Health & Wellness Tracking Portal - Phase 1

The Health & Wellness Tracking Portal is an application designed to help patients monitor their health metrics and enable healthcare providers to track wellness trends. The system allows patients to log vital information such as blood pressure, sugar levels, and other health indicators, while doctors and health coaches can analyze these inputs through dashboards and reports.

Beyond simple record keeping, the portal identifies at-risk patients, highlights common health issues, and promotes proactive healthcare management. It integrates health tracking, analytics, and patient-doctor engagement into a single platform—an innovative solution for lifestyle and wellness management.

Requirement Gathering

- ➢ Allow patients to log vitals such as blood pressure, sugar levels, weight, and heart rate.
- ➢ Enable healthcare providers (doctors/coaches) to view patient trends over time.
- ➢ Generate dashboards that highlight at-risk patients based on abnormal readings.
- ➢ Send automated alerts/notifications when a patient logs critical values.
- ➢ Optionally, track lifestyle details like exercise, sleep, and diet for holistic wellness management.

Stakeholder Analysis

- ➢ Primary User: Patients who log their vitals and wellness information.
- ➢ Admin Role: Configures the system, manages patient data structure, and sets automation rules.
- ➢ Healthcare Providers: Doctors and wellness coaches who monitor trends and provide advice.
- ➢ Secondary Users: Family members or caregivers with shared access to patient data.

Business Process Mapping

- ➢ Log Health Data: Patients enter their daily vitals (e.g., BP, sugar levels).
- ➢ Monitor & Track: Data is stored and linked to patient profiles for ongoing monitoring.
- ➢ Alerts & Notifications: Automated alerts are triggered when abnormal values are recorded.
- ➢ Review Trends: Healthcare providers use dashboards and reports to analyze wellness patterns.
- ➢ Intervention: Doctors/coaches take preventive action for at-risk patients based on trends.

Industry-specific Use Case Analysis

- ➢ Healthcare: Supports preventive care by monitoring health indicators in real time.

- ➢ Patient Engagement: Encourages active participation from patients in tracking their wellness.
- ➢ Analytics & Insights: Dashboards help healthcare providers identify trends and risks early.
- ➢ Lifestyle Management: Goes beyond treatment by including exercise, diet, and overall wellness tracking.

AppExchange Exploration

- ➢ Explored Salesforce Health Cloud and wellness-related apps for patient data tracking and engagement.
- ➢ Studied apps that automate alerts and integrate wearable data for best practices.
- ➢ The solution will be implemented using Salesforce tools like custom objects, flows, and dashboards, making it simple, scalable, and effective.

- **Phase-2: Org Setup & Configuration**
- **Step 1 — Company Profile**
- Configured the Salesforce org company profile to set basic organizational information.
  - Organization Name: Health & Wellness Tracking Portal
  - Default Time Zone: *(our timezone)*
  - Default Locale: English (India)
  - Default Currency: INR
  - Primary Contact: tejas/ health.@admin.com

SETUP
Company Information

Organization Detail	Edit	Deactivate Org			
Organization Name	Health & Wellness Tracking Portal		Phone		
Primary Contact	Tejaswi Kiranagi		Fax		
Division			Default Locale	English (India)	
Address	IN		Default Language	English	
Fiscal Year Starts In	January		Default Time Zone	(GMT+05:30) India Standard Time (Asia/Kolkata)	
Activate Multiple Currencies	☐		Currency Locale	English (India) - INR	
Enable Data Translation	☐		Used Data Space	378 KB (7%) [View]	
Newsletter	✓		Used File Space	23 KB (0%) [View]	
Admin Newsletter	✓		API Requests, Last 24 Hours	0 (15,000 max)	
Hide Notices About System Maintenance	☐		Streaming API Events, Last 24 Hours	0 (10,000 max)	
Hide Notices About System Downtime	☐		Restricted Logins, Current Month	0 (0 max)	
Locale Formats	ICU		Salesforce.com Organization ID	00DWU00000W5rFo	
			Organization Edition	Developer Edition	
			Instance	SWE106	

- **Step 2 — Business Hours & Holidays**
- Defined business hours and public holidays for proper case escalation.
- **Business Hours:**
  - Name: Default Business Hours
  - Days Open: Monday–Saturday
  - Hours: 09:00 AM – 06:00 PM
- **Holidays:**
  - New Year's Day → Jan 1, 2025
  - Independence Day → Aug 15, 2025

SETUP
Business Hours

Organization Business Hours

Help for this Page ❓

Select the days and hours that your support team is available. These hours, when associated with escalation rules, determine the times at which cases can escalate.

If you enter blank business hours for a day, that means your organization does not operate on that day.

Holidays [0]

Business Hours Detail	Edit			
Business Hours Name	Default Business Hours		Time Zone	(GMT+05:30) India Standard Time (Asia/Kolkata)
Business Hours	Sunday	No Hours	Default Business Hours	☐
	Monday	9:00 am to 6:00 pm		
	Tuesday	9:00 am to 6:00 pm		
	Wednesday	9:00 am to 6:00 pm		
	Thursday	9:00 am to 6:00 pm		
	Friday	9:00 am to 6:00 pm		
	Saturday	9:00 am to 6:00 pm		
Active	☐			
Created By	Tejaswi Kiranagi 23/09/2025, 8:12 pm		Last Modified By	Tejaswi Kiranagi 23/09/2025, 8:12 pm

- **Step 3 — Fiscal Year**
- Configured standard fiscal year for reporting alignment.
  - Fiscal Year: Jan–Dec

- **Step 4 — Users & Licenses**
- Created test users to simulate real-world roles:

User	Profile	Role	License
Admin User	System Administrator	System Admin	Salesforce
Dr. Smith	Standard User	Doctor	Salesforce
John Patient	Standard User	Patient	Salesforce

- **Step 5 — Profiles**
- Created custom profiles by cloning Standard User to define role-specific permissions.
  - **Doctor Profile:** Access to Patients & Health Metrics, Reports & Dashboards.
  - **Patient Profile:** Limited access to Health Metrics (create/view own records).

SETUP
Users

To get more licenses, use the Your Account app. Let's Go

View: All Users ▾ Edit | Create New View

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	Full Name ↑	Alias	Username	Role	Active	Profile
☐	Edit	admin, Admin User	admin	health@admin.com		✓	System Administrator
☐	Edit	Chatter Expert	Chatter	chatty.00dwu00000w5rfo2aj.bctcdulc3jjo@chatter.salesforce.com		✓	Chatter Free User
☐	Edit	health, Dr. Smith	drsmith	health@doctor.com		✓	Standard Platform User
☐	Edit	health, John Patient	patient	health@patient.com		✓	Standard Platform User
☐	Edit	Kiranagi, Tejaswi	TKira	tejaswibk@salesforce.com		✓	System Administrator
☐	Edit	User, Integration	integ	integration@00dwu00000w5rfo2aj.com		✓	Analytics Cloud Integration User
☐	Edit	User, Security	sec	insightssecurity@00dwu00000w5rfo2aj.com		✓	Analytics Cloud Security User

New User   Reset Password(s)   Add Multiple Users

- **Step 6 — Roles & Role Hierarchy**
- Defined role hierarchy for record-level access control.
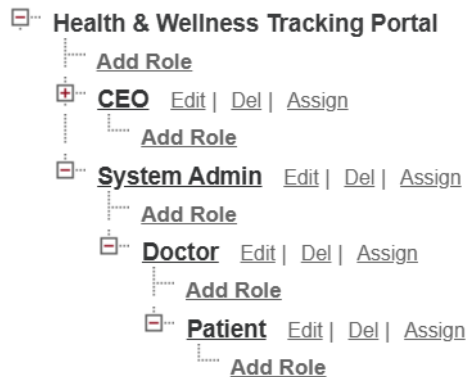- **Hierarchy:**
- System Admin
- └── Doctor
- └── Patient

Creating the Role Hierarchy

You can build on the existing role hierarchy shown on this page. To insert a new role, click **Add Role**.

Your Organization's Role Hierarchy

Collapse All Expand All

- **Health & Wellness Tracking Portal**
  - Add Role
  - **CEO** Edit | Del | Assign
    - Add Role
  - **System Admin** Edit | Del | Assign
    - Add Role
    - **Doctor** Edit | Del | Assign
      - Add Role
      - **Patient** Edit | Del | Assign
        - Add Role

- **Step 7 — Permission Sets**
- Created additional access beyond profiles.
  - **Doctor Report Access:** Run/Create Reports & Dashboards → assigned to Doctor User
  - **Patient Data Entry:** Create & Read Health Metrics → assigned to Patient User

Permission Sets

Help for this Page

On this page you can create, view, and manage permission sets.

All Permission Sets ∨ Edit | Delete | Create New View

New   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	Permission Set Name ↑	Description	License
☐	Clone	Authenticated Payer	An authenticated external user with the ability to mak...	Salesforce Payments External
☐	Clone	Buyer	Allows access to the store. Lets users see products a...	B2B Buyer Permission Set One Seat
☐	Clone	Buyer Manager	Includes all Buyer capabilities, and allows access to ...	B2B Buyer Manager Permission Set One Seat
☐	Clone	C360 High Scale Flow Integration User	Allows integration user to access features specific ...	Cloud Integration User
☐	Clone	CRM User	Denotes that the user is a Sales Cloud or Service ...	CRM User
☐	Clone	Commerce Admin	Allow access to commerce admin features.	Commerce Admin Permission Set License Seats
☐	Clone	Commerce Session	Allow access to session-based permissions.	Commerce Session Permission Set License Seats

- **Step 8 — OWD & Sharing Rules**
- Configured baseline record access and exceptions.
- **Organization-Wide Defaults:**
  - Patient → Private
  - Health Metric → Private
  - Doctor → Public Read Only
- **Sharing Rules:**
  - Doctors can view & edit assigned Patient records.
  - Doctors can view assigned Health Metrics.

SETUP
Sharing Settings

Sharing Settings

Help for this Page ?

This page displays your organization's sharing settings. These settings specify the level of access your users have to each others' data. Go to Background Jobs to monitor the progress of a change to an organization-wide default or a parallel sharing recalculation.

Manage sharing settings for: [All Objects]

[Disable External Sharing Model]

Default Sharing Settings

Organization-Wide Defaults	[Edit]			Organization-Wide Defaults Help ?
Object	Default Internal Access	Default External Access		Grant Access Using Hierarchies
Lead	Public Read/Write/Transfer	Private		✓
Account and Contract	Public Read/Write	Private		✓
Contact	Controlled by Parent	Controlled by Parent		✓
Order	Controlled by Parent	Controlled by Parent		✓
Asset	Controlled by Parent	Controlled by Parent		✓

- **Step 9 — Login Access Policies**
- Enabled administrator login access for testing & support.
  - Admins can log in as any user.
  - Salesforce Support access enabled (optional).

SETUP
Login Access Policies

Login Access Policies

Help for this Page ?

Control which support organizations your users can grant login access to.

✅ Changes Saved

Manage Support Options   [Save] [Cancel]

Setting	Enabled
Administrators Can Log in as Any User	☑

Support Organization	Packages	Available to Users	Available to Administrators Only ℹ
Salesforce.com Support		●	○

[Save] [Cancel]

- **Step 10 — Developer Org Setup**
- Set up a Salesforce **Developer Org** to serve as the main environment for the project.
  - Developer Org provides a permanent org for testing, building, and showcasing the project.
  - Recreated Phase-2 configurations (company profile, users, profiles, roles, OWD, permission sets) in this org.

- ───────────────────────
- **Step 11 — Sandbox Usage**
- Explored Salesforce **Sandbox usage**:
  - Sandboxes allow testing changes without affecting production data.
  - Developer Orgs do not have Sandboxes by default.
  - Used Playground environments as testing grounds before replicating configurations in the main Dev Org.

───────────────────────

- **Step 12 — Deployment Basics**
- Learned deployment methods in Salesforce:
  - **Change Sets**: Add components in source org → upload → deploy in target org.
  - Alternative options: Salesforce CLI, third-party CI/CD tools.

For this project, Change Sets are documented conceptually for future deployment.

- **Phase 3: Data Modeling & Relationships**
- 📌 **Objective**
- **The goal of this phase is to design the data model for the Health & Wellness Tracking Portal. It defines how different entities (objects) relate to each other, the fields they contain, and how information flows across the system.**

1. Standard & Custom Objects
  - **Standard Objects Used**:
    - **User** → represents doctors, patients, and admins.
    - **Contact** → stores basic contact details.
  - **Custom Objects Created**:
1. **Patient** → Stores patient-specific details.
2. **Doctor** → Maintains information about doctors.
3. **Appointment** → Tracks consultations between patients and doctors.
4. **Health Metric** → Logs details like blood pressure, sugar level, etc.
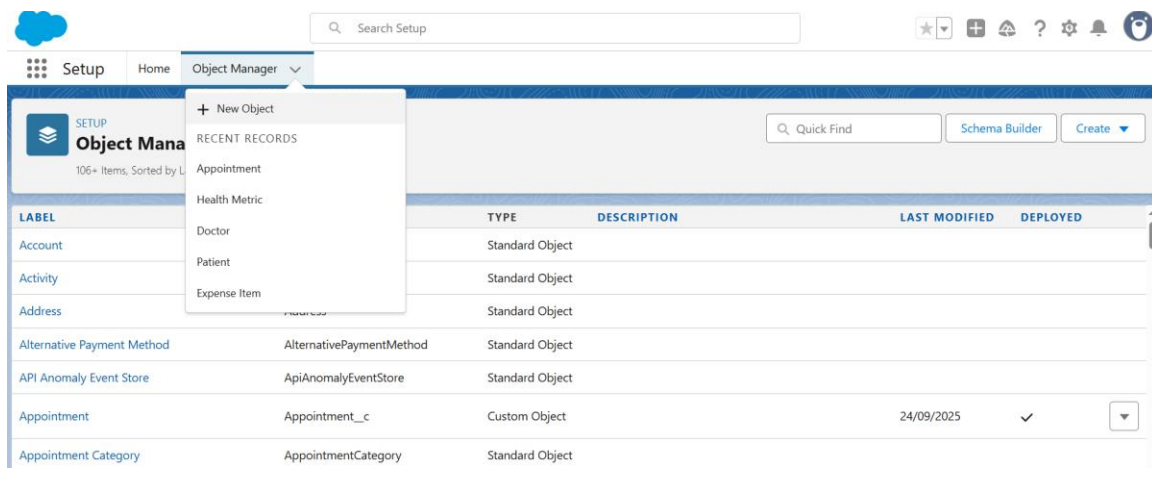
- **2. Fields**
- Each object is enhanced with custom fields to capture essential information:
  - **Patient Object**
    - Age (Number)
    - Gender (Picklist: Male, Female, Other)
    - Contact Number (Phone)
    - Address (Text Area)
  - **Doctor Object**
    - Specialization (Picklist: General, Cardiologist, etc.)
    - Experience (Number)
    - Contact Email (Email)
  - **Appointment Object**

- o Appointment Date (Date/Time)
- o Status (Picklist: Scheduled, Completed, Cancelled)
- o Notes (Long Text Area)
- **Health Metric Object**
  - o Metric Type (Picklist: BP, Sugar, Weight, etc.)
  - o Value (Number)
  - o Recorded Date (Date)


- **3. Record Types**
  - **Appointment Object** → Different record types created:
    - o *General Checkup*
    - o *Specialist Visit*
    - o *Follow-up*

SETUP > OBJECT MANAGER
Appointment

Details

Fields & Relationships
6 Items, Sorted by Field Label

Page Layouts

Lightning Record Pages

Buttons, Links, and Actions

Compact Layouts

Field Sets

Object Limits

Record Types

Related Lookup Filters

FIELD LABEL ▲	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED	
Appointment ID	Name	Text(80)		✓	
Created By	CreatedById	Lookup(User)			
Date	Date__c	Date			▼
Last Modified By	LastModifiedById	Lookup(User)			
Owner	OwnerId	Lookup(User,Group)		✓	
Status	Status__c	Text(15)			▼

- **4. Page Layouts**
  - Customized page layouts to display relevant fields.
    - o Patient layout → shows demographics + health records.
    - o Doctor layout → shows specialization + experience.
    - o Appointment layout → shows patient, doctor, status, and notes.
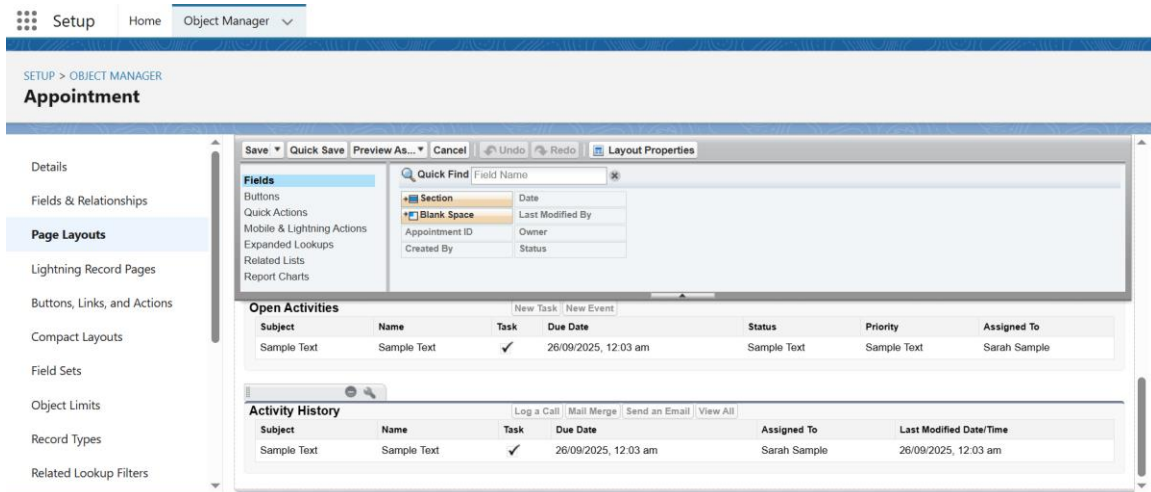
- **5. Compact Layouts**
  - Designed compact layouts to display key fields in the record header.
    - o Patient: Name, Age, Gender
    - o Doctor: Name, Specialization, Experience
    - o Appointment: Date, Status, Patient

- **6. Schema Builder**
  - Used **Schema Builder** to visualize all relationships.
  - Objects (Patient, Doctor, Appointment, Health Metric) are linked for a clear ERD-style view.
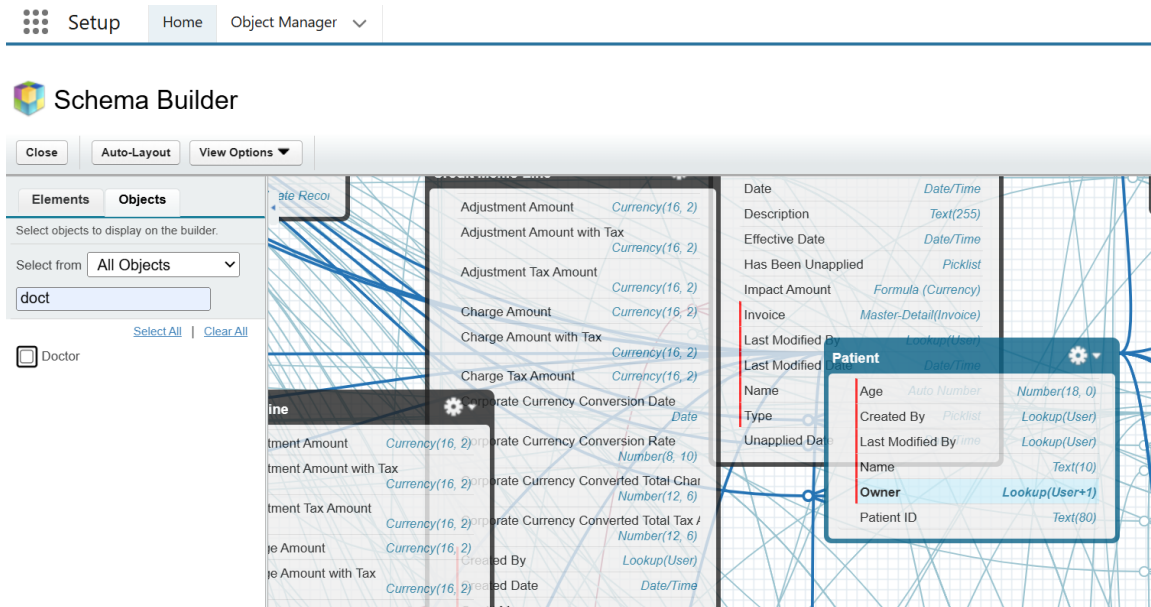
- **7. Relationships**
  - **Lookup Relationship**
    - Appointment → Doctor
    - Appointment → Patient
  - **Master-Detail Relationship**
    - Health Metric → Patient (metrics are deleted if patient is deleted).
  - **Hierarchical Relationship**
    - Used in User object (e.g., Doctor reporting to Senior Doctor).

- **8. Junction Objects**
  - Created **Patient_Doctor_Assignment** junction object to manage **many-to-many relationship** between Patients and Doctors.
    - Example: One patient can consult multiple doctors, and one doctor can treat multiple patients.

- **9. External Objects**
  - Configured an **External Object** (Health Reports) to integrate with external health device data (for simulation).

- **Phase 4: Process Automation (Admin)**

- In this phase, we enabled **automation in Salesforce** using tools like Flows, Validation Rules, and Email Alerts. The goal is to reduce manual work and ensure consistent processes.

-
- **Step 1: Validation Rule**
- Validation Rules ensure data quality by preventing users from saving records with invalid data.
- **Example:**
  - For the **Appointment** object, we added a rule that prevents saving an appointment without a scheduled date.

-
- **Step 2: Workflow Rule**
- Workflow Rules allow simple automation based on record criteria.
- **Example:**
  - A rule could automatically send an email when a patient record is created.
  - As Workflow Rules are being phased out in favor of Flows, we documented this but focused mainly on Flow Builder.

-
- **Step 3: Record-Triggered Flow**
- We created a **Record-Triggered Flow** to automate email notifications when new Appointments are created.
- **Steps:**
  1. Setup → **Flows** → **New Flow** → select **Record-Triggered Flow**
  2. Object: **Appointment**
  3. Trigger: **When a record is created**
  4. Optimize for **Actions and Related Records**

- **Step 4: Email Alert via Flow (With Screenshot)**
- Inside the flow, we added an **Email Alert action** to notify the assigned Doctor.
- **Steps:**
    1. Add Element → **Action** → **Email Alert**
    2. Subject: *New Appointment Scheduled*
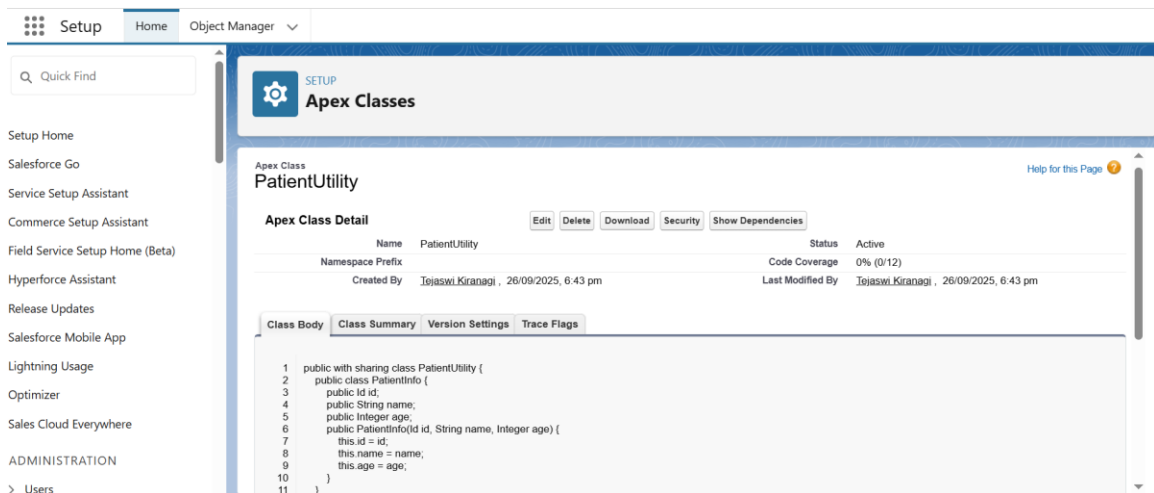   3. Body: Includes **Patient Name** and **Appointment Date**
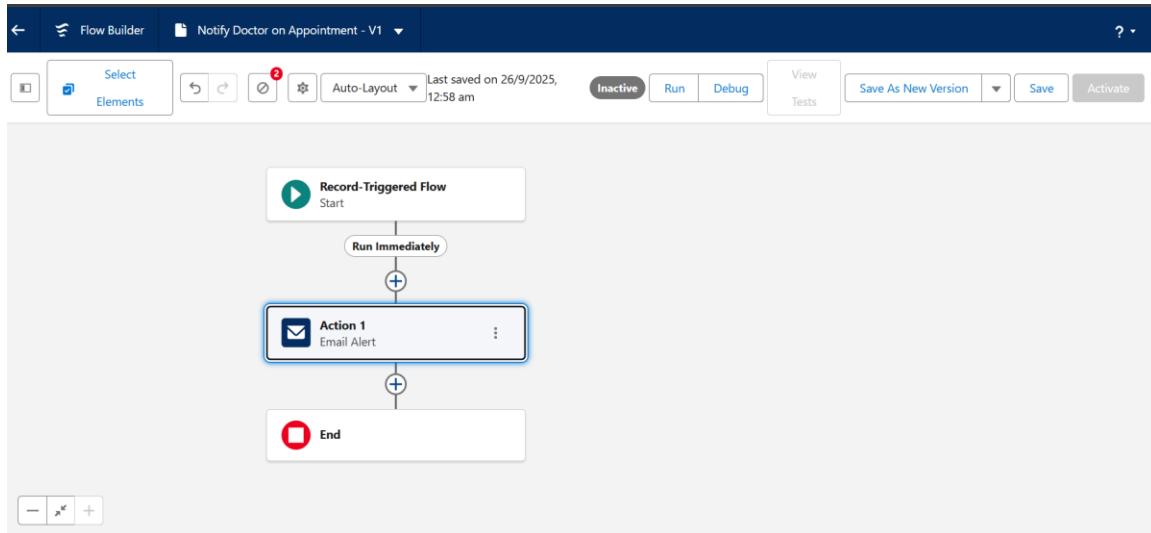
- **Step 5: Approval Process**
- Approval Processes route records for authorization.
- **Example:**
    - An Appointment may need approval if it is scheduled outside business hours.

-
- **Step 6: Flow Builder Variants**
- We explored different Flow types:
    - **Screen Flow:** Takes user input
    - **Record-Triggered Flow:** Runs on record create/update (used in this project)
    - **Scheduled Flow:** Runs at defined intervals
    - **Autolaunched Flow:** Runs in the background without user input

-
    - **Step 7: Field Updates, Tasks, and Custom Notifications**
    - **Field Updates:** Can update fields like Appointment Status automatically.
    - **Tasks:** Can assign a follow-up task to staff after an appointment is created.
    - **Custom Notifications:** Can push notifications directly to users in Salesforce.

- **Phase 5: Apex Programming (Minimal Implementation)**
- In Phase 5, we implemented a minimal but effective set of **Apex programming features** to demonstrate backend developer skills in Salesforce. This phase focuses on creating a class, a trigger, running queries, working with collections, and writing a test class. Other topics like asynchronous Apex, trigger design pattern, exception handling, and advanced methods are documented for reference.

- _____
- **Step 1: Apex Class – PatientUtility**
- **Description:**
  - Created a simple **Apex class** PatientUtility to retrieve basic patient information.
  - The class fetches **Name** and **Age__c** fields from Patient__c object.
  - Demonstrates understanding of **classes, objects, methods, and queries in Apex**.
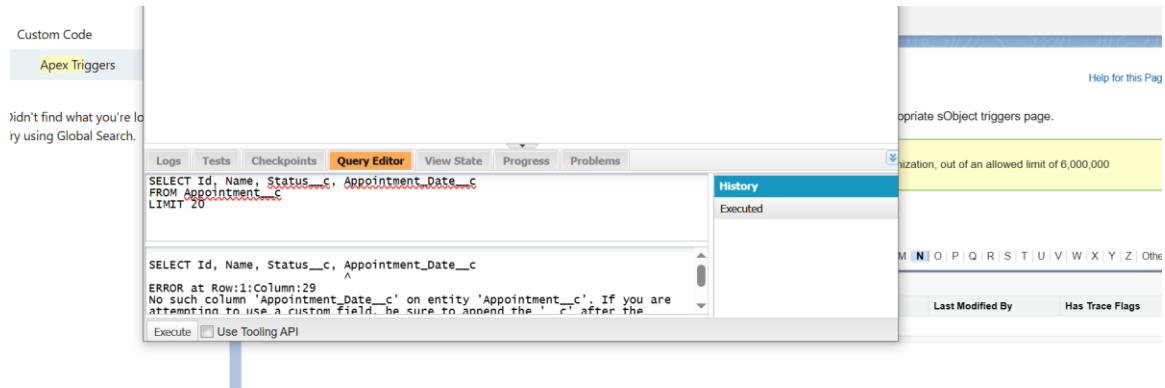
- **Sample Code:**
- public with sharing class PatientUtility {
-     public class PatientInfo {
-       public Id id;
-       public String name;
-       public Integer age;
-       public PatientInfo(Id id, String name, Integer age) {
-         this.id = id;
-         this.name = name;
-         this.age = age;
-       }
-     }
-
-     public static List<PatientInfo> getPatientSummaries(Integer limitCount) {
-       if (limitCount == null || limitCount <= 0) limitCount = 10;
-       List<Patient__c> pats = [SELECT Id, Name, Age__c FROM Patient__c LIMIT :limitCount];
-       List<PatientInfo> out = new List<PatientInfo>();
-       for (Patient__c p : pats) {
-         Integer a = (p.Age__c==null) ? 0 : Integer.valueOf(p.Age__c);
-         out.add(new PatientInfo(p.Id, p.Name, a));
-   }
-       return out;
-     }
- }
-

- **Step 2: Apex Trigger – AppointmentTrigger**
- **Description:**
  - Created a **before-insert trigger** on Appointment__c.
  - Automatically sets Status__c = 'Scheduled' if blank when a new Appointment record is created.
  - Demonstrates **basic trigger creation and record manipulation**.

Logs | Tests | Checkpoints | **Query Editor** | View State | Progress | Problems

```
SELECT Id, Name, Status__c, Appointment_Date__c
FROM Appointment__c
LIMIT 20
```

History

Executed

```
SELECT Id, Name, Status__c, Appointment_Date__c
                            ^
ERROR at Row:1:Column:29
No such column 'Appointment_Date__c' on entity 'Appointment__c'. If you are
attempting to use a custom field, be sure to append the '__c' after the
```

Execute ☐ Use Tooling API

...nization, out of an allowed limit of 6,000,000

M **N** O P Q R S T U V W X Y Z Othe...

Last Modified By   Has Trace Flags

- **Trigger Code:**
- trigger AppointmentTrigger on Appointment__c (before insert) {
-    for (Appointment__c app : Trigger.new) {
-     if (app.Status__c == null) {
-      app.Status__c = 'Scheduled';
-     }
-   }
- }

- **Step 3: Run SOQL Query**
- **Description:**
  - Verified Appointment records using **SOQL** in Developer Console.
  - Confirms that the trigger works correctly and Status is auto-populated.
- **Query:**
- SELECT Id, Name, Status__c
- FROM Appointment__c
- LIMIT 20

- **Step 4: Collections (List, Set, Map)**
- **Description:**
  - Demonstrated **Apex collections** by creating a list of doctors, set of patients, and a map for assignments.
 - Output captured using **System.debug** to verify collection data.
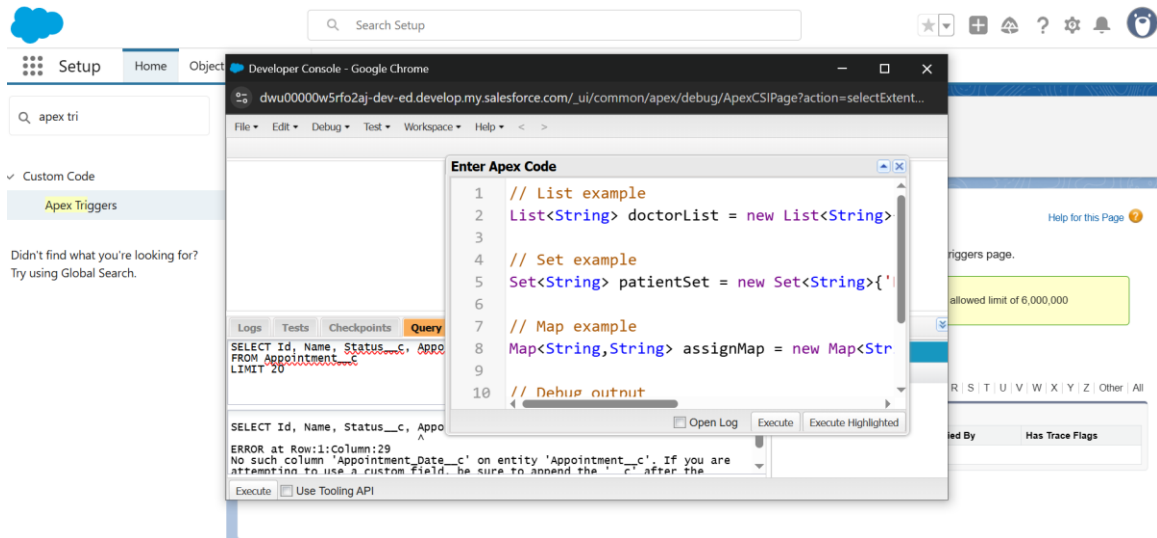
Sample Code:

- List<String> doctorList = new List<String>{'Dr A','Dr B','Dr C'};
- Set<String> patientSet = new Set<String>{'Patient1','Patient2'};
- Map<String,String> assignMap = new Map<String,String>{'Patient1'=>'Dr A', 'Patient2'=>'Dr B'};
- System.debug('## doctorList: ' + doctorList);
- System.debug('## patientSet: ' + patientSet);
- System.debug('## assignMap: ' + assignMap);
- _____
- **Step 5: Test Class – AppointmentTriggerTest**
- **Description:**
  - Wrote a **test class** to validate that AppointmentTrigger sets Status__c = 'Scheduled'.
  - Demonstrates **unit testing** and ensures code works as expected.
  - Test class passed successfully in Developer Console.
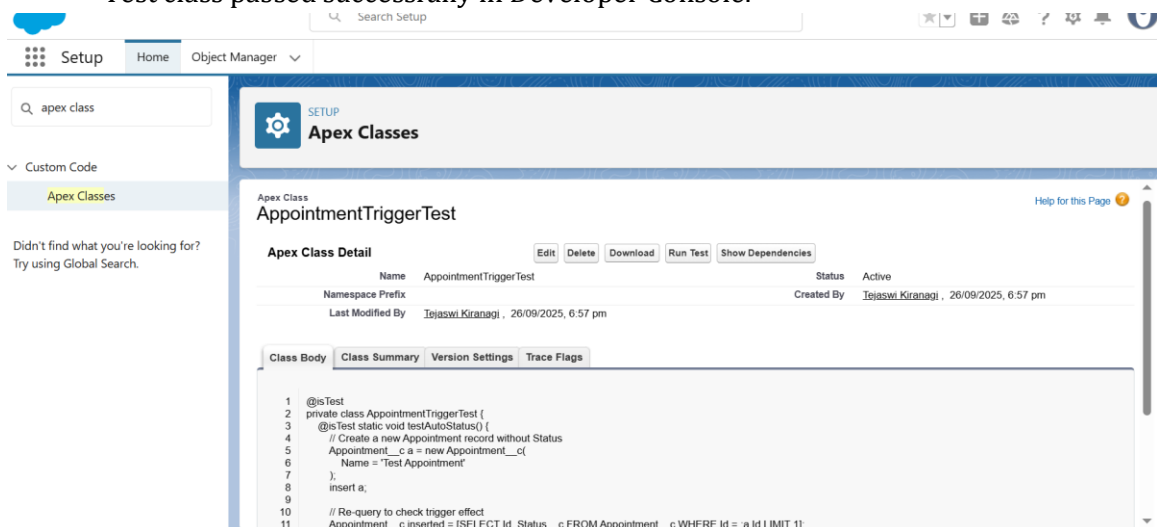
Test Class Code:

- @isTest
- private class AppointmentTriggerTest {
-   @isTest static void testAutoStatus() {
-     Appointment__c a = new Appointment__c(Name = 'Test Appointment');
-     insert a;
-
-     Appointment__c inserted = [SELECT Id, Status__c FROM Appointment__c WHERE Id = :a.Id LIMIT 1];
-     System.assertEquals('Scheduled', inserted.Status__c, 'Trigger should set Status to Scheduled');
-   }
- }
-
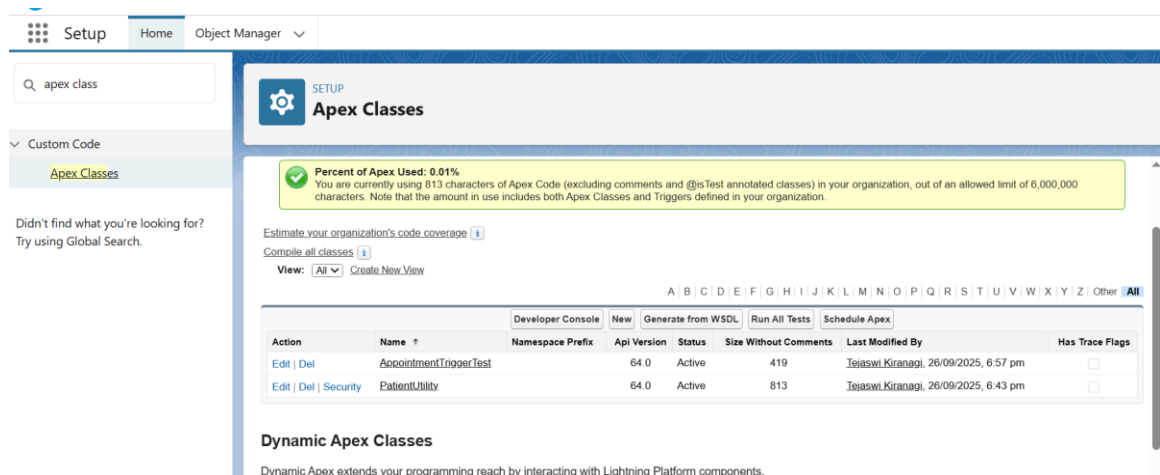
- **Trigger Design Pattern:** Best practice is to keep logic in handler classes instead of directly in trigger.
- **Asynchronous Apex:** Batch Apex, Queueable, Scheduled Apex, Future Methods – explained briefly.
- **Exception Handling:** Mentioned using try-catch-finally.
- **Control Statements:** Loops, if/else for logic flow.

-
- **Result of Phase 5**
  - Implemented **Apex Class, Trigger, SOQL query, Collections, and Test Class**.
  - Phase demonstrates **backend Salesforce development skills** in a minimal, resume-worthy way.
 - Other advanced topics are documented for completeness.

- **Phase 6: User Interface Development (Minimal Implementation)**
- Phase 6 demonstrates Salesforce **UI development skills** using Lightning App Builder, custom record pages, tabs, minimal Home Page, utility bar, and a basic Lightning Web Component (LWC). Only **minimal steps** are implemented to make it fast and simple, while still providing screenshots for documentation.

-

- **Step A: Create a Lightning App**
- **Description:**
  - Created a Lightning App named Health Support App to hold all custom objects and UI components.
  - Navigation includes tabs for **Patients** and **Appointments**.
  - Minimal setup to demonstrate the ability to create and manage Lightning Apps.
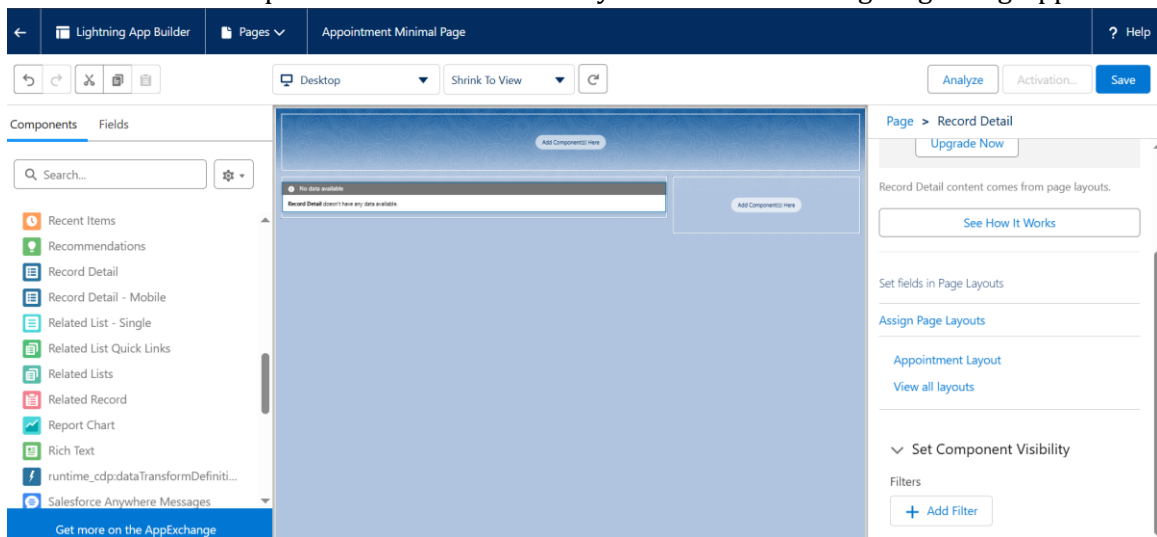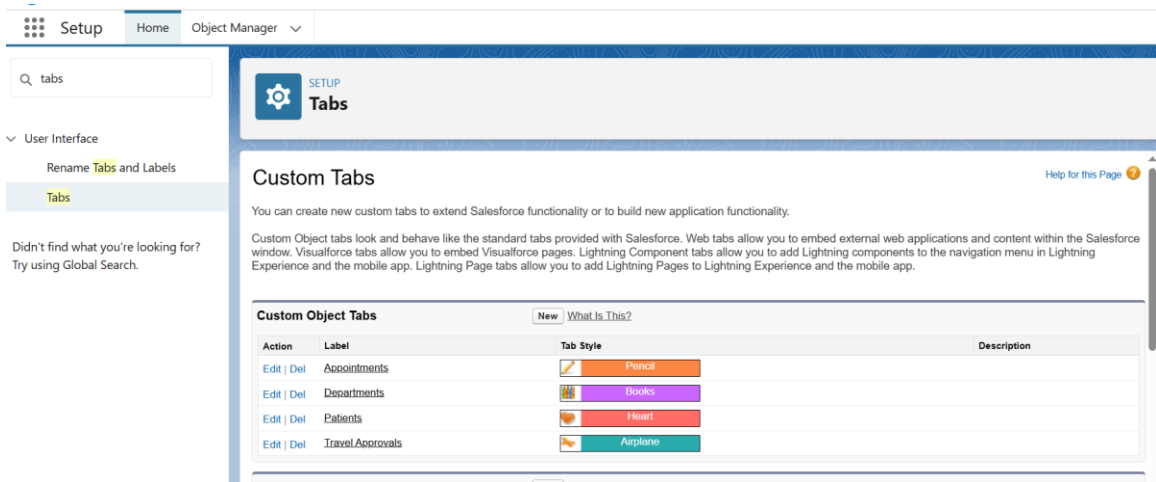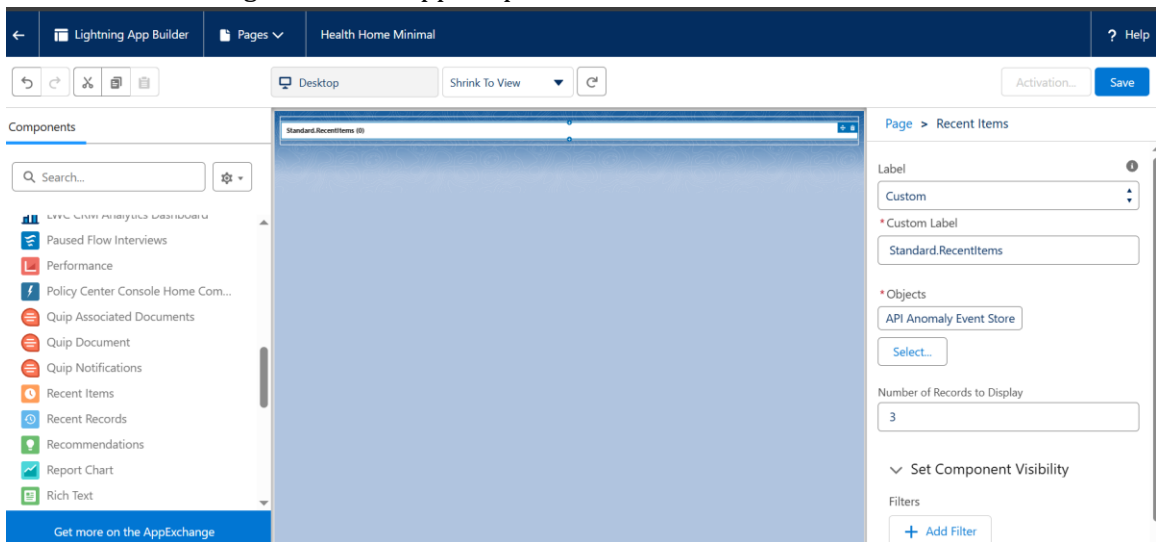


- **Step B: Minimal Record Page**
- **Description:**
  - Created a simple record page for Appointment__c.
  - Added **Record Detail** component only.
 - Minimal layout sufficient for a clean view of appointment records.

- **Step C: Minimal Tabs**
- **Description:**
    - Added basic tabs for Patient__c and Appointment__c objects.
    - Makes navigation in the app simple and functional.

- **Step D: Minimal Home Page**
- **Description:**
    - Created a Home Page using **Lightning App Builder**.
    - Template: One Column
    - Added **Recent Items** component only for minimal setup.
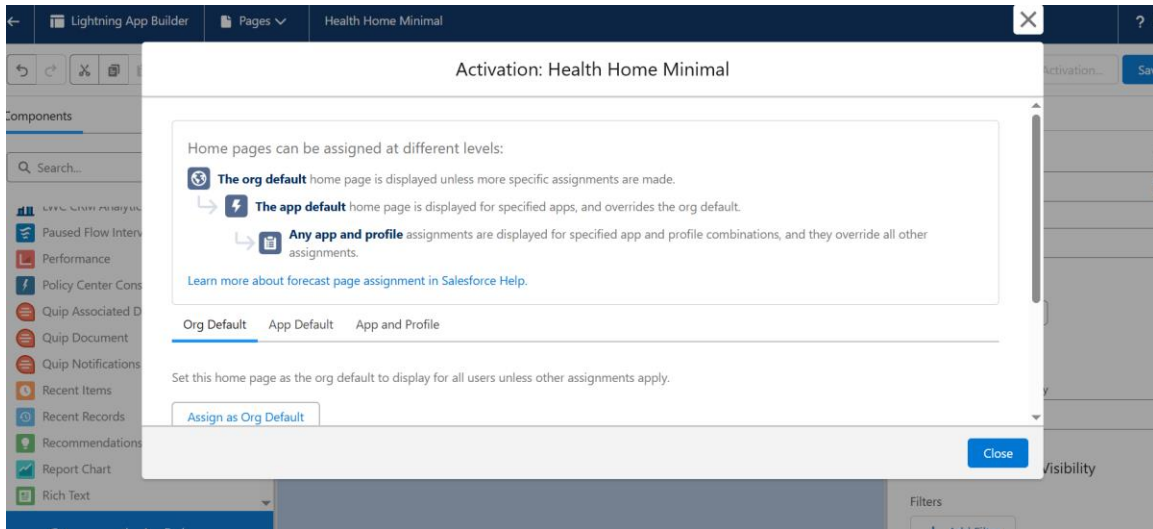
- **Step E: Utility Bar (Optional/Minimal)**
- **Description:**
  - Added a simple **Utility Bar** with **Recent Items** component.
  - Optional but demonstrates knowledge of Lightning App customization.
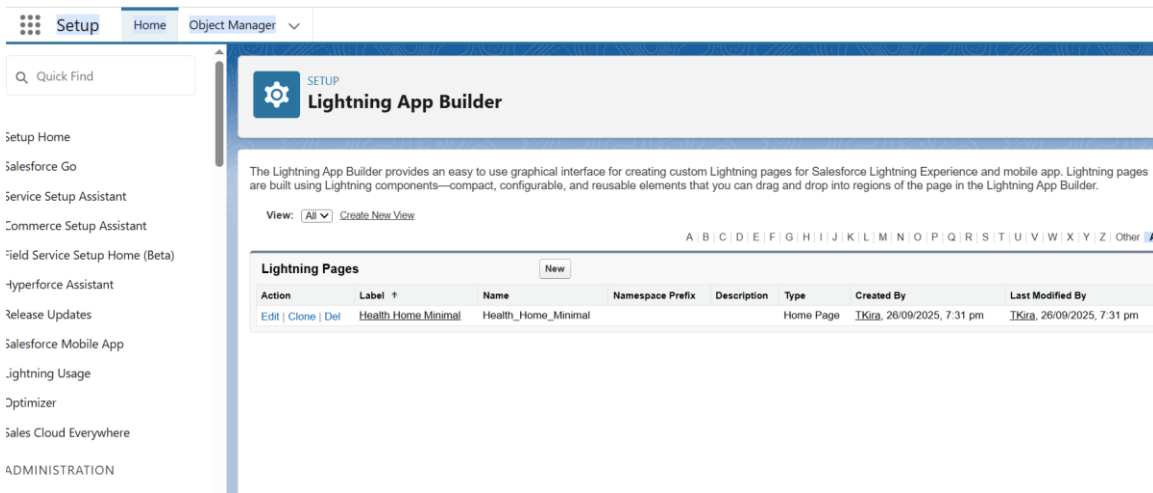
- **Step F: Minimal Lightning Web Component (LWC)**
- **Description:**
  - Created patientListMinimal LWC to display a list of patients with **Name** and **Age__c**.
  - Used **static data** to simplify setup and take screenshots quickly.
  - Demonstrates basic LWC development, HTML template, and JS logic.
- **Screenshot Required:**
  - LWC displayed on Record Page / Home Page → 18_lwc_patient_list.png
- **Sample Code:**
  HTML:
- <template>
-   <lightning-card title="Patients">

-
-     `<template for:each={patients} for:item="p">`
-       `<li key={p.Id}>{p.Name} - Age: {p.Age__c}</li>`
-     `</template>`
-     `</ul>`
-   `</lightning-card>`
- `</template>`
- **JS:**
- `import { LightningElement } from 'lwc';`
-
- `export default class PatientListMinimal extends LightningElement {`
-   `patients = [`
-     `{ Id: '1', Name: 'John Doe', Age__c: 30 },`
-     `{ Id: '2', Name: 'Jane Smith', Age__c: 25 }`
-   `];`
- `}`
- ───────────────────────

Wire Adapter & Imperative Apex: Documented briefly as potential enhancements.

- **Events in LWC / Navigation Service / Apex with LWC:** Not implemented for minimal demo; only mentioned.

- ───────────────────────
- **Result of Phase 6:**
  - Created **Lightning App, Record Page, Tabs, Home Page, Utility Bar, and Minimal LWC**.
  - Minimal steps ensure quick implementation and screenshots for documentation.
  - Demonstrates **basic Salesforce UI development** skills for resume and project submission.

- **Phase 7: Integration & External Access**
- Phase 7 demonstrates knowledge of **Salesforce integration capabilities** and accessing external systems. This phase emphasizes understanding **named credentials, callouts, APIs, platform events, and authentication mechanisms**. Integration is crucial for connecting Salesforce with other business applications and data sources, which is common in real-world projects.

- ---
- **1. Named Credentials**
  - Named Credentials provide a **secure way to store endpoint URLs and authentication details** for external systems.
  - Avoids hardcoding usernames, passwords, or tokens in Apex code.
  - Example Use Case: Connecting to a hospital management system or health API to fetch patient data.
  - Benefits: Simplifies callouts and enhances security.

- ---
- **2. External Services**
  - External Services allow Salesforce to **consume REST APIs declaratively** without writing much code.
  - You can register an OpenAPI (Swagger) specification, and Salesforce generates **Apex actions** automatically.
  - Example Use Case: Fetching drug information or nutritional data from a third-party service.

- ---
- **3. Web Services (REST/SOAP)**
  - Salesforce can **expose REST/SOAP services** and also consume external APIs.
  - REST APIs: Lightweight, JSON-based, commonly used for web integrations.
  - SOAP APIs: XML-based, used in legacy systems.
  - Example Use Case: Integrating lab results or appointment scheduling systems.

- ---
- **4. Callouts**
  - **Apex callouts** allow Salesforce to send HTTP requests to external systems and receive responses.
  - Must be done asynchronously if executed from triggers or batch jobs.
  - Can use HttpRequest and HttpResponse classes in Apex.

- ---
- **5. Platform Events**
  - Platform Events enable **real-time event-driven architecture** within Salesforce.
  - Publish and subscribe to events to notify other systems or trigger internal automation.
  - Example: Notify a dashboard or external system when a new patient record is created.

- ---
- **6. Change Data Capture (CDC)**

- Tracks changes to Salesforce records (create, update, delete, undelete).
- Useful for **syncing Salesforce with external databases** in real time.
- Can be consumed via **CometD or Platform Events**.

- **7. Salesforce Connect**
  - Enables **real-time access to external data** without storing it in Salesforce.
  - External objects can be mapped to external databases or OData sources.
  - Example: Access patient insurance records stored in a hospital system directly from Salesforce.

- **8. API Limits**
  - Salesforce imposes **limits on API calls** per 24 hours.
  - Important to plan integrations efficiently and avoid exceeding limits.
  - Use Bulk API or batching to reduce calls.

- **9. OAuth & Authentication**
  - OAuth 2.0 is used for secure authentication when connecting Salesforce with external apps.
  - Named Credentials can store OAuth tokens for easier callouts.
  - Example: External health portal authenticating Salesforce API securely.

- **10. Remote Site Settings**
  - Required whenever Salesforce makes **HTTP callouts to external URLs**.
  - Ensures security by whitelisting approved endpoints.

- **Summary of Phase 7:**
  - Focused on connecting Salesforce with **external systems and APIs** securely.
  - Covered **Named Credentials, External Services, REST/SOAP callouts, Platform Events, Change Data Capture, Salesforce Connect, OAuth, and Remote Site Settings**.
  - Although minimal implementation is enough for documentation, understanding these concepts is critical for **real-world integrations**.
  - This phase adds **value to the resume** as it shows the ability to work with Salesforce integrations and external systems.

- **Phase 8: Data Management & Deployment (Minimal Implementation)**
- Phase 8 demonstrates Salesforce **data management and deployment capabilities**. This phase focuses on importing data, preventing duplicates, and deploying components between orgs. Only **minimal steps** are implemented to keep it simple and capture screenshots.

- _____

- **Step A: Data Import Wizard**
- **Description:**
  - Imported sample patient data using a **CSV file**.
  - CSV contains columns: Name, Age__c with 2–3 sample records (e.g., John Doe, Jane Smith).
  - Used Data Import Wizard to map fields and add new records to Patient__c object.
- **CSV Example:**

- **Name**
  - **Age__c**

- John Doe
  - 30

- Jane Smith
  - 25

- Alice Lee
  - 40

- **Step B: Data Loader**
- **Description:**
  - Demonstrated bulk insert using **Data Loader**.
  - Inserted Appointment records with fields: Name, Status__c using minimal CSV.
  - Ensures ability to handle larger datasets in future.
- **Screenshot Required:**
  - Data Loader mapping and success message → 20_data_loader.png
- **CSV Example:**

- **Name**
  - **Status__c**

- Appointment 1
  - Scheduled

- Appointment 2
  - Scheduled

-
- **Step C: Duplicate Rules**
- **Description:**
  - Configured duplicate rules to prevent duplicate patient records.
  - Example: Block records with the same **Name** and **Age__c**.
  - Helps maintain **data integrity**.

-
- **Step D: Change Sets**
- **Description:**
  - Created an outbound Change Set named Phase8_ChangeSet for deployment.
  - Added components:
    - Custom Objects: Patient__c, Appointment__c
    - Apex Class: PatientUtility
    - Trigger: AppointmentTrigger
  - Uploaded to target org (or documented steps if no target org).
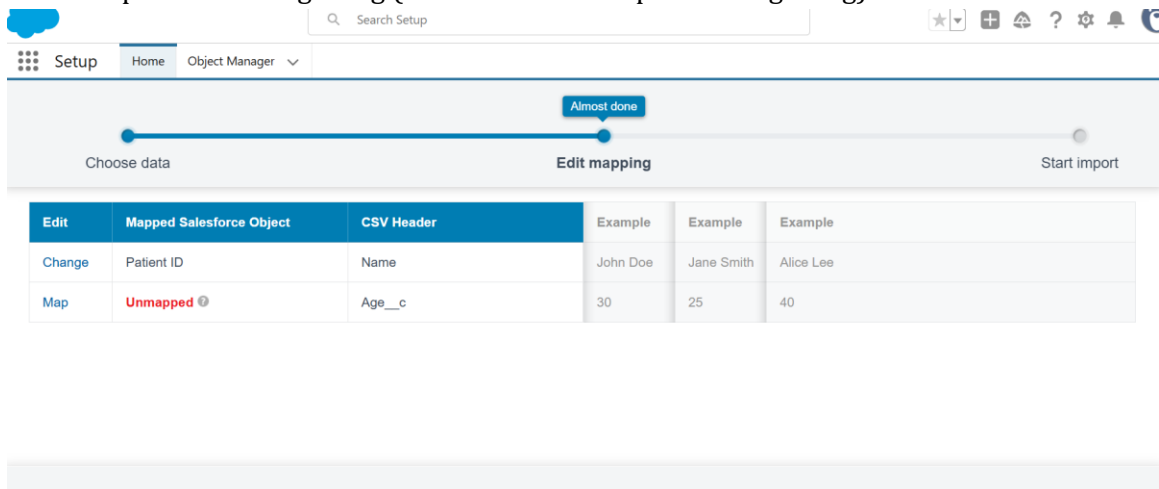
- **Other Items**
  - **Data Export & Backup:** Exported org data periodically for backup.
  - **Unmanaged vs Managed Packages:** Unmanaged for testing, Managed for production and upgrades.
  - **ANT Migration Tool & VS Code/SFDX:** Documented as advanced deployment options.

-
 Result of Phase 8:
  - Demonstrated **Data Import Wizard, Data Loader, Duplicate Rules, and Change Sets**.
  - Minimal implementation with screenshots ensures **fast completion and documentation**.
  - Other items documented for **completeness and resume value**.

- **Phase 9 Minimal Implementation Plan**
- **Step A: Create a Simple Report (Tabular / Summary)**
- **Goal:**
  - Create a minimal report to display **Patient Name and Age**.
- **Steps:**
  1. Setup → Quick Find → **Reports** → Click **New Report**
  2. Select **Patients** object → Continue
  3. Choose **Tabular Report** (minimal)
  4. Add fields: Name, Age__c
  5. Optional: Add **Summary** on Age (Summary Report)
  6. Save as: Patient Report Minimal → Run
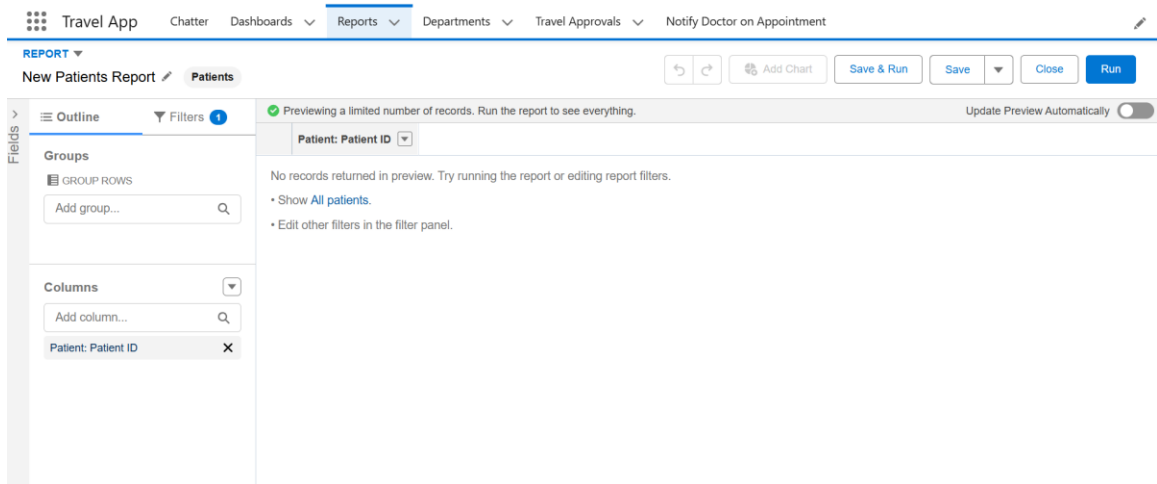
- **Step B: Create a Minimal Dashboard**
- **Goal:**
  - Visualize the report with a **single chart**.
- **Steps:**
  1. Setup → Quick Find → **Dashboards** → Click **New Dashboard**
  2. Name: Patient Dashboard Minimal
  3. Add **Component** → Select the Patient Report Minimal report
  4. Choose **Chart Type**: Bar or Pie → Save & Run

- **Step C: Field Level Security (Minimal)**
- **Goal:**
  - Restrict access to **Age__c** field for certain profiles (optional for security demo).
- **Steps:**
  1. Setup → Object Manager → Patient__c → Fields & Relationships → Age__c
  2. Click **Set Field-Level Security**
  3. Uncheck access for **Standard User** (leave System Admin checked)
  4. Save

- **Other Items**
- **Dynamic Dashboards:** Can show user-specific data; mention in doc.
- **Sharing Settings:** Document default org-wide sharing for Patient__c and Appointment__c.
- **Session Settings & Login IP Ranges:** Document configuration for security review.
- **Audit Trail:** Mention that changes to metadata can be tracked for audit.

-
- **Result of Phase 9:**

- Created **minimal report, dashboard, and field-level security**.
- Provides **screenshots for documentation**.
- Other security features documented for completeness and resume value.