858                }
859            }
860        }
861    }
862
863    public void handleDelete(List<Prescription__c> newList) {
864        try {
865            for (Prescription__c pres : newList) {
866                if (pres.Medicine__c != null) {
867                    Medicine_Inventory__c med = [
868                        SELECT Id, Quantity__c
869                        FROM Medicine_Inventory__c
870                        WHERE Id = :pres.Medicine__c
871                        LIMIT 1
872                    ];
873
874                    if (med.Quantity__c < pres.Quantity__c) {
875                        throw new CustomExceptions.MedicineOutOfStockException();
876                    }
877                }
878            }
879        }
880    }
881
882    public void handleUpdate(List<Prescription__c> newList, Map<Id, Prescription__c> oldList) {
883        try {
884            for (Prescription__c pres : newList) {
885                if (pres.Medicine__c != null && pres.Quantity__c != null) {
886                    Medicine_Inventory__c med = [
887                        SELECT Id, Quantity__c
888                        FROM Medicine_Inventory__c
889                        WHERE Id = :pres.Medicine__c
890                        LIMIT 1
891                    ];
892
893                    if (med.Quantity__c < pres.Quantity__c) {
894                        throw new CustomExceptions.MedicineOutOfStockException();
895                    }
896                }
897            }
898        }
899    }
900
901    public void handleInsert(List<Prescription__c> newList) {
902        try {
903            for (Prescription__c pres : newList) {
904                if (pres.Medicine__c != null) {
905                    Medicine_Inventory__c med = [
906                        SELECT Id, Quantity__c
907                        FROM Medicine_Inventory__c
908                        WHERE Id = :pres.Medicine__c
909                        LIMIT 1
910                    ];
911
912                    if (med.Quantity__c < pres.Quantity__c) {
913                        throw new CustomExceptions.MedicineOutOfStockException();
914                    }
915                }
916            }
917        }
918    }
919
920    public void handleDelete(List<Prescription__c> newList) {
921        try {
922            for (Prescription__c pres : newList) {
923                if (pres.Medicine__c != null) {
924                    Medicine_Inventory__c med = [
925                        SELECT Id, Quantity__c
926                        FROM Medicine_Inventory__c
927                        WHERE Id = :pres.Medicine__c
928                        LIMIT 1
929                    ];
930
931                    if (med.Quantity__c < pres.Quantity__c) {
932                        throw new CustomExceptions.MedicineOutOfStockException();
933                    }
934                }
935            }
936        }
937    }
938
939    public void handleUpdate(List<Prescription__c> newList, Map<Id, Prescription__c> oldList) {
940        try {
941            for (Prescription__c pres : newList) {
942                if (pres.Medicine__c != null && pres.Quantity__c != null) {
943                    Medicine_Inventory__c med = [
944                        SELECT Id, Quantity__c
945                        FROM Medicine_Inventory__c
946                        WHERE Id = :pres.Medicine__c
947                        LIMIT 1
948                    ];
949
950                    if (med.Quantity__c < pres.Quantity__c) {
951                        throw new CustomExceptions.MedicineOutOfStockException();
952                    }
953                }
954            }
955        }
956    }
957
958    public void handleInsert(List<Prescription__c> newList) {
959        try {
960            for (Prescription__c pres : newList) {
961                if (pres.Medicine__c != null) {
962                    Medicine_Inventory__c med = [
963                        SELECT Id, Quantity__c
964                        FROM Medicine_Inventory__c
965                        WHERE Id = :pres.Medicine__c
966                        LIMIT 1
967                    ];
968
969                    if (med.Quantity__c < pres.Quantity__c) {
970                        throw new CustomExceptions.MedicineOutOfStockException();
971                    }
972                }
973            }
974        }
975    }
976
977    public void handleDelete(List<Prescription__c> newList) {
978        try {
979            for (Prescription__c pres : newList) {
980                if (pres.Medicine__c != null) {
981                    Medicine_Inventory__c med = [
982                        SELECT Id, Quantity__c
983                        FROM Medicine_Inventory__c
984                        WHERE Id = :pres.Medicine__c
985                        LIMIT 1
986                    ];
987
988                    if (med.Quantity__c < pres.Quantity__c) {
989                        throw new CustomExceptions.MedicineOutOfStockException();
990                    }
991                }
992            }
993        }
994    }
995
996    public void handleUpdate(List<Prescription__c> newList, Map<Id, Prescription__c> oldList) {
997        try {
998            for (Prescription__c pres : newList) {
999                if (pres.Medicine__c != null && pres.Quantity__c != null) {
1000                    Medicine_Inventory__c med = [
1001                        SELECT Id, Quantity__c
1002                        FROM Medicine_Inventory__c
1003                        WHERE Id = :pres.Medicine__c
1004                        LIMIT 1
1005                    ];
1006
1007                    if (med.Quantity__c < pres.Quantity__c) {
1008                        throw new CustomExceptions.MedicineOutOfStockException();
1009                    }
1010                }
1011            }
1012        }
1013    }
1014
1015    public void handleInsert(List<Prescription__c> newList) {
1016        try {
1017            for (Prescription__c pres : newList) {
1018                if (pres.Medicine__c != null) {
1019                    Medicine_Inventory__c med = [
1020                        SELECT Id, Quantity__c
1021                        FROM Medicine_Inventory__c
1022                        WHERE Id = :pres.Medicine__c
1023                        LIMIT 1
1024                    ];
1025
1026                    if (med.Quantity__c < pres.Quantity__c) {
1027                        throw new CustomExceptions.MedicineOutOfStockException();
1028                    }
1029                }
1030            }
1031        }
1032    }
1033
1034    public void handleDelete(List<Prescription__c> newList) {
1035        try {
1036            for (Prescription__c pres : newList) {
1037                if (pres.Medicine__c != null) {
1038                    Medicine_Inventory__c med = [
1039                        SELECT Id, Quantity__c
1040                        FROM Medicine_Inventory__c
1041                        WHERE Id = :pres.Medicine__c
1042                        LIMIT 1
1043                    ];
1044
1045                    if (med.Quantity__c < pres.Quantity__c) {
1046                        throw new CustomExceptions.MedicineOutOfStockException();
1047                    }
1048                }
1049            }
1050        }
1051    }
1052
1053    public void handleUpdate(List<Prescription__c> newList, Map<Id, Prescription__c> oldList) {
1054        try {
1055            for (Prescription__c pres : newList) {
1056                if (pres.Medicine__c != null && pres.Quantity__c != null) {
1057                    Medicine_Inventory__c med = [
1058                        SELECT Id, Quantity__c
1059                        FROM Medicine_Inventory__c
1060                        WHERE Id = :pres.Medicine__c
1061                        LIMIT 1
1062                    ];
1063
1064                    if (med.Quantity__c < pres.Quantity__c) {
1065                        throw new CustomExceptions.MedicineOutOfStockException();
1066                    }
1067                }
1068            }
1069        }
1070    }
1071
1072    public void handleInsert(List<Prescription__c> newList) {
1073        try {
1074            for (Prescription__c pres : newList) {
1075                if (pres.Medicine__c != null) {
1076                    Medicine_Inventory__c med = [
1077                        SELECT Id, Quantity__c
1078                        FROM Medicine_Inventory__c
1079                        WHERE Id = :pres.Medicine__c
1080                        LIMIT 1
1081                    ];
1082
1083                    if (med.Quantity__c < pres.Quantity__c) {
1084                        throw new CustomExceptions.MedicineOutOfStockException();
1085                    }
1086                }
1087            }
1088        }
108
```

➤ Collections

- List : To store multiple appointments in bulk insert.
- Set : To ensure no duplicate patient IDs when scheduling.
- Map : DoctorId
 - List of Appointments (used for bulk assignment logic).

It is present in Trigger + batch class.

➤ Control Statements

- IF-ELSE for appointment conflict detection.
- FOR loops for iterating through bulk patient records.
- SWITCH statements for handling medicine stock statuses.

Used in if/else checks inside handlers + services

➤ Batch Apex

Used for “large-volume data processing”.

- AppointmentArchiveBatch class created.

```
• AppointmentArchiveBatch.cls U X
nabha > force-app > main > default > classes > • AppointmentArchiveBatch.cls > ...
1 global class AppointmentArchiveBatch implements Database.Batchable<sObject>, Database.Stateful {
2     global Database.QueryLocator start(Database.BatchableContext bc) {
3         Datetime cutoff = System.now().addYears(-1);
4         return Database.getQueryLocator([SELECT Id FROM Appointment__c WHERE Appointment_DateTime__c < :cutoff AND (Archived__c = false OR Archived__c = true)]);
5     }
6     global void execute(Database.BatchableContext bc, List<sobject> scope){
7         List<Appointment__c> toUpdate = new List<Appointment__c>();
8         for(sObject s : scope){
9             Appointment__c a = (Appointment__c)s;
10            a.Archived__c = true;
11            toUpdate.add(a);
12        }
13        if(!toUpdate.isEmpty()) update toUpdate;
14    }
15    global void finish(Database.BatchableContext bc){
16        System.debug('AppointmentArchiveBatch finished.');
17    }
18 }
```

➤ Queueable Apex

Used for “chained async operations”.

- AppointmentNotificationQueueable

```
► AppointmentNotificationQueueable.cls U X
nabha > force-app > main > default > classes > ► AppointmentNotificationQueueable.cls > ...
1  public with sharing class AppointmentNotificationQueueable implements Queueable {
2      private List<Id> appointmentIds;
3
4      // Constructor takes list of Appointment IDs
5      public AppointmentNotificationQueueable(List<Id> appointmentIds){
6          this.appointmentIds = appointmentIds;
7      }
8
9      public void execute(QueueableContext qc){
10         try {
11             if(appointmentIds == null || appointmentIds.isEmpty()) return;
12
13             // Fetch appointments with patient details
14             List<Appointment__c> appts = [
15                 SELECT Id, Appointment_DateTime__c, Patient__c,
16                 | Patient__r.Name, Patient__r.Phone__c, Patient__r.Email__c
17                 FROM Appointment__c
18                 WHERE Id IN :appointmentIds
19             ];
20
21             List<String> phones = new List<String>();
22
23             for(Appointment__c a : appts){
24                 String msg = 'Dear ' + (a.Patient__r != null ? a.Patient__r.Name : 'Patient') +
25                 | | | , your appointment is scheduled on ' +
26                 | | | String.valueOf(a.Appointment_DateTime__c) + '.';
27
28                 // Collect phones for SMS
29                 if(a.Patient__r != null && a.Patient__r.Phone__c != null) {
30                     phones.add(a.Patient__r.Phone__c);
31                 }
32
33                 // Send Email
34                 if(a.Patient__r != null && a.Patient__r.Email__c != null){
35                     Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
36                     mail.setToAddresses(new String[]{a.Patient__r.Email__c});
37                     mail.setSubject('Appointment Scheduled');
38                 }
39             }
40         }
41     }
42 }
```

- SendPrescriptionQueueable

```
► SendPrescriptionQueueable.cls U X
nabha > force-app > main > default > classes > ► SendPrescriptionQueueable.cls > ...
1  public with sharing class SendPrescriptionQueueable implements Queueable {
2      private Id prescId;
3
4      public SendPrescriptionQueueable(Id prescId){
5          this.precId = prescId;
6      }
7
8      public void execute(QueueableContext qc){
9          try {
10             Prescription__c p = [SELECT Id, Notes__c, Appointment__c, Appointment__r.Patient__r.Email__c FROM Prescription__c WHERE Id = :prescId];
11             if(p != null && p.Appointment__r != null && p.Appointment__r.Patient__r.Email__c != null){
12                 Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
13                 mail.setToAddresses(new String[]{p.Appointment__r.Patient__r.Email__c});
14                 mail.setSubject('Your Prescription from Telemedicine Portal');
15                 mail.setPlainTextBody('Dear Patient,\n\nPlease find your prescription notes:\n' + p.Notes__c + '\n\nRegards');
16                 Messaging.sendEmail(new Messaging.SingleEmailMessage[]{mail});
17             }
18         } catch(Exception e){
19             ErrorHandler.logException(e, 'SendPrescriptionQueueable.execute');
20         }
21     }
22 }
```

➤ Scheduled Apex

Nightly jobs to sync external pharmacy data (via Salesforce Connect).

Example: **PharmacySyncScheduler**

→ Runs every night at 2 AM.

Created: **AppointmentArchiveSched** for the same.

The screenshot shows the code for the **AppointmentArchiveSched** class. The code is as follows:

```
nabha > force-app > main > default > classes > AppointmentArchiveSched.cls > ...
1  global class AppointmentArchiveSched implements Schedulable {
2      global void execute(SchedulableContext sc){
3          Database.executeBatch(new AppointmentArchiveBatch(), 200);
4      }
5  }
```

➤ Future Methods

- Used for lightweight async tasks.

Example: @future method for sending SMS confirmation to patients after booking.

➤ Exception Handling

- Custom exception classes created like **AppointmentConflictException** and **MedicineOutOfStockException**.

The screenshot shows the code for the **CustomExceptions** class. The code is as follows:

```
nabha > force-app > main > default > classes > CustomExceptions.cls > ...
1  public class CustomExceptions {
2      public class AppointmentConflictException extends Exception {}
3      public class MedicineOutOfStockException extends Exception {}
4  }
```

- All Apex code wrapped with try...catch blocks and errors logged to a custom object **Error_Log__c**.

The screenshot shows the code for the **ErrorHandler** class. The code is as follows:

```
nabha > force-app > main > default > classes > ErrorHandler.cls > ...
1  public with sharing class ErrorHandler {
2      public static void logException(Exception e, String contextMessage) {
3          try {
4              Error_Log__c logRec = new Error_Log__c();
5              logRec.Message__c = e.getMessage();
6              logRec.Stack_Trace__c = e.getStackTraceString();
7              logRec.Context__c = contextMessage;
8              insert logRec;
9          } catch (Exception ex) {
10              System.debug('ErrorHandler failed to log: ' + ex.getMessage());
11          }
12      }
13  }
```

➤ Test Classes

TelemedicineApexTests ensures:

- Double-booking is prevented.
- Notifications are sent after booking.
- Medicine inventory updates correctly.

```
● TelemedicineApexTests.cls U 
nabha > force-app > main > default > classes > ● TelemedicineApexTests.cls > 📈 TelemedicineApexTests > ⚙ testAppointmentInsert_and_notification(): void
1  @isTest
2  private class TelemedicineApexTests {
3      @isTest static void testAppointmentInsert_and_notification() {
4          // Create patient
5          Patient__c p = new Patient__c(Name__c='Test Patient', Aadhaar_Number__c='123456789012', Status__c='New', Phone__c='9999999999', Email__c='testp
6          insert p;
7          // Create doctor
8          Doctor__c d = new Doctor__c(Name__c='Dr Test', Specialization__c='General');
9          insert d;
10         // Create appointment
11         Test.startTest();
12         Appointment__c a = new Appointment__c(Patient__c = p.Id, Doctor__c = d.Id, Appointment_DateTime__c = System.now().addDays(2), Mode__c='V
13         insert a;
14         Test.stopTest();
15         Appointment__c fetched = [SELECT Id, Status__c FROM Appointment__c WHERE Id = :a.Id];
16         System.assertEquals('Scheduled', fetched.Status__c);
17     }
18
19     @isTest static void testPrescription_deducts_stock() {
20         // Create medicine
21         Medicine_Inventory__c m = new Medicine_Inventory__c(Medicine_Name__c='Para', Quantity__c=50);
22         insert m;
23         // Create patient/doctor/appointment
24         Patient__c p = new Patient__c(Name__c='Pat2', Aadhaar_Number__c='111111111111', Status__c='New', Email__c='p2@example.com');
25         insert p;
26         Doctor__c d = new Doctor__c(Name__c='Dr Two', Specialization__c='General');
27         insert d;
28         Appointment__c a = new Appointment__c(Patient__c=p.Id, Doctor__c=d.Id, Appointment_DateTime__c=System.now().addDays(1), Mode__c='Video');
29         insert a;
30         Test.startTest();
31         Prescription__c pr = new Prescription__c(Appointment__c = a.Id, Medicine__c = m.Id, Quantity__c = 5, Notes__c='Take after food');
32         insert pr;
33         Test.stopTest();
34         Medicine_Inventory__c mAAfter = [SELECT Quantity__c FROM Medicine_Inventory__c WHERE Id = :m.Id];
35         System.assertEquals(45, Integer.valueOf(mAAfter.Quantity__c));
36     }
37 }
```

➤ Asynchronous Processing

- Combined “**Batch + Queueable + Scheduled**” for handling bulk jobs.
- This ensures the system works smoothly for thousands of records, supporting scalability for other rural regions.

Outcome of Phase 5

With Phase 5 complete, our project moves beyond point-and-click automation into “**scalable enterprise-level programming**”.

```
PS C:\Users\anjali\Desktop\Nabha_Telemedicine\nabha> sf project deploy start --source-dir force-app --target-org telemedicineOrg
```

Deploying Metadata

Deploying v64.0 metadata to anjali.raikwar.cs22269@agentforce.com using the v64.0 SOAP API.

✓ Preparing 466ms
() Waiting for the org to respond - skipped
✓ Deploying Metadata 5.54s
 ► Components: 83/83 (100%)
 () Running Tests - Skipped
 () Updating Source Tracking - skipped
✓ Done 0ms

Status: **Succeeded**

Deploy ID: 0AfgL00000AXdfxSAD

Target Org: anjali.raikwar.cs22269@agentforce.com

Elapsed Time: 5.99s

Deployed Source

State	Name	Type	Path
Created	AppointmentArchiveBatch	ApexClass	force-app\main\default\classes\AppointmentArchiveBatch.cls
Created	AppointmentArchiveBatch	ApexClass	force-app\main\default\classes\AppointmentArchiveBatch.cls-meta.xml
Created	AppointmentArchiveSched	ApexClass	force-app\main\default\classes\AppointmentArchiveSched.cls
Created	AppointmentArchivesched	ApexClass	force-app\main\default\classes\AppointmentArchivesched.cls-meta.xml
Created	AppointmentNotificationQueueable	ApexClass	force-app\main\default\classes\AppointmentNotificationQueueable.cls
Created	AppointmentNotificationQueueable	ApexClass	force-app\main\default\classes\AppointmentNotificationQueueable.cls-meta.xml
Created	AppointmentTriggerHandler	ApexClass	force-app\main\default\classes\AppointmentTriggerHandler.cls
Created	AppointmentTriggerHandler	ApexClass	force-app\main\default\classes\AppointmentTriggerHandler.cls-meta.xml
Unchanged	CustomExceptions	ApexClass	force-app\main\default\classes\CustomExceptions.cls
Unchanged	CustomExceptions	ApexClass	force-app\main\default\classes\CustomExceptions.cls-meta.xml
Unchanged	ErrorHandler	ApexClass	force-app\main\default\classes\ErrorHandler.cls
Unchanged	ErrorHandler	ApexClass	force-app\main\default\classes\ErrorHandler.cls-meta.xml
Unchanged	PrescriptionTriggerHandler	ApexClass	force-app\main\default\classes\PrescriptionTriggerHandler.cls
Unchanged	PrescriptionTriggerHandler	ApexClass	force-app\main\default\classes\PrescriptionTriggerHandler.cls-meta.xml
Created	PurchaseOrderService	ApexClass	force-app\main\default\classes\PurchaseOrderService.cls

Have successfully Deployed all the Apex Classes !!

SALESFORCE PROJECT IMPLEMENTATION PHASES

Phase 6: User Interface Development

In this phase, I have build the **UI layer** of my Telemedicine Salesforce project. The goal is to design intuitive Lightning Pages and extend them with **Lightning Web Components (LWC)** connected to Apex for dynamic features.

➤ Lightning App Builder

Tool: *Setup → Lightning App Builder*.

- Create custom pages for **Doctor Dashboard, Patient Portal, and Pharmacy Management.**
- Add components (Record Details, Related Lists, Charts)
- Assign to Apps, Record Types or Profiles

The screenshot shows the 'Lightning Page Detail' screen for the 'Doctor_Dashboard' page. At the top, there's a header with the page name and buttons for 'Edit', 'Clone', and 'Delete'. Below the header, there's a section titled 'Information' with fields for 'Name' (set to 'Doctor_Dashboard') and 'Label' (set to 'Doctor Dashboard'). There's also a 'Description' field and another set of 'Edit', 'Clone', and 'Delete' buttons at the bottom of the section.

The screenshot shows the 'Lightning Page Detail' screen for the 'Appointment_Record_Telemedicine' page. It has a similar structure to the first screenshot, with a header, an 'Information' section (Name: 'Appointment_Record_Telemedicine', Label: 'Appointment Record - Telemedicine'), and a detailed section at the bottom.

The screenshot shows the 'Lightning Page Detail' screen for the 'TeleMedicine' page. It follows the same pattern: a header, an 'Information' section (Name: 'TeleMedicine', Label: 'TeleMedicine'), and a detailed section at the bottom.

➤ Record Pages

Customize **Appointment__c Record Page** to show:

- Appointment Details (Time, Doctor, Patient).
- Related List for Prescription__c.
- Custom LWC for “Reschedule Appointment”.

App Details & Branding

Give your Lightning app a name and description. Upload an image and choose the highlight color for its navigation bar.

App Details

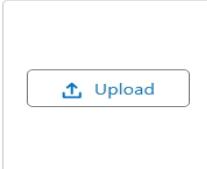
* App Name ⓘ
Telemedicine - Admin

* Developer Name ⓘ
Telemedicine_Admin

Description ⓘ
Enter a description...

App Branding

Image ⓘ



Primary Color Hex Value ⓘ

#0070D2

Org Theme Options

Use the app's image and color instead of the org's custom theme

App Launcher Preview



App Details & Branding

Give your Lightning app a name and description. Upload an image and choose the highlight color for its navigation bar.

App Details

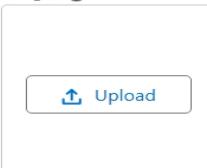
* App Name ⓘ
Telemedicine - Pharmacy

* Developer Name ⓘ
Telemedicine_Pharmacy

Description ⓘ
Enter a description...

App Branding

Image ⓘ



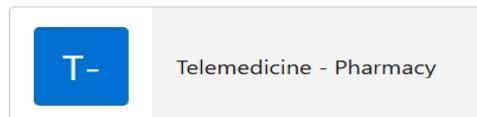
Primary Color Hex Value ⓘ

#0070D2

Org Theme Options

Use the app's image and color instead of the org's custom theme

App Launcher Preview



➤ Tabs

Add **Custom Tabs** for:

- Appointment__c
- Prescription__c
- Purchase_Order__c
- report__c
- Medicine_inventory__c
- Error_log__c

- Patient __c

Custom Object Tabs		New What Is This?
Action	Label	Tab Style
Edit Del	Appointment	 Big top
Edit Del	Error Logs	 Cup
Edit Del	Medicine Inventories	 Box
Edit Del	Mentors	 Star
Edit Del	Patients	 Big top
Edit Del	Prescriptions	 Big top
Edit Del	Purchase Orders	 Box
Edit Del	reports	 Box
Edit Del	Students	 Car

Tabs help users access records easily from the App Launcher.

➤ Home Page Layouts

Modify Home Page for Doctors:

- Component: Today's Appointments (List View).
- Component: Recent Patients.
- Dashboard (Appointments by Status).

App Details & Branding

Give your Lightning app a name and description. Upload an image and choose the highlight color for its navigation bar.

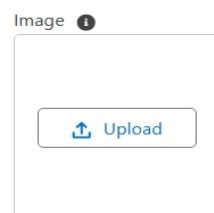
App Details

* App Name !

* Developer Name !

Description !

App Branding



Primary Color Hex Value !
 #0070D2

Org Theme Options
 Use the app's image and color instead of the org's custom theme

App Launcher Preview



Modify Home Page for Patients:

- Component: Upcoming Appointments.
- Component: My Prescriptions.

App Details & Branding

Give your Lightning app a name and description. Upload an image and choose the highlight color for its navigation bar.

App Details

* App Name i

* Developer Name i

Description i

App Branding

Image i

Primary Color Hex Value i
 #0070D2

Org Theme Options

Use the app's image and color instead of the org's custom theme

App Launcher Preview



➤ Utility Bar

Add Utility Bar in the Doctor's App:

- Quick Actions : Create Appointment
- Chat (Future Integration)
- Notes

Utility Items (Desktop Only)

Give your users quick access to productivity tools and add background utility items to your app.

Add Utility Item Utility Bar Alignment i Default

 Notes	PROPERTIES Notes	<input type="button" value="↑"/> <input type="button" value="↓"/> <input type="button" value="Remove"/>
Utility Item Properties		
Label	<input type="text" value="Notes"/>	
Icon		
Panel Width	<input type="text" value="340"/>	
Panel Height	<input type="text" value="480"/>	
<input type="checkbox"/> Start automatically		

➤ LWC (Lightning Web Components)

Custom components built for our project:

- **AppointmentCard** → Display patient + doctor appointment details.

```
appointmentCard.html U X
nabha > force-app > main > default > lwc > appointmentCard > appointmentCard.html > ...
1  <template>
2    <lightning-card title="Appointment Summary" icon-name="standard:event">
3      <div class="slds-p-horizontal_medium">
4        <p><strong>Doctor:</strong> {doctorName}</p>
5        <p><strong>Patient:</strong> {patientName}</p>
6        <p><strong>Date:</strong> {appointmentDate}</p>
7        <p><strong>Status:</strong> {status}</p>
8      </div>
9    </lightning-card>
10   </template>
11
```

```
appointmentCard.js U X
nabha > force-app > main > default > lwc > appointmentCard > appointmentCard.js > ...
1  import { LightningElement, api } from 'lwc';
2
3  export default class AppointmentCard extends LightningElement {
4    @api doctorName;
5    @api patientName;
6    @api appointmentDate;
7    @api status;
8  }
9
```

```
appointmentCard.js-meta.xml U X
nabha > force-app > main > default > lwc > appointmentCard > appointmentCard.js-meta.xml
1  <?xml version="1.0" encoding="UTF-8"?>
2  <LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
3    <apiVersion>59.0</apiVersion>
4    <isExposed>true</isExposed>
5    <masterLabel>AppointmentCard</masterLabel>
6    <targets>
7      <target>lightning_RecordPage</target>
8      <target>lightning_AppPage</target>
9    </targets>
10   </LightningComponentBundle>
11
```

- **PrescriptionForm** → Capture medicines and dosage.

```
⌚ prescriptionForm.html U 
nabha > force-app > main > default > lwc > prescriptionForm > ⌚ prescriptionForm.html > ...
1   <template>
2     <lightning-card title="Create Prescription">
3       <div class="slds-p-around_medium">
4         <lightning-input name="medicine" label="Medicine Id" onchange={handleChange}></lightning-input>
5         <lightning-input name="quantity" label="Quantity" type="number" onchange={handleChange}></lightning-input>
6         <lightning-button variant="brand" label="Save" onclick={saveHandler}></lightning-button>
7       </div>
8     </lightning-card>
9   </template>
10 
```

```
JS prescriptionForm.js U X 
nabha > force-app > main > default > lwc > prescriptionForm > JS prescriptionForm.js > ...
4  export default class PrescriptionForm extends LightningElement {
5    ...
15
16    saveHandler(){
17      const pres = {
18        subjectType: 'Prescription__c',
19        Appointment__c: this.appointmentId,
20        Medicine__c: this.medicineId,
21        Quantity__c: this.quantity
22      };
23      savePrescription({ p: pres })
24        .then(id => {
25          this.dispatchEvent(new CustomEvent('saved', { detail: { id } }));
26          // optional: show toast (requires import)
27        })
28        .catch(err => {
29          // error handling
30          console.error(err);
31        });
32    }
33  }
34 
```

```
RSS prescriptionForm.js-meta.xml U X 
nabha > force-app > main > default > lwc > prescriptionForm > RSS prescriptionForm.js-meta.xml
1  <?xml version="1.0" encoding="UTF-8"?>
2  <LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
3    <apiVersion>64.0</apiVersion>
4    <isExposed>true</isExposed>
5    <targets>
6      <target>lightning_RecordPage</target>
7      <target>lightning_AppPage</target>
8    </targets>
9  </LightningComponentBundle>
10 
```

- PurchaseOrderTracker → Show Purchase Orders with Pharma Suppliers.

```
purchaseOrderTracker.html U X
nabha > force-app > main > default > lwc > purchaseOrderTracker > purchaseOrderTracker.html > ...
1  <template>
2    <lightning-card title="Purchase Order Tracker" icon-name="standard:orders">
3      <div class="slds-p-horizontal_medium">
4        <template if:true={orders}>
5          <template for:each={orders} for:item="order">
6            |   <p key={order.Id}><strong>{order.Name}</strong> - {order.status__c}</p>
7            |   </template>
8        </template>
9        <template if:false={orders}>
10         |   <p>No purchase orders found.</p>
11       </template>
12     </div>
13   </lightning-card>
14 </template>
```

```
purchaseOrderTracker.js U X
nabha > force-app > main > default > lwc > purchaseOrderTracker > purchaseOrderTracker.js > PurchaseOrderTracker > w
1  import { LightningElement, wire } from 'lwc';
2  import getPurchaseOrders from '@salesforce/apex/PurchaseOrderController.getPurchaseOrders';
3
4  export default class PurchaseOrderTracker extends LightningElement {
5    orders;
6
7    @wire(getPurchaseOrders)
8    wiredOrders({ error, data }) {
9      if (data) {
10        this.orders = data;
11      } else {
12        this.orders = null;
13      }
14    }
15 }
```

```
purchaseOrderTracker.js-meta.xml U X
nabha > force-app > main > default > lwc > purchaseOrderTracker > purchaseOrderTracker.js-meta.xml
1  <?xml version="1.0" encoding="UTF-8"?>
2  <LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
3    <apiVersion>59.0</apiVersion>
4    <isExposed>true</isExposed>
5    <targets>
6      <target>lightning_AppPage</target>
7      <target>lightning_RecordPage</target>
8    </targets>
9  </LightningComponentBundle>
10
```

➤ Apex with LWC

Call Apex class TelemedicineService from LWC to:

- Fetch appointments for logged-in doctor.
- Save a new prescription.

```
● TelemedicineService.cls U X
nabha > force-app > main > default > classes > ● TelemedicineService.cls > ...
1  public with sharing class TelemedicineService {
2
3      @AuraEnabled(cacheable=true)
4      public static List<Appointment__c> getUpcomingAppointmentsForDoctor(Id doctorId, Integer daysAhead) {
5          Datetime nowDT = System.now();
6          Datetime endDT = System.now().addDays(daysAhead != null ? daysAhead : 7);
7          return [SELECT Id, Appointment_DateTime__c, Patient__c, Patient__r.Name, Patient__r.Email__c, Patient__r.Phone__c, Status__c
8                  FROM Appointment__c
9                  WHERE Doctor__c = :doctorId
10                 AND Appointment_DateTime__c >= :nowDT
11                 AND Appointment_DateTime__c <= :endDT
12                 ORDER BY Appointment_DateTime__c ASC
13             ];
14     }
15
16     @AuraEnabled
17     public static Id savePrescription(Prescription__c p) {
18         upsert p;
19         return p.Id;
20     }
21
22     @AuraEnabled(cacheable=true)
23     public static List<Purchase_Order__c> getRecentPurchaseOrders(Integer limitSize){
24         return [SELECT Id, Name, Medicine__c, Quantity__c, Status__c, Supplier__c FROM Purchase_Order__c ORDER BY Order_Date__c DESC LIMIT : (limits
25     }
26 }
```

```
● TelemedicineService.cls-meta.xml U X
nabha > force-app > main > default > classes > ● TelemedicineService.cls-meta.xml
1  <?xml version="1.0" encoding="UTF-8"?>
2  <ApexClass xmlns="http://soap.sforce.com/2006/04/metadata">
3      <apiVersion>61.0</apiVersion>
4      <status>Active</status>
5  </ApexClass>
6
```

➤ Events in LWC

Use events for communication:

- **Child → Parent:** PrescriptionForm sends “PrescriptionSaved” event to parent component.

Child dispatches CustomEvent('saved', { detail: { id } }) after saving. Parent component listens:

```
<c-prescription-form onSaved={handleSaved}></c-prescription-form>
```

- **Parent → Child:** Refresh AppointmentCard when appointment is rescheduled.

Parent handleSaved(event) can refresh list or call this.template.querySelector('c-appointment-card').loadAppointments().

➤ Wire Adapters

Use @wire with Salesforce data:

- @wire(getRecord, { recordId: '\$recordId', fields: [...] }) for Appointment record.
- @wire(getListUi, { objectApiName: 'Appointment__c', listViewApiName: 'All' }) to show appointments in UI.

➤ Imperative Apex Calls

In AppointmentCard LWC:

```
import getUpcomingAppointments from '@salesforce/apex/TelemedicineService.getUpco
...
async loadAppointments() {
    this.appts = await getUpcomingAppointments();
}
```

➤ Navigation Service

Use NavigationMixin in LWCs to redirect:

- Navigate from AppointmentCard → Appointment Record Page.
- Navigate from Patient → Related Prescriptions.

Deliverables in Phase 6

- Custom Record Pages for Appointment, Prescription, Patient.
- Custom Tabs and Home Pages.
- Utility Bar for Doctor App.
- LWC components (AppointmentCard, PrescriptionForm, PurchaseOrderTracker).
- Apex integrated with LWC (Wire + Imperative).
- Navigation between components.

SALESFORCE PROJECT IMPLEMENTATION PHASES

Phase 7: Integration & External Access

This phase focuses on connecting Salesforce with external systems and managing data exchange securely. The following components are implemented:

➤ Named Credentials

A secure way to store authentication settings (endpoint URL, username, password, OAuth tokens).

Use Case: Instead of hardcoding API credentials in Apex, use Named Credentials.

SETUP > NAMED CREDENTIALS
PharmaAPI

Label	PharmaAPI	Name	PharmaAPI
URL	https://api.pharmasupplier.example.com		
Enabled for Callouts	<input checked="" type="checkbox"/>		
Authentication			
External Credential	PharmaExternalCred		
Client Certificate			
Callout Options			
Generate Authorization Header	<input checked="" type="checkbox"/>		
Allow Formulas in HTTP Header	<input checked="" type="checkbox"/>		
Allow Formulas in HTTP Body	<input type="checkbox"/>		
Outbound Network Connection			
Managed Package Access			
Created By Namespace			
Allowed Namespaces for Callouts			

SETUP
Auth. Providers

Auth. Provider

Auth. Provider Detail

Auth. Provider ID	OSOgl000000shuP	Edit Delete Clone
Provider Type	Salesforce	
Name	PharmaAuth	
URL Suffix	PharmaAuth	
Consumer Key	PharmaAuth	
Consumer Secret	Click to reveal	
Authorize Endpoint URL	https://login.salesforce.com/services/oauth2/authorize	
Token Endpoint URL	https://login.salesforce.com/services/oauth2/token	
Use Proof Key for Code Exchange (PKCE) Extension	<input checked="" type="checkbox"/> i	
Default Scopes	read write	
Include Consumer Secret in SOAP API Responses	<input checked="" type="checkbox"/> i	
Custom Error URL		
Custom Logout URL		
Registration Handler Type	None	
Portal		
Icon URL		
Use Salesforce MFA for this SSO Provider	<input type="checkbox"/> i	

Salesforce Configuration

Test-Only Initialization URL: https://orgfarm-f66b8c094f-dev-ed-develop.my.salesforce.com/services/auth/test/PharmaAuth

PharmaExternalCred

Label: PharmaExternalCred

Name: PharmaExternalCred

Authentication Protocol: OAuth 2.0

Authentication Flow Type: Browser Flow

Identity Provider: PharmaAuth

Callout Options

Managed Package Access

Created By Namespace

➤ External Services

Declarative way to call REST APIs without writing Apex, using an OpenAPI (Swagger) schema.

Use Case: Connect to hospital's Appointment API or demo Pet Store API.

```
! pharma.openapi.yaml ✘
nabha > force-app > main > default > ExternalServices > ! pharma.openapi.yaml
  1  swagger: "2.0"
  2  info:
  3    version: "1.0.0"
  4    title: "Pharma Supplier - Appointment API (Demo)"
  5    host: api.pharmasupplier.example.com
  6    basePath: /v1
  7    schemes:
  8      - https
  9    paths:
10      /appointments:
11        get:
12          summary: "Get appointments for a date"
13          parameters:
14            - name: date
15              in: query
16              description: "Date in YYYY-MM-DD"
17              required: true
18          type: string
19          responses:
20            200:
21              description: "List of appointments"
22              schema:
23                type: array
24                items:
25                  $ref: "#/definitions/Appointment"
26          post:
27            summary: "Create appointment"
```

PharmaInventoryES

Service name	Creation source	Description	Type	File Name	Named credentials	Created by
PharmaInventoryES	From API specification		OpenApi	pharma.openapi.yaml	PharmaAPI	Anjali Raikwar
Created date		Last modified by	Last modified date			
September 25, 2025 at 07:47 PM		Anjali Raikwar	September 25, 2025 at 07:47 PM			

Operations 2 Items · Sorted by Operation Name

Operation Name ↑	Description	Input parameters	Output parameters
getAppointments	Get appointments for a date	date	default, responseCode, 200
postAppointments	Create appointment	body	default, 201, responseCode

```
► TM_AppointmentAPI.cls U X
nabha > force-app > main > default > classes > ► TM_AppointmentAPI.cls > ...
1   @RestResource(urlMapping='/telemedicine/appointments/*')
2   global with sharing class TM_AppointmentAPI {
3
4       @HttpGet
5       global static Appointment__c getAppointment() {
6           RestRequest req = RestContext.request;
7           String id = req.requestURI.substring(req.requestURI.lastIndexOf('/')+1);
8
9           return [
10              SELECT Id, Appointment_DateTime__c, Patient__c
11              FROM Appointment__c
12              WHERE Id = :id
13              LIMIT 1
14          ];
15      }
16  }
17 }
```

➤ Web Services (REST/SOAP)

Salesforce can expose its logic/data as web services or consume external ones.

Exposing:

REST → Use @RestResource annotation in Apex.

SOAP → Use global class with webService keyword.

```
► TM_AppointmentAPI.cls U X
nabha > force-app > main > default > classes > ► TM_AppointmentAPI.cls > ...
1   @RestResource(urlMapping='/telemedicine/appointments/*')
2   global with sharing class TM_AppointmentAPI {
3
4       @HttpGet
5       global static Appointment__c getAppointment() {
6           RestRequest req = RestContext.request;
7           String id = req.requestURI.substring(req.requestURI.lastIndexOf('/')+1);
8
9           return [
10              SELECT Id, Appointment_DateTime__c, Patient__c
11              FROM Appointment__c
12              WHERE Id = :id
13              LIMIT 1
14          ];
15      }
16  }
17 }
```

➤ Callouts

Outbound HTTP requests from Salesforce to external services.

```
➡ PharmaIntegration.cls U X
nabha > force-app > main > default > classes > ➡ PharmaIntegration.cls > ...
1  public with sharing class PharmaIntegration {
2
3      // Sync callout (example GET request)
4      public static HttpResponse getInventory(String sku) {
5          HttpRequest req = new HttpRequest();
6
7          // Endpoint via Named Credential
8          req.setEndpoint('callout:PharmaAPI/v1/inventory?sku=' +
9              EncodingUtil.urlEncode(sku, 'UTF-8'));
10         req.setMethod('GET');
11         req.setTimeout(120000); // 2 min timeout
12
13         Http http = new Http();
14         HttpResponse res;
15
16         try {
17             res = http.send(req);
18             System.debug('Response: ' + res.getBody());
19         } catch (Exception e) {
20             System.debug('Callout error: ' + e.getMessage());
21             throw e;
22         }
23         return res;
24     }
25 }
26
```

```
➡ PharmaIntegrationTest.cls U ●
nabha > force-app > main > default > classes > ➡ PharmaIntegrationTest.cls > ➡ PharmaIntegrationTest > ➡ MockHttpCalloutMock.cls > ...
1  @IsTest
2  private class PharmaIntegrationTest {
3
4      private class MockHttpResponseGenerator implements HttpCalloutMock {
5          public HttpResponse respond(HTTPRequest req) {
6              HttpResponse res = new HttpResponse();
7              res.setHeader('Content-Type', 'application/json');
8              res.setBody('{"sku": "ABC123", "stock": 50}');
9              res.setStatusCode(200);
10             return res;
11         }
12     }
13
14     @IsTest
15     static void testGetInventory() {
16         Test.setMock(HttpCalloutMock.class, new MockHttpResponseGenerator());
17
18         HttpResponse res = PharmaIntegration.getInventory('ABC123');
19         System.assertEquals(200, res.getStatusCode());
20         System.assert(res.getBody().contains('"stock": 50'));
21     }
22 }
23
```

➤ Platform Events

Event-driven architecture in Salesforce (publish-subscribe model).

Use Case: Notify external systems when an appointment is booked or cancelled.

The screenshot shows the Salesforce setup interface for Platform Events. At the top, there's a table titled "Custom Fields & Relationships" with three rows:

Action	Field Label	API Name	Data Type	Indexed	Controlling Field	Modified By
Edit Del	AppointmentId	AppointmentId_c	Text(18)			Anjali Raikwar, 9/25/2025, 8:03 AM
Edit Del	Payload	Payload_c	Long Text Area(32768)			Anjali Raikwar, 9/25/2025, 8:04 AM

Below this is the "Platform Event Definition Detail" page for "Appointment_Event". It shows the following details:

Singular Label	Plural Label	Object Name	API Name	Description	Deployment Status
Appointment_Event	Appointment_Events	Appointment_Event	Appointment_Event_e		Deployed

Event Type: High Volume. Publish Behavior: Publish Immediately. Created By: Anjali Raikwar. Modified By: Anjali Raikwar.

Under "Standard Fields", there's a table with four rows:

Action	Field Label	Field Name	Data Type	Controlling Field	Indexed
	Created By	CreatedBy	Lookup(User)		
	Created Date	CreatedDate	Date/Time		
	Event UUID	EventUuid	Text(36)		

The screenshot shows the code editor for the Apex class `AppointmentEventPublisher.cls`. The code is as follows:

```
1  public with sharing class AppointmentEventPublisher {
2      public static void publishAppointmentEvent(Id apptId, String status) {
3          try {
4              Appointment_Event__e evt = new Appointment_Event__e(
5                  AppointmentId__c = apptId,
6                  Status__c = status,
7                  Payload__c = 'Appointment status changed to ' + status
8              );
9              Database.SaveResult sr = EventBus.publish(evt);
10             System.debug('Event published? ' + sr.isSuccess());
11         } catch (Exception e) {
12             System.debug('Error publishing event: ' + e.getMessage());
13         }
14     }
15 }
16 }
```

```

≡ AppointmentEventTrigger.trigger U •
nabha > force-app > main > default > triggers > ≡ AppointmentEventTrigger.trigger > AppointmentEventTrigger
1   trigger AppointmentEventTrigger on Appointment_Event__e (after insert) {
2     for (Appointment_Event__e evt : Trigger.new) {
3       System.debug('Received Appointment Event: ' + evt.AppointmentId__c +
4           ' Status: ' + evt.Status__c);
5       // Example: Log or update something
6       // Could update a related object or push to another queue
7     }
8   }
9

```

➤ Change Data Capture

Automatically streams changes (Create, Update, Delete, Undelete) of Salesforce records.

Available Entities	Selected Entities
Account (Account)	Doctor (Doctor__c)
Account Clean Info (AccountCleanInfo)	Appointment (Appointment__c)
Account Contact Data (AccountContactData)	Patient (Patient__c)

➤ Salesforce Connect

Lets Salesforce access external data in real time without storing it.

External Data Source	Pharma Supplier
Name	Pharma_Supplier
Type	Salesforce Connect: OData 4.0
Parameters	
URL	https://services.odata.org/V4/Northwind/Northwind.svc/
Connection Timeout (Seconds)	120
Writable External Objects	<input type="checkbox"/>
High Data Volume	<input type="checkbox"/>
Server Driven Pagination	<input type="checkbox"/>
Request Row Counts	<input checked="" type="checkbox"/>
Compress Requests	<input type="checkbox"/>
Enable Search	<input checked="" type="checkbox"/>
Use Free-Text Search Expressions	<input type="checkbox"/>

➤ API Limits

Salesforce enforces API limits per 24 hours (depends on edition & licenses).

API Usage Notification

Help for this Page 

API Usage Notification Detail	
Notification Recipient	Priva Singh Singh
Threshold	65%
Notification Interval (Hours)	24
Created By	Anjali Raikwar, 9/25/2025, 8:18 AM
Modified By	Anjali Raikwar, 9/25/2025, 8:18 AM
	Edit Delete Clone

➤ OAuth & Authentication

Secure authentication mechanism for APIs.

Manage External Client Apps	
 Telemedicine Integration	
Contact Email	App Authorization
anjali.raikwar.cs22@gkits.net	All users can self-authorize
Type	App Status
Local	Enabled
Help for this Page  Disable 	

➤ Remote Site Settings

Whitelist external domains before Salesforce can make callouts.

SETUP																									
 Remote Site Settings																									
Remote Site Details																									
Remote Site Detail																									
<table border="1"><thead><tr><th></th><th></th><th>Edit Delete Clone</th></tr></thead><tbody><tr><td>Remote Site Name</td><td>PharmaAPI_Remote</td><td>Modified By Anjali Raikwar, 9/25/2025, 8:24 AM</td></tr><tr><td>Remote Site URL</td><td>https://api.pharmasupplier.example.com</td><td></td></tr><tr><td>Disable Protocol Security</td><td><input type="checkbox"/></td><td></td></tr><tr><td>Description</td><td>Pharma supplier</td><td></td></tr><tr><td>Active</td><td><input checked="" type="checkbox"/></td><td></td></tr><tr><td>Created By</td><td>Anjali Raikwar, 9/25/2025, 8:24 AM</td><td></td></tr><tr><td></td><td>Edit Delete Clone</td><td></td></tr></tbody></table>				Edit Delete Clone	Remote Site Name	PharmaAPI_Remote	Modified By Anjali Raikwar , 9/25/2025, 8:24 AM	Remote Site URL	https://api.pharmasupplier.example.com		Disable Protocol Security	<input type="checkbox"/>		Description	Pharma supplier		Active	<input checked="" type="checkbox"/>		Created By	Anjali Raikwar, 9/25/2025, 8:24 AM			Edit Delete Clone	
		Edit Delete Clone																							
Remote Site Name	PharmaAPI_Remote	Modified By Anjali Raikwar , 9/25/2025, 8:24 AM																							
Remote Site URL	https://api.pharmasupplier.example.com																								
Disable Protocol Security	<input type="checkbox"/>																								
Description	Pharma supplier																								
Active	<input checked="" type="checkbox"/>																								
Created By	Anjali Raikwar, 9/25/2025, 8:24 AM																								
	Edit Delete Clone																								

Deliverables in Phase 7

- Named Credentials → Store endpoint + auth.
- External Services → Declarative API calls.
- Web Services → Expose/consume SOAP & REST.
- Callouts → Programmatic API calls.
- Platform Events & CDC → Event-driven data sync.
- Salesforce Connect → Real-time external data access.
- API Limits → Monitor & optimize API usage.
- OAuth & Auth → Secure integrations.
- Remote Site Settings → Whitelist domains.

SALESFORCE PROJECT IMPLEMENTATION PHASES

Phase 8: Data Management & Deployment

Managing Salesforce data and deploying configurations across environments is crucial for maintaining data integrity and smooth release cycles.

➤ Data Import Wizard

The Data Import Wizard is a simple, browser-based tool that allows admins to import data into Salesforce without writing code. It is best suited for smaller datasets (up to 50,000 records).

Key Features

- Supports Standard Objects (Leads, Accounts, Contacts, Solutions, Campaign Members) and Custom Objects.
- Provides an intuitive step-by-step interface.
- Allows mapping of CSV fields to Salesforce fields.

Data Import Wizard

Help for this page 

Recent Import Jobs

Status	Object	Records Created	Records Updated	Records Failed	Start Date	Processing Time (ms)
Closed	Patient	0	0	2	09-25-2025 06:49	73

Bulk API Monitoring

Batches													
View Request	View Result	Batch ID	Start Time	End Time	Total Processing Time (ms)	API Active Processing Time (ms)	Apex Processing Time (ms)	Records Processed	Records Failed	Retry Count	State Message	Status	
View Request	View Result	751gL00000Bg1KZ	9/25/2025, 11:49 AM	9/25/2025, 11:49 AM	73	10	0	2	2	0	Completed		

FirstName LastName Email	Phone	DOB_c	
Anjali Raikwar anjali@test.com	9876543210	1999-05-02	
Rohan Sharma rohan@test.com	9876500011	1998-07-10	

➤ Data Loader

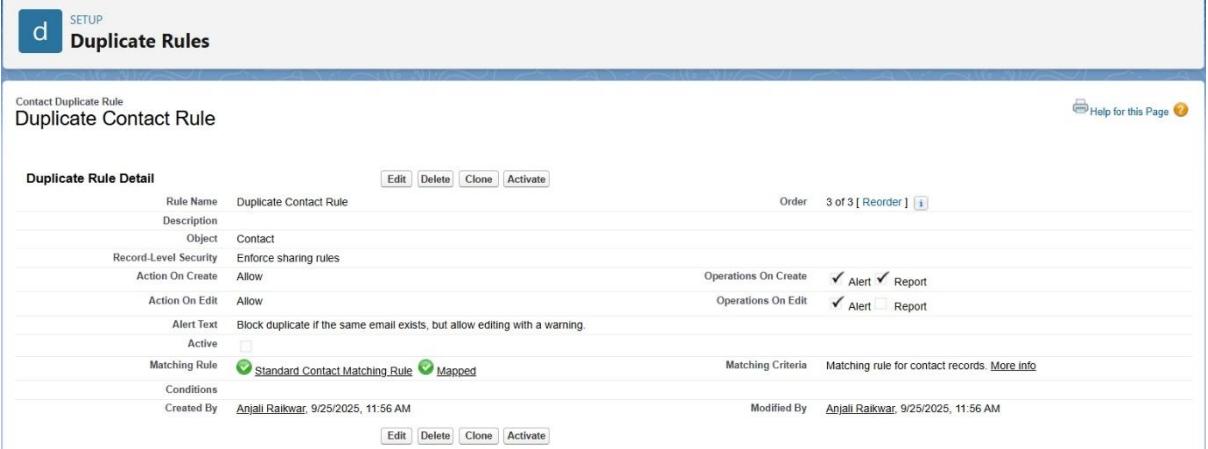
The Salesforce Data Loader is a client application used for bulk data operations (up to 5 million records). It provides more control and is ideal for larger datasets.

Key Features

- Supports Insert, Update, Upsert, Delete, Export.
- Works with CSV files.
- Can be scheduled via CLI for automation.

➤ Duplicate Rules

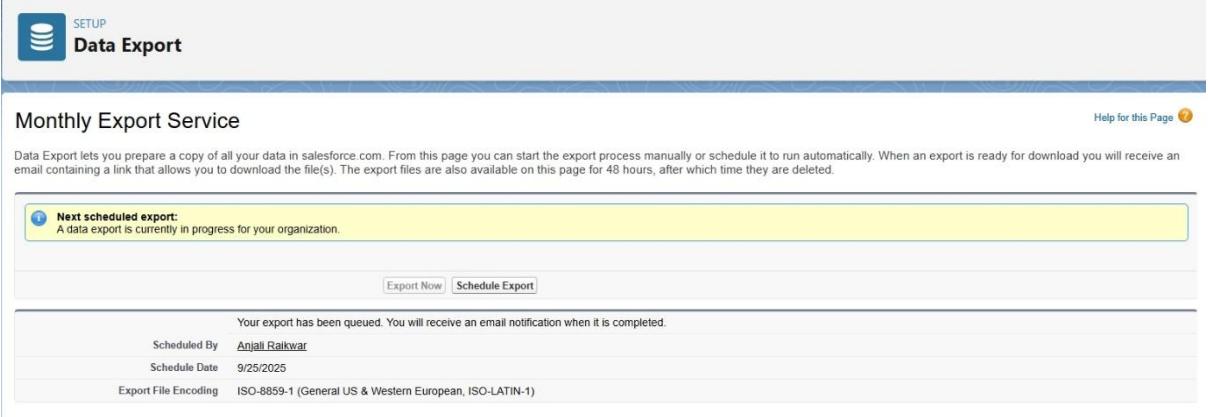
Duplicate Rules prevent creation of duplicate records, thereby ensuring data quality. These rules can block duplicates or alert users when similar records already exist.



The screenshot shows the 'Duplicate Rules' page in the Salesforce Setup. The title bar says 'd SETUP Duplicate Rules'. Below it, a sub-header says 'Contact Duplicate Rule' and 'Duplicate Contact Rule'. On the right, there's a 'Help for this Page' link. The main area is titled 'Duplicate Rule Detail' with a sub-section 'Rule Name: Duplicate Contact Rule'. It includes fields for 'Description' (Contact), 'Record-Level Security' (Enforce sharing rules), 'Action On Create' (Allow), 'Action On Edit' (Allow), 'Alert Text' (Block duplicate if the same email exists, but allow editing with a warning), and 'Active' (unchecked). Under 'Matching Rule', it shows 'Standard Contact Matching Rule' (Mapped). The 'Matching Criteria' section indicates 'Matching rule for contact records' with a 'More info' link. At the bottom, there are 'Edit', 'Delete', 'Clone', and 'Activate' buttons. The top right shows 'Order: 3 of 3 [Reorder]'.

➤ Data Export & Backup

Salesforce allows exporting full datasets for backup or compliance. Exports can be scheduled weekly or monthly.



The screenshot shows the 'Data Export' page in the Salesforce Setup. The title bar says 'SETUP Data Export'. Below it, a sub-header says 'Monthly Export Service'. On the right, there's a 'Help for this Page' link. The main area shows a message: 'Next scheduled export: A data export is currently in progress for your organization.' Below this is a button for 'Export Now' and another for 'Schedule Export'. A note below says 'Your export has been queued. You will receive an email notification when it is completed.' It lists 'Scheduled By: Anjali Raikwar', 'Schedule Date: 9/25/2025', and 'Export File Encoding: ISO-8859-1 (General US & Western European, ISO-LATIN-1)'.

➤ Change Sets

Change Sets are Salesforce's point-and-click tool for moving metadata (custom fields, objects, Apex, Flows, etc.) between environments like Sandbox → Production.

➤ Unmanaged vs Managed Packages

Packages are used to group components for distribution.

- Unmanaged Package:
 - o Components are editable after installation.
 - o Ideal for one-time deployments or sharing code.

- Managed Package:
 - o Code is locked after installation.
 - o Supports versioning and upgrade paths.
 - o Required for AppExchange distribution.

Change Sets: Used to move metadata between related Salesforce orgs (Sandbox → Production).

Not available in Developer Edition (since it has no Sandbox). For this project, we demonstrate deployment using **VS Code (SFDX)** instead.

➤ **ANT Migration Tool**

The ANT Migration Tool is a Java/Apache ANT-based tool that enables scripted deployments between Salesforce orgs. It is powerful for CI/CD automation.

- **build.properties**

```
sf.username=your_username@example.com  
sf.password=your_password+securityToken  
sf.serverurl=https://login.salesforce.com
```

- **package.xml**

```
<?xml version="1.0" encoding="UTF-8"?>  
<Package xmlns="http://soap.sforce.com/2006/04/metadata">  
    <types>  
        <members>*        <name>ApexClass</name>  
    </types>  
    <types>  
        <members>*        <name>CustomObject</name>  
    </types>  
    <version>58.0</version>  
</Package>
```

- build.xml

```
<project name="Sample" default="deployCode">
    <taskdef resource="com/salesforce/antlib.xml"
        classpath="ant-salesforce.jar"/>

    <target name="retrieveCode">
        <sf:retrieve username="${sf.username}" password="${sf.password}" serverurl="${sf.serverurl}"
    </target>

    <target name="deployCode">
        <sf:deploy username="${sf.username}" password="${sf.password}" serverurl="${sf.serverurl}"
    </target>
</project>
```

- retrieveCode → pulls metadata from org.
- deployCode → pushes metadata to org.

➤ VS Code & SFDX (Salesforce DX)

Salesforce DX (SFDX) with Visual Studio Code is the modern way to develop and deploy Salesforce applications. It enables source-driven development, version control, scratch orgs, and CI/CD.

SALESFORCE PROJECT IMPLEMENTATION PHASES

Phase 9: Reporting, Dashboards & Security Review

This phase ensures that our **Telemedicine CRM** delivers actionable insights through reports and dashboards, while also maintaining enterprise-grade security through Salesforce's built-in controls.

➤ Reports (Tabular, Summary, Matrix, Joined)

Reports are used to analyze data such as patient appointments, prescriptions, or doctor workload.

Types

- **Tabular** → Simple list of records (e.g., all appointments today).

The screenshot shows the Salesforce Report Builder interface. At the top, there is a header with buttons for 'Preview Layout', 'Edit Layout', 'Clone', 'Delete', and 'Close'. Below the header, the report title is 'Today's Appointments' and a description states: 'Below is the information for this custom report type. You can click the buttons on this to preview or update information for the custom report type.' The report structure is divided into sections: 'Details' (containing fields like Display Label, API Name, Description, Created By, etc.), 'Fields' (listing Source Object as Appointment and Included Fields as 21), and 'Object Relationships' (showing a diagram where 'Appointment (A)' is connected to a list of records). Below this, the 'REPORT' tab is selected, showing the report configuration with 'New Today's Appointments Report' and 'Today's Appointments' as the preview. The 'Filters' section is expanded, showing 'Last Modified Date' set to 'Current FQ (Jul 1, 2025 - Sep 30, 2025)' and 'Appointment Date equals Sep 26, 2025'.

The screenshot shows the report configuration interface. In the 'Groups' section, 'Appointment ID', 'Appointment Date', and 'Appointment DateTime' are listed under 'GROUP ROWS'. In the 'Columns' section, 'Appointment Number', 'Status', 'Patient__c: Patient Name', and 'Doctor: Full Name' are listed.

- Summary → Grouped by a field (e.g., appointments grouped by doctor).

The screenshot shows the report preview screen for 'Appointments by Doctor - Week Report'. The filters applied are 'Doctor: Full Name', 'Status', 'Appointment DateTime', and 'Appointment Number'. The preview message indicates no records returned in preview. The filters panel shows 'Add filter...', 'Show Me All appointment', 'Appointment Date Current FQ (Jul 1, 2025 - Sep 30, 2025)', and 'Appointment DateTime equals THIS WEEK'.

Outline Filters (2)

Groups

- GROUP ROWS
- Doctor: Full Name
- Status
- Appointment DateTime

GROUP COLUMNS

Columns

- Add column...
- Appointment Number

appointments by Doctor - Week

Below is the information for this custom report type. You can click the buttons on this to preview or update information for the custom report type.

Preview Layout Edit Layout Clone Delete Close

Details

Display Label: appointments by Doctor - Week
API Name: appointments_by_Doctor_Week
Description: This is the weekly appointment report list for doctor

Created By: Anjali Raikwar, 26/09/25, 9:44 pm
Store in Category: other
Deployment: Deployed
Modified By: Anjali Raikwar, 26/09/25, 9:44 pm

Fields

Source Object	Included Fields
Appointment	21

Object Relationships

```

graph TD
    A([Appointment (A)]) --> B[A]
  
```

- **Matrix** → Grouped both rows & columns (e.g., doctors vs. appointment status).

REPORT ▾

New Matrix Doctor vs Status Report / Matrix Doctor vs Status

Previewing a limited number of records. Run the report to see everything.

Update Preview Automatically

Doctor: Full Name ▾ Status → ▾ Total

No records returned in preview. Try running the report or editing report filters.

- Set the Appointment Date filter to All Time.
- Edit other filters in the filter panel.

Outline Filters (2)

Groups

- GROUP ROWS
- Add group...
- Doctor: Full Name

GROUP COLUMNS

- Add group...
- Status

Columns

- Add column...
- Appointment Number

Details (0 Rows) Click an intersection in the table above to filter details.

☰ Outline ⚙ Filters 2

Filters

Add filter... 🔍

Show Me
All appointment

Appointment Date
Current FQ (Jul 1, 2025 - Sep 30, 2025)

Appointment DateTime
equals THIS MONTH trash

Matrix Doctor vs Status

Preview Layout Edit Layout Clone Delete Close

Below is the information for this custom report type. You can click the buttons on this to preview or update information for the custom report type

Details

Display Label	Matrix Doctor vs Status
API Name	Matrix_Doctor_vs_Status
Description	This is the matrix for doctor and their status
Created By	Anjali Raikwar, 26/09/25, 9:50 pm
Store in Category	other
Deployment ...	Deployed
Modified By	Anjali Raikwar, 26/09/25, 9:50 pm

Fields

Source Object	Included Fields
Appointment	21

Object Relationships

Appointment (A) edit

- Joined → Combine multiple report types (patient records + prescription data).

Joined Appointments Prescriptions

Preview Layout Edit Layout Clone Delete Close

Below is the information for this custom report type. You can click the buttons on this to preview or update information for the custom report type

Details

Display Label	Joined Appointments Prescriptions
API Name	Joined_Appointments_Prescriptions
Description	This display the appointment prescription
Created By	Anjali Raikwar, 26/09/25, 9:59 pm
Store in Category	other
Deployment ...	Deployed
Modified By	Anjali Raikwar, 26/09/25, 10:07 pm

Fields

Source Object	Included Fields
Appointment	20
Prescriptions	5

Object Relationships

Appointment (A) edit

... with at least one related record from Prescriptions (B)

Best Use Case → Monitor doctor performance, patient volume, and prescription trends.

➤ Report Types

Define which objects and fields are available in reports.

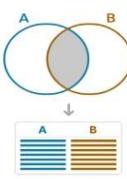
Healthcare Reports

Below is the information for this custom report type. You can click the buttons on this to preview or update information for the custom report type.

Preview Layout Edit Layout Clone Delete Close

Details	
Display Label	Healthcare Reports
API Name	Healthcare_Reports
Description	This is an Healthcare Report
Created By	Anjali Raikwar, 27/09/25, 12:39 am
Store in Category	other
Deployment Status	Deployed
Modified By	Anjali Raikwar, 27/09/25, 12:39 am

Object Relationships	
Appointment (A)	with at least one related record from Prescriptions (B)



Source Object	Included Fields
Appointment	21
Prescriptions	18

Best Use Case → Custom objects reporting (Appointments + Prescriptions).

➤ Dashboards

Visualize multiple reports in one view (graphs, charts, gauges).

Doctor Dashboard

Lead conversion rate

We can't draw this chart because there is no data.

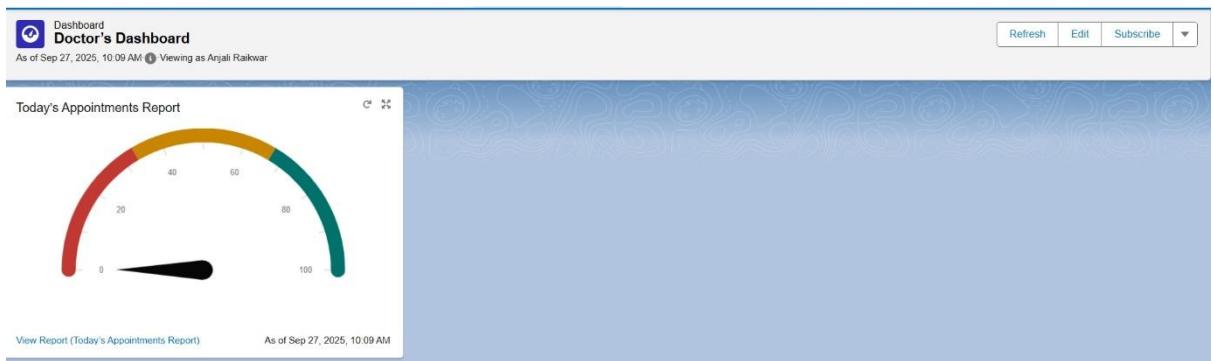
View Report As of Today at 10:06 AM ⓘ

All

All 0 items, sorted by Event Name >

Appointment Summary

Doctor:
Patient:
Date:
Status:



Best Use Case → Doctor's dashboard:

- Todays Appointments (list)
- Appointment Status (pie chart)
- Top 5 Patients (bar chart)

➤ **Dynamic Dashboards**

Dashboards that run as the logged-in user → each doctor sees their own data without creating multiple dashboards.

Best Use Case → Each doctor automatically sees only their patients/appointments.

➤ **Sharing Settings**

Controls record-level access (private vs public data).

SETUP			
Sharing Settings			
Work Type Group	Public Read/Write	Private	✓
Appointment	Controlled by Parent	Controlled by Parent	✓
Archived Prescription	Public Read/Write	Private	✓
Doctor	Public Read/Write	Private	✓
Error Log	Public Read/Write	Private	✓
Medicine	Public Read/Write	Private	✓
Medicine Inventory	Public Read/Write	Private	✓
Mentor	Public Read/Write	Private	✓
Notification Log	Public Read/Write	Private	✓
Patient	Public Read/Write	Private	✓
Prescription	Controlled by Parent	Controlled by Parent	✓

The screenshot shows the Salesforce Setup Roles page. The main title is "SETUP Roles". Below it is a tree view of roles:

- VP, International Sales** (Edit | Del | Assign)
 - Add Role
- VP, Marketing** (Edit | Del | Assign)
 - Add Role
- Marketing Team** (Edit | Del | Assign)
 - Add Role
- VP, North American Sales** (Edit | Del | Assign)
 - Add Role
 - Director, Channel Sales** (Edit | Del | Assign)
 - Add Role
 - Channel Sales Team** (Edit | Del | Assign)
 - Add Role
 - Director, Direct Sales** (Edit | Del | Assign)
 - Add Role
 - Eastern Sales Team** (Edit | Del | Assign)
 - Add Role
 - Western Sales Team** (Edit | Del | Assign)
 - Add Role
- Hospital Admin** (Edit | Del | Assign)
 - Add Role
 - Admin User** (Edit | Del | Assign)
 - Add Role
 - Doctor** (Edit | Del | Assign)
 - Add Role
 - Patient User** (Edit | Del | Assign)
 - Add Role
 - Pharmacy Staff** (Edit | Del | Assign)
 - Add Role

Best Use Case → Protect patient confidentiality.

➤ Field Level Security

Hide sensitive fields (e.g., Patient Medical History) from unauthorized users.

The screenshot shows the Salesforce Setup Permission Sets page for "Doctor Access Aadhaar". The main title is "SETUP Permission Sets". Below it is a "Permission Set Overview" section with tabs for "Object Settings" and "Patients".

Patients

Tab Settings

Available	Visible
<input type="checkbox"/>	<input checked="" type="checkbox"/>

Object Permissions

Permission Name	Enabled
Read	<input type="checkbox"/>
Create	<input type="checkbox"/>
Edit	<input type="checkbox"/>
Delete	<input type="checkbox"/>
View All Records	<input type="checkbox"/>
Modify All Records	<input type="checkbox"/>
View All Fields	<input type="checkbox"/>

Field Permissions

Field Name	Field API Name	Read Access	Edit Access
Aadhaar Number	Aadhaar_Number__c	<input type="checkbox"/>	<input checked="" type="checkbox"/>

The screenshot shows two related pages from the Salesforce setup interface.

Layout Properties: This page allows you to define the layout of a record page. It includes sections for "Fields" (with a "Quick Find" search bar), "Standard Buttons" (Edit, Delete, Clone, Change Owner, Change Record Type, Printable View, Sharing, Sharing Hierarchy, Edit Labels), and "Custom Buttons". A note at the bottom states: "Actions in this section are predefined by Salesforce. You can override the predefined actions to set a customized list of actions on Lightning Experience and mobile app pages that use this layout. If you customize the actions in the Quick Actions or the Salesforce Classic Publisher section, and have saved the layout, then this section inherits that set of actions by default when you click to override."

Field-Level Security: This page is titled "Set Field-Level Security" for the "Aadhaar Number" field. It shows the current field details: "Field Label: Aadhaar Number" and "Data Type: Text(12) (Unique Case Insensitive)". Below this is a table titled "Field-Level Security for Profile" showing security settings for various profiles. The columns are "Profile" (list of profiles like Analytics Cloud Integration User, Analytics Cloud Security User, Anypoint Integration, etc.), "Visible" (checkboxes), and "Read-Only" (checkboxes). Most profiles have the "Visible" checkbox checked, while "Authenticated Website" and "Custom: Marketing Profile" have it unchecked. All profiles have the "Read-Only" checkbox unchecked.

Best Use Case → Admin sees everything, Receptionist only sees Appointment Time & Patient Name.

➤ Session Settings

Controls login security (timeouts, session restrictions).

Session Settings

Set the session security and session expiration timeout for your organization.

Session Timeout

Timeout Value	30 minutes
<input type="checkbox"/> Disable session timeout warning popup	
<input checked="" type="checkbox"/> Force logout on session timeout	

Session Settings

<input checked="" type="checkbox"/> Lock sessions to the IP address from which they originated
<input checked="" type="checkbox"/> Lock sessions to the domain in which they were first used
<input type="checkbox"/> Terminate all of a user's sessions when an admin resets that user's password
<input checked="" type="checkbox"/> Force relogin after Login-As-User
<input type="checkbox"/> Require HttpOnly attribute
<input type="checkbox"/> Use POST requests for cross-domain sessions
<input type="checkbox"/> Enforce login IP ranges on every request
<input type="checkbox"/> When embedding a Lightning application in a third-party site, use a session token instead of a session cookie.

Extended use of IE11 with Lightning Experience

EXTENDED USE OF IE11 WITH LIGHTNING EXPERIENCE HAS NOW ENDED
AS OF DECEMBER 31, THE EXTENDED PERIOD HAS ENDED, AND USE OF INTERNET EXPLORER 11 (IE 11) WITH LIGHTNING EXPERIENCE IS NO LONGER SUPPORTED. ISSUES WITH PERFORMANCE OR FUNCTIONALITY THAT AFFECT ONLY IE 11 WILL NOT BE FIXED. PLEASE SWITCH TO A SUPPORTED BROWSER.

Best Use Case → Auto log out idle users to prevent unauthorized access.

➤ Login IP Ranges

Restrict user logins to trusted networks (e.g., hospital Wi-Fi).

Network Access

The list below contains IP address ranges from sources that your organization trusts. Users logging in to salesforce.com with a browser from trusted networks are allowed to access salesforce.com without having to activate their computers.

Trusted IP Ranges

Action	Start IP Address	End IP Address	Description
Edit Del	103.25.64.1	103.25.64.255	

Login IP Ranges

Action	IP Start Address	IP End Address	Description
Edit Del	27.60.10.23	27.60.10.255	

Best Use Case → Prevent external logins outside hospital network.

➤ Audit Trail

Track who changed what in setup for compliance.



SETUP

View Setup Audit Trail

[Help for this Page](#)

View Setup Audit Trail

The last 20 entries for your organization are listed below. You can [download](#) your organization's setup audit trail for the last six months (Excel .csv file).

View Setup Audit Trail

Date	User	Source Namespace Prefix	Action	Section	Delegate User
9/26/2025, 11:59:02 AM PDT	anjali.raikwar.cs22269@agentforce.com		Added Login Ip Range to Doctor Profile from 27.60.10.23 to 27.60.10.255	Manage Users	
9/26/2025, 11:41:48 AM PDT	anjali.raikwar.cs22269@agentforce.com		Added Login Ip Range to Doctor Profile from 27.60.10.23 to 103.25.64.255	Manage Users	
9/26/2025, 11:39:38 AM PDT	anjali.raikwar.cs22269@agentforce.com		Added Ip AllowList from 103.25.64.1 to 103.25.64.255	Security Controls	
9/26/2025, 11:37:21 AM PDT	anjali.raikwar.cs22269@agentforce.com		Changed Session Timeout Value from 120 to 30 minutes	Session Settings	
9/26/2025, 11:37:21 AM PDT	anjali.raikwar.cs22269@agentforce.com		Change Lock Session IP from off to on	Session Settings	
9/26/2025, 11:37:21 AM PDT	anjali.raikwar.cs22269@agentforce.com		Session Security Level for Multi-Factor Authentication was set to High Assurance	Session Settings	
9/26/2025, 11:37:21 AM PDT	anjali.raikwar.cs22269@agentforce.com		Session Security Level for PharmaAuth was set to Standard	Session Settings	
9/26/2025, 11:37:21 AM PDT	anjali.raikwar.cs22269@agentforce.com		Session Security Level for Passwordless Login was set to Standard	Session Settings	
9/26/2025, 11:37:21 AM PDT	anjali.raikwar.cs22269@agentforce.com		Session Security Level for Lightning Login was set to Standard	Session Settings	
9/26/2025, 11:37:21 AM PDT	anjali.raikwar.cs22269@agentforce.com		Session Security Level for Activation was set to Standard	Session Settings	
9/26/2025, 11:37:21 AM PDT	anjali.raikwar.cs22269@agentforce.com		Session Security Level for Delegated Authentication was set to Standard	Session Settings	

Best Use Case → Security reviews, compliance with healthcare regulations (HIPAA/GDPR).