Complete Project Documentation

📌 Phase 1: Problem Understanding & Industry Analysis

This phase sets the **foundation** for your CRM project. The goal is to deeply understand the **problem space**, analyze the **industry requirements**, and align the CRM solution with **stakeholders' expectations**.

🔹 1. Requirement Gathering

What it means:
Requirement gathering is the process of collecting, analyzing, and documenting the needs of the business and end-users. It helps define *what* the CRM system must achieve.

How to do it:

- Conduct **interviews** with stakeholders (customers, agents, managers).
- Send out **surveys** to understand common pain points in complaint handling.
- Analyze **existing complaint logs** (emails, Excel sheets, or manual registers).

Key questions to ask:

- How are customer complaints currently tracked?
- What challenges do agents face in resolving complaints?
- What reports do managers want to see?
- Do customers get regular updates on their complaints?

Example requirements for Customer Complaint CRM:

- Customers should be able to log complaints through **website, email, or phone**.
- Agents should be able to **view, assign, and close complaints**.
- Managers should be able to **analyze complaint trends** and **agent performance**.
- System should send **automatic notifications** when complaint status changes.

Deliverable:
A **Requirement Specification Document** with **must-have features (core)** and **nice-to-have features (future scope)**.

🔹 2. Stakeholder Analysis

What it means:
Identifying the key people (stakeholders) who will be impacted by the CRM and understanding their expectations.

Stakeholder Categories for this project:

1. **Customers (External Stakeholders)**
    - Role: Log complaints and track their resolution.
    - Expectation: Fast response, transparency, updates.
2. **Support Agents (Internal Stakeholders)**
    - Role: Handle and resolve complaints.
    - Expectation: Easy access to customer data, clear case assignment, reduced manual effort.
3. **Managers/Supervisors**
    - Role: Monitor agent performance, oversee complaint resolution.
    - Expectation: Dashboards, SLA compliance tracking, escalations.
4. **System Administrators**

- Role: Configure Salesforce, manage user roles, maintain security.
- Expectation: Customization flexibility, easy maintenance.

5. **Top Management**
   - Role: Ensure customer satisfaction and business growth.
   - Expectation: Reports to measure overall complaint trends and decision-making insights.

Deliverable:
A **Stakeholder Map** (a table or diagram with roles, responsibilities, and expectations).

🔹 3. Business Process Mapping

What it means:
Visualizing how complaints are currently handled (**AS-IS process**) and how the new CRM will improve it (**TO-BE process**).

AS-IS Example (Manual Process):

1. Customer calls customer care → details written on paper.
2. Complaint forwarded via email/WhatsApp.
3. No tracking system → customer keeps following up.
4. Resolution happens but no record of timelines or satisfaction.

TO-BE Example (CRM-enabled Process in Salesforce):

1. Customer submits complaint via **web portal/email/chatbot**.
2. Salesforce automatically creates a **case record**.
3. Case is **auto-assigned** to the right agent (based on skill/rules).
4. Agent updates case progress in real-time.

5. Customer gets **automated notifications** (case created, in-progress, resolved).
6. Managers view **dashboards** for trends, escalations, SLA breaches.

Deliverable:
Business Process Diagrams:

- **AS-IS model** (manual flow)
- **TO-BE model** (Salesforce-enabled flow)

🔹 4. Industry-Specific Use Case Analysis

What it means:
Every industry faces different kinds of customer complaints. Aligning your CRM to one industry makes the project realistic.

Example: Car Rental Industry (if you choose this):

- Common Complaints:
    - Vehicle not available on booking.
    - Poor vehicle condition.
    - Overbilling/hidden charges.
    - Late pickup/drop-off.
    - Refund delays.
- CRM Use Cases:
    - Auto-logging complaints directly from the booking system.
    - Assigning complaints to specialized teams (billing, maintenance, scheduling).
    - Tracking SLAs (e.g., refund within 7 business days).
    - Manager dashboards to identify frequent complaint categories.

Other industries you could consider:

- **Banking**: Failed transactions, fraud reports.
- **Telecom**: Network downtime, billing errors.
- **Retail/E-commerce**: Product quality, delivery issues.

Deliverable:
An **Industry Use Case Document** describing complaint categories and CRM solutions.

🔹 5. AppExchange Exploration

What it means:
Salesforce AppExchange is like an **app store** where you can find ready-made solutions. Exploring it helps you decide whether to build from scratch or enhance existing apps.

How to explore:

1. Go to AppExchange.
2. Search for: *Complaint Management, Case Management, Customer Service*.
3. Review apps like:
    - **Salesforce Service Cloud Extensions** (case management add-ons).
    - **Complaint Management Apps** (industry-specific).
    - **Chatbots/AI Tools** (for customer service automation).

Deliverable:
A **comparison table** of top 2–3 apps with features, pros/cons, and decision whether to use them or not.

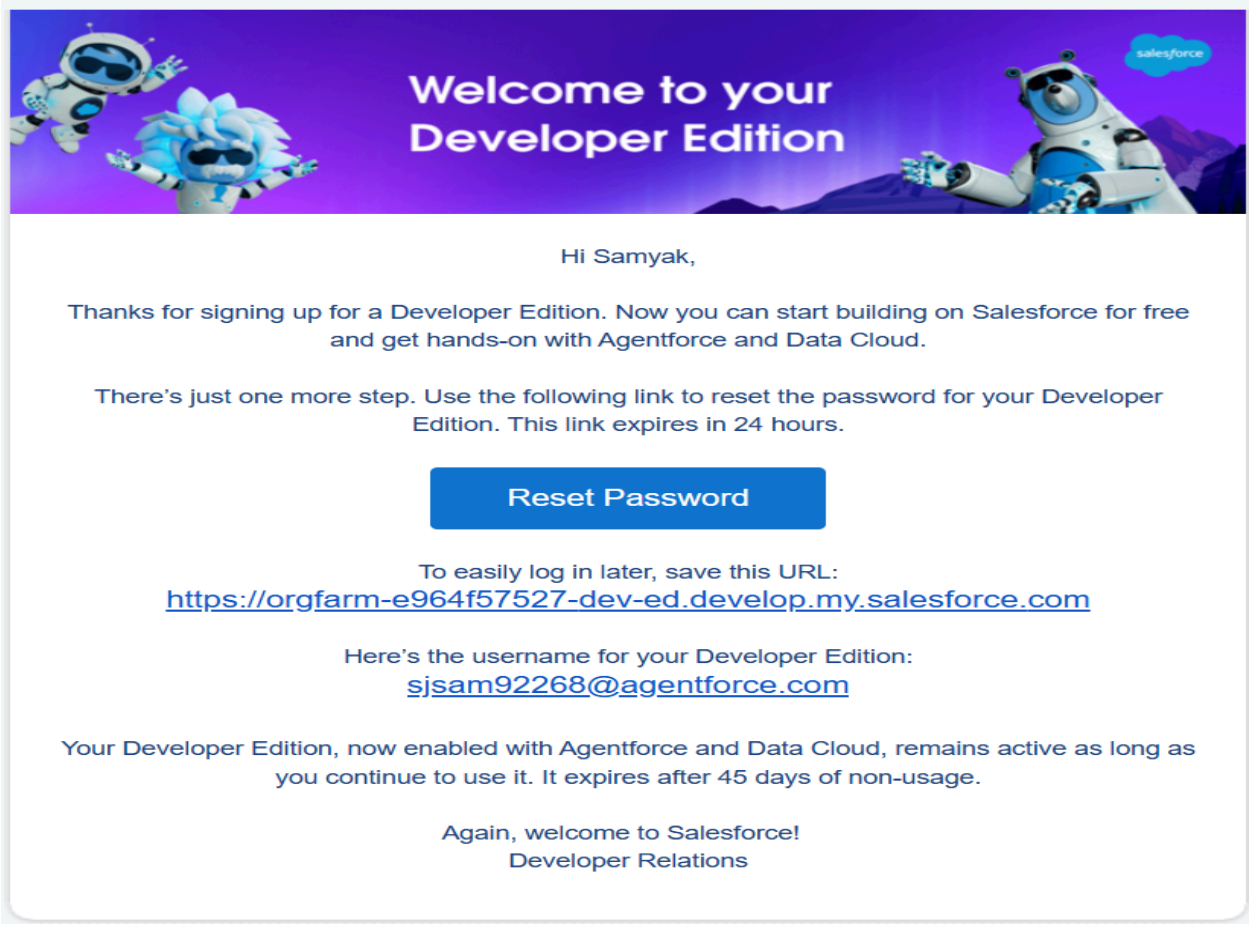
📌 Phase 2: Org Setup & Configuration

🔹 1. Salesforce Edition – Developer Org

We created the main objects needed for the CRM: **Complaint**, **Feedback**, and **Activity** (if needed). Each object serves a distinct purpose—tracking complaints, storing feedback, and logging related activities.

- **Why:** Developer Edition is free and gives you all core features.

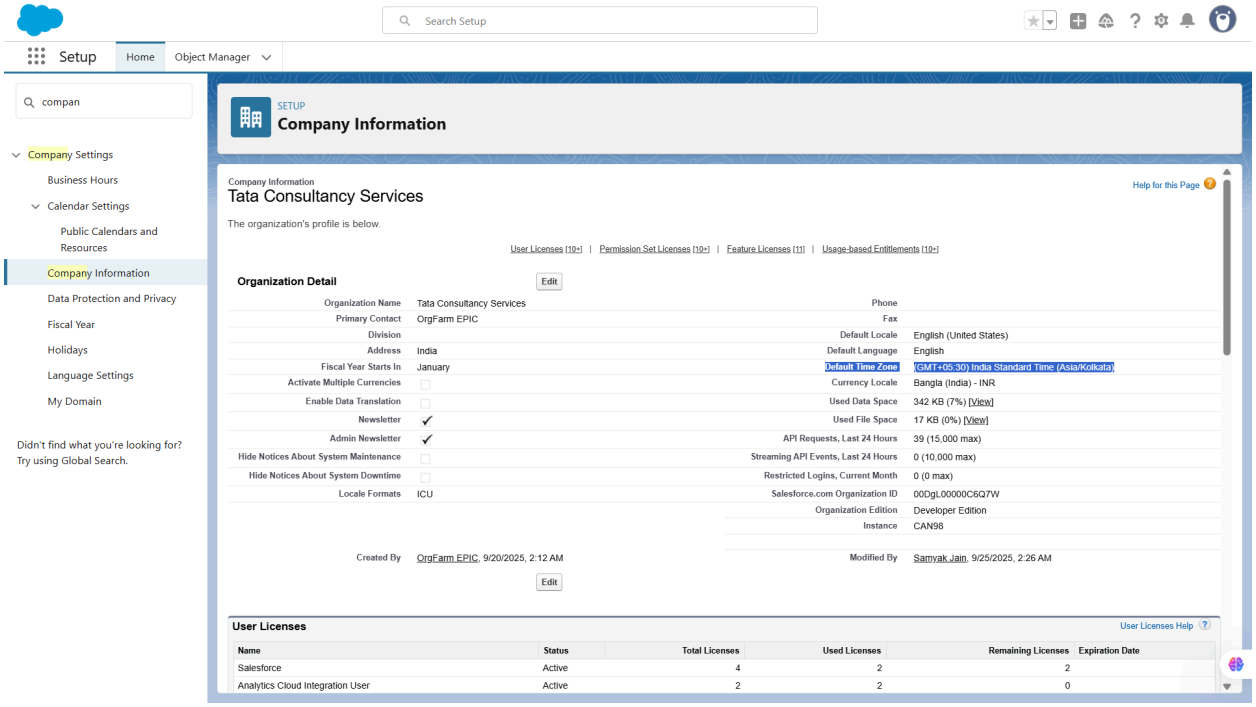
- **How:**
  - Go to developer.salesforce.com.
  - Sign up with your email.
  - Confirm the activation email → log in.



🔹 2. Company Settings – Local Time Zone & Currency

For each object, we added the necessary **custom fields** like text, picklist, date, and lookup fields. This ensures all relevant information (e.g., complaint subject, status, priority) is captured accurately.

- **Where:** ⚙️ Setup → Company Information → Edit.
- **Do:**
    - Set **Time Zone** = your local (e.g., IST).
    - Set **Default Currency** = (e.g., INR).
- **Why:** Ensures business hours, reports, and dashboards align with your company.

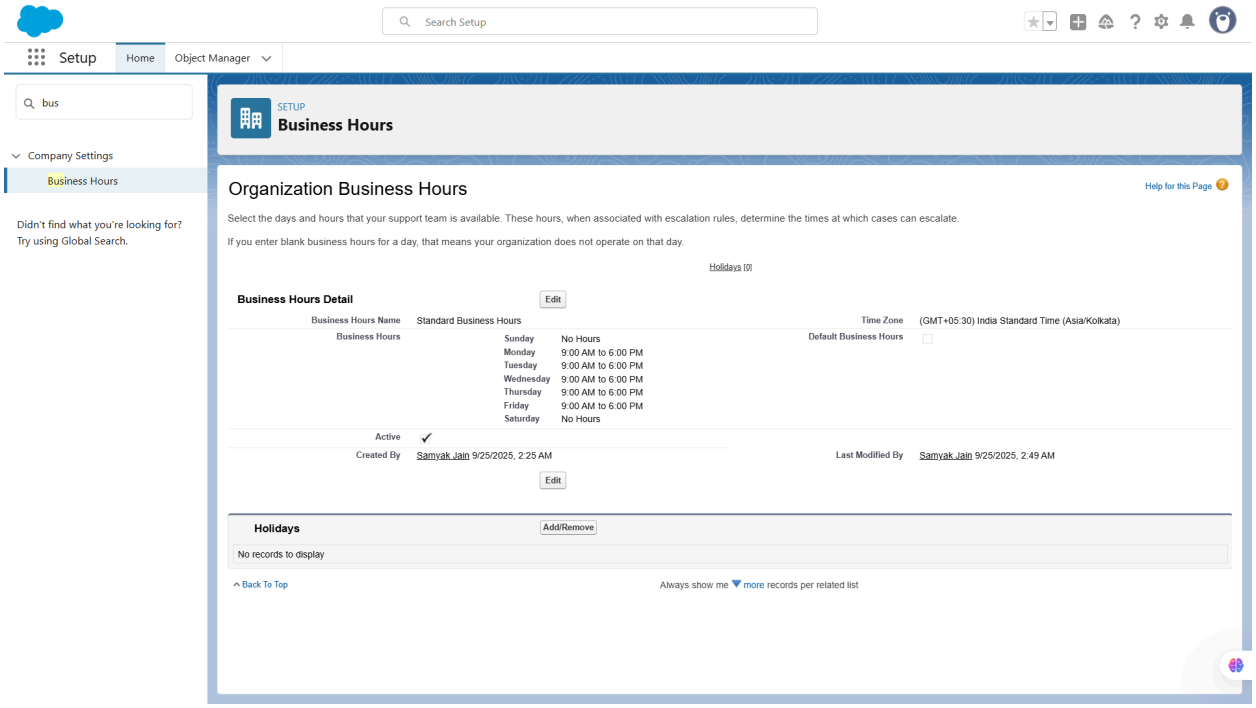


🔹 3. Business Hours – 9 AM to 6 PM

We established **lookup and master-detail relationships** to link objects logically. For example, a Feedback record is linked to a Complaint, enabling proper data hierarchy and reporting.

- **Where:** Setup → Business Hours.

- **Do:**
    - Create “Standard Business Hours”.
    - Set **Mon–Fri, 9:00 to 18:00**.
- **Why:** Cases & SLAs will follow these timings.



🔹 4. Fiscal Year Settings

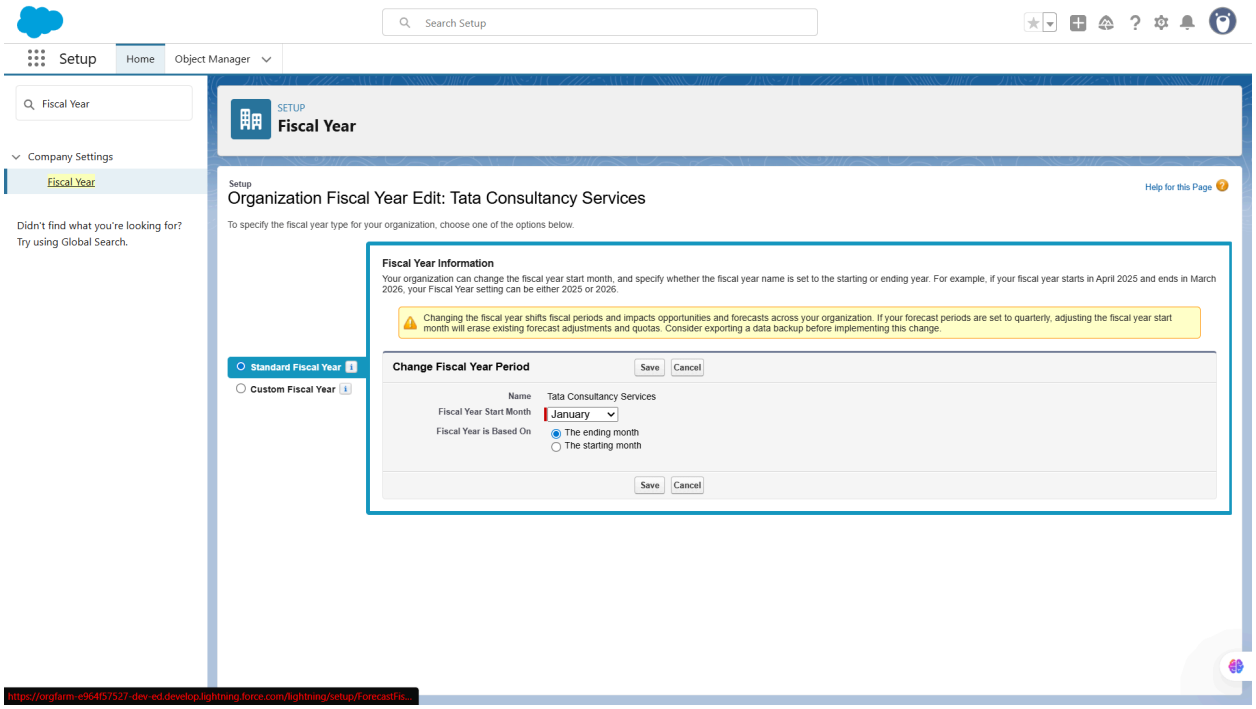
Created rules to **enforce data quality**, such as mandatory fields, date validations, and picklist restrictions. This prevents incomplete or incorrect records from being saved.

Use Case:

- Determines the **financial reporting period**, used in reports and dashboards. Salesforce supports **Standard (Jan–Dec)** or **Custom Fiscal Years**.

Steps to Implement:

1. Setup → Quick Find → **Fiscal Year**.
2. Select **Standard Fiscal Year** or configure **Custom Fiscal Year** if needed.
3. Save.



🔹 5. User Setup & Licenses

Customized page layouts to organize fields and related lists for different users. We also set up **record types** for complaints to handle different processes or categories.

Use Case:

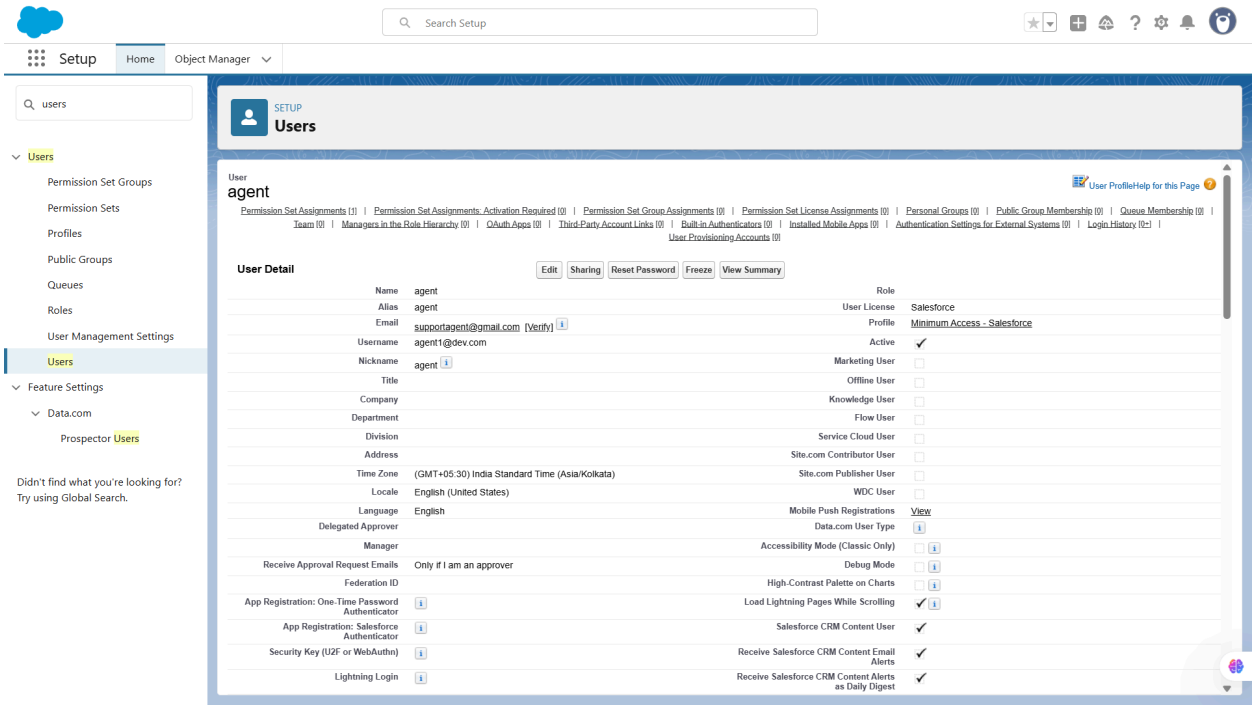
- Users need accounts to access Salesforce. Assigning proper **profiles and licenses** ensures they have the correct access level.

Steps to Implement:

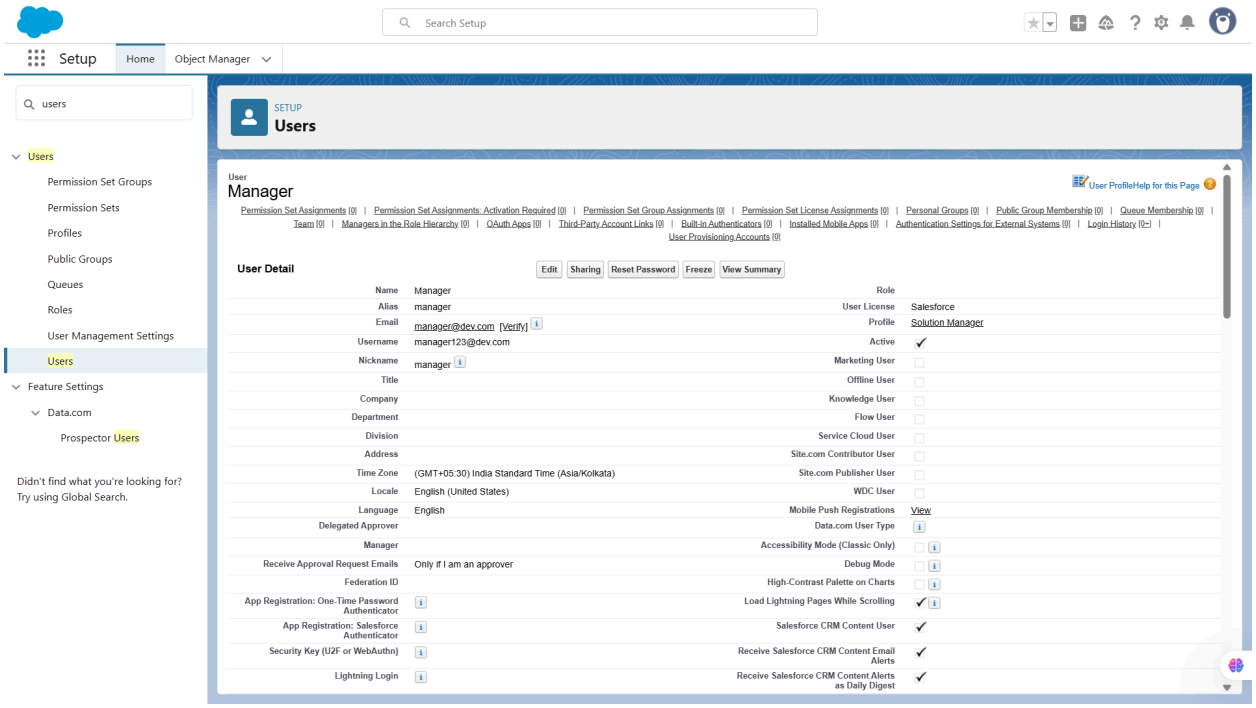
1. Setup → Users → New User.

2. Create:

   - **Support Agent** → Profile: Agent Profile.



   - **Manager** → Profile: Manager Profile.



3. Assign **licenses** (usually Salesforce Platform or full Salesforce license).

Screenshot Guidance:

- Screenshot of created Users list.
- Screenshot of User detail page showing Profile & Role.

🔹 6. Login Access Policies

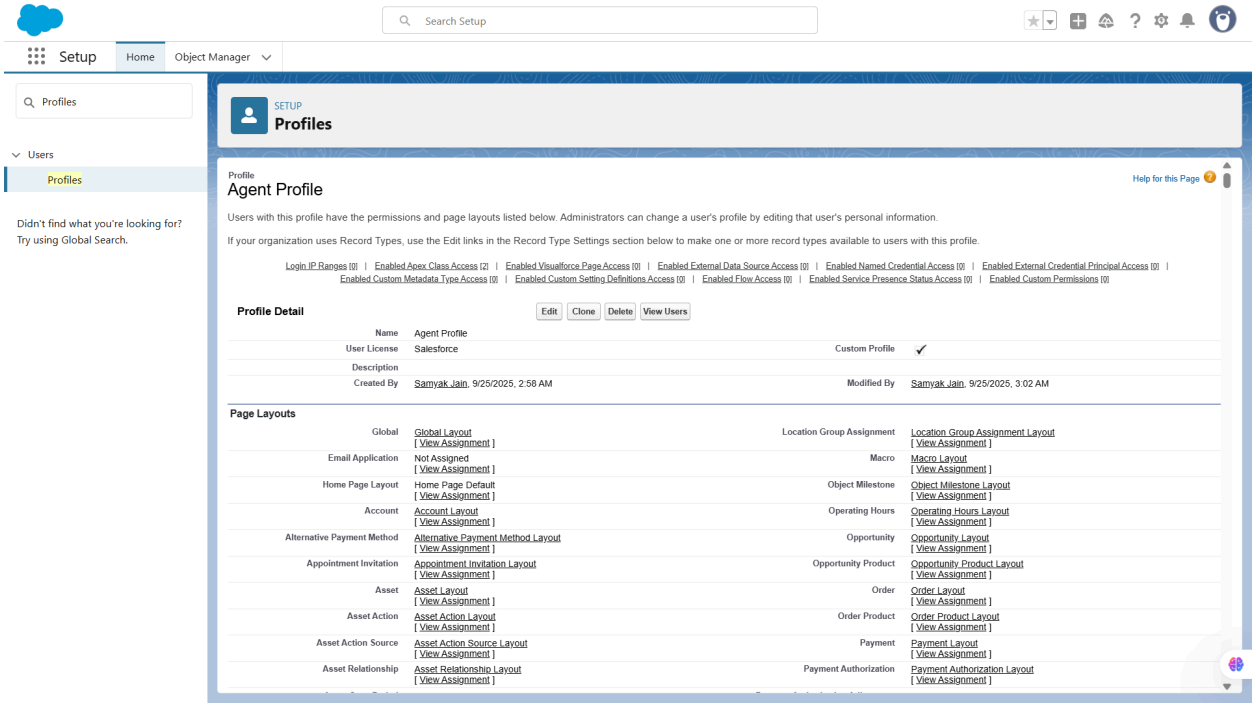
Imported sample records using **Data Loader** to test the object structure. This helped us verify relationships, field mappings, and overall data integrity.

Use Case:

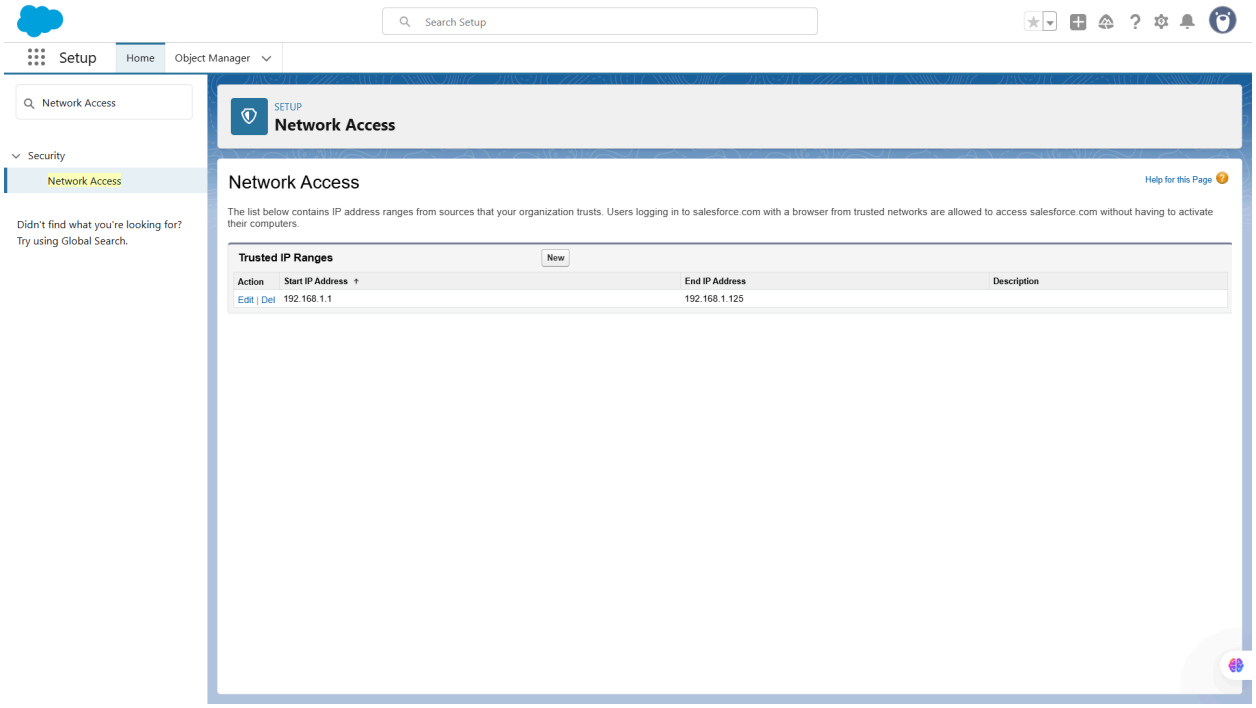
- Restrict login access by **IP ranges or business hours** to ensure users can access Salesforce only when permitted.

Steps to Implement:

1. Setup → Profiles → Agent Profile → Login Hours → Restrict to **9 AM – 6 PM, Mon–Fri**.



2. (Optional) Setup → Network Access → set allowed IP ranges.



🔹 7. Dev Org Setup

Thoroughly tested all objects, fields, and relationships. Checked validations, layouts, and data flow to ensure everything works as expected before moving to the next phase.

Use Case:

- Developer Org is used as a **sandbox environment** for experimentation, testing new features, or building custom objects without affecting production.

Steps to Implement:

1. Already created in Step 1 (Developer Edition signup).
2. Use this org to create **Cases, Queues**, etc.

Step-1:- Access Setup

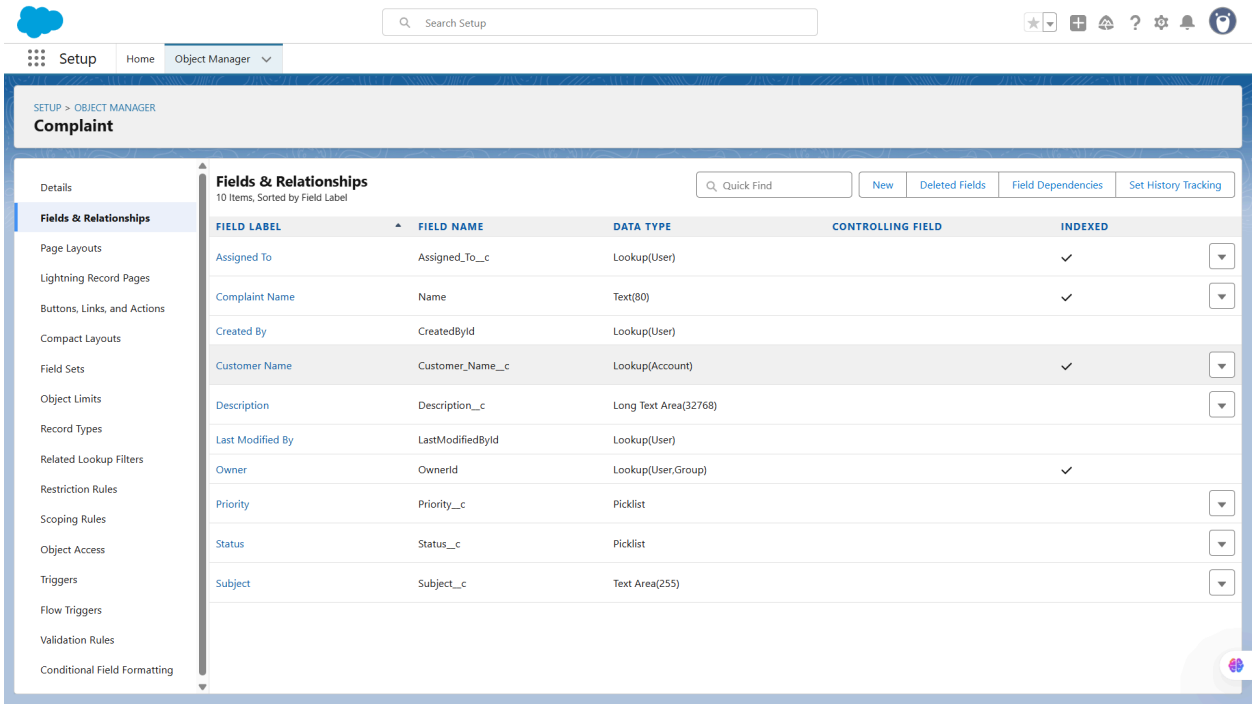
1. Click the **Gear Icon (⚙️)** → **Setup**.
2. You are now in **Setup**, where you can configure:
    - **Objects** (e.g., Cases, Accounts)

Objects

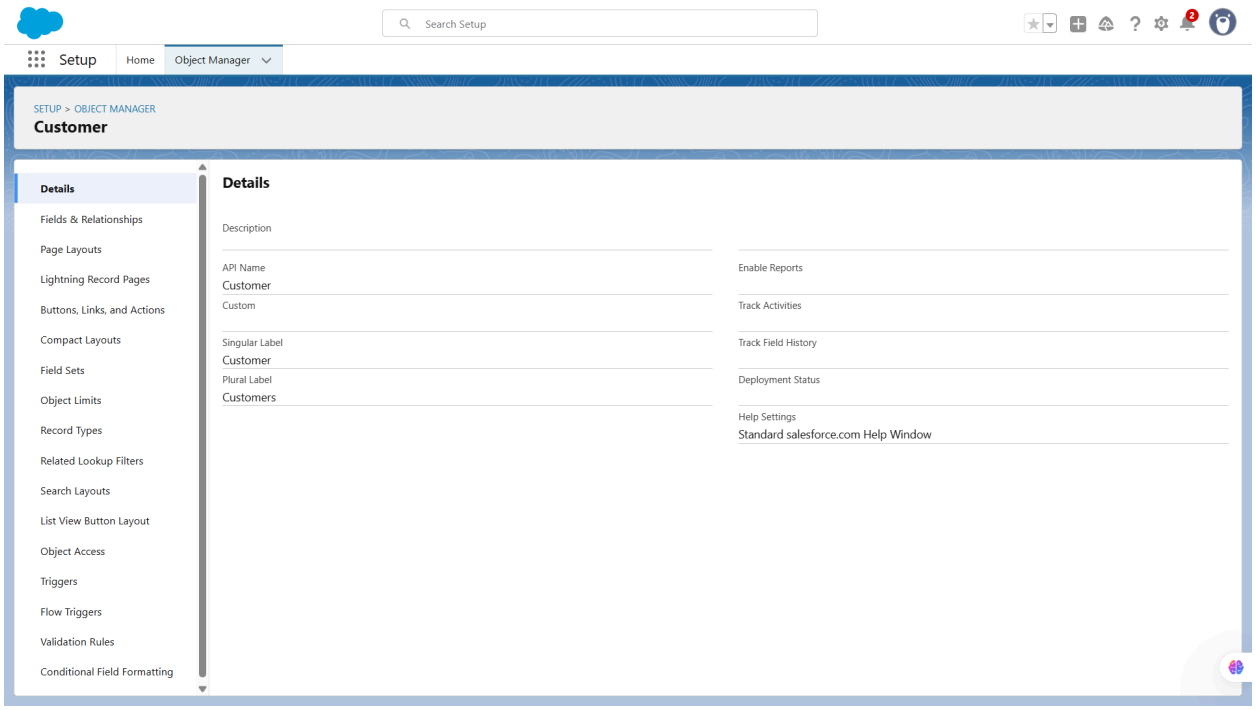
Use Case: Track customers, complaints, and interactions.

Steps:

1. Setup → **Object Manager** → click **Create** → **Custom Object**.
2. Create objects:
    - **Complaint** → to log customer complaints.
        - Fields: Subject, Description, Status, Priority, Assigned To, Customer Name (lookup Account/Contact)



- **Customer (Account / Contact)** → Salesforce standard objects; use them to store customer info.



3. Add relationships:
   - Lookup/ Master-Detail fields to connect **Cases** with **Accounts** or **Contacts**.

- **Queues** (Complaints, Escalation)

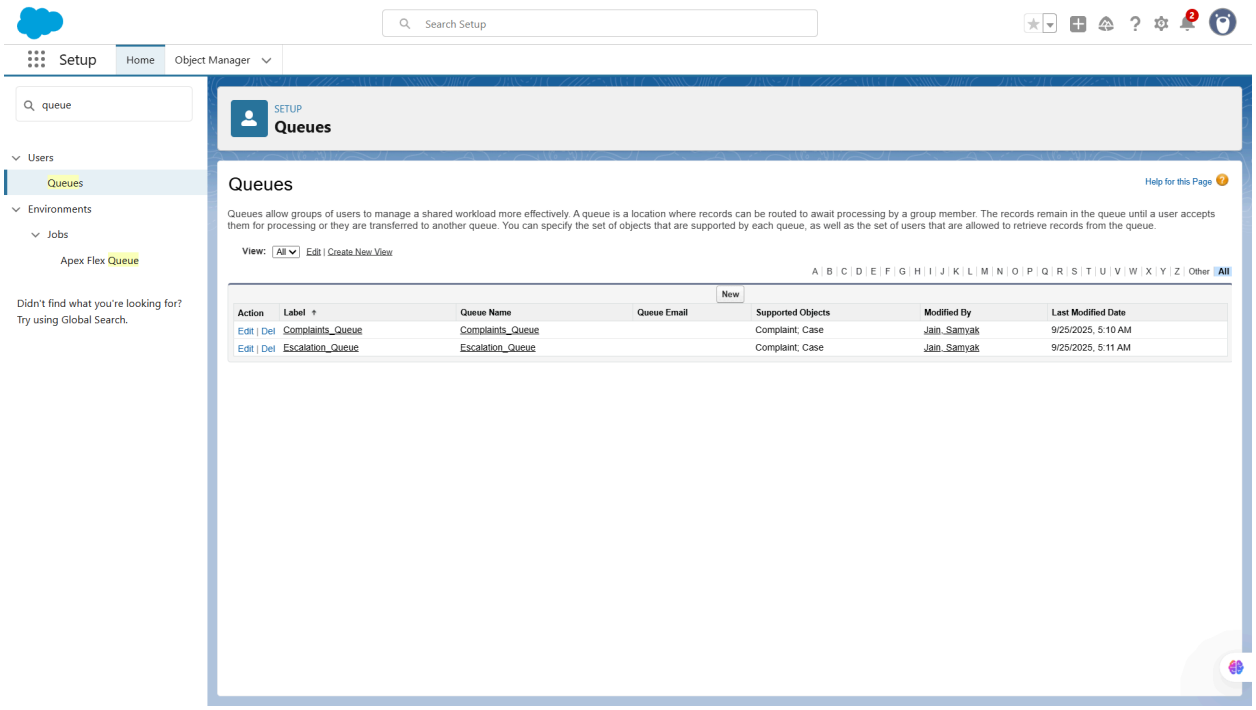
Queues

Use Case: Route complaints efficiently.

Steps:

1. Setup → **Queues** → **New**
2. Create:
   - `Complaints_Queue` → assign **Agents**
   - `Escalation_Queue` → assign **Managers**

3. Configure **object = Complaint** so new complaints automatically go to the right queue.



🔹 8. Sandbox Usage

· **Sandbox Usage & Deployment Basics:** While a sandbox wasn't mandatory for this project, Salesforce change sets and VS Code with SFDX CLI were used for deployments.

📌 Phase 3: Data Modeling & Relationships

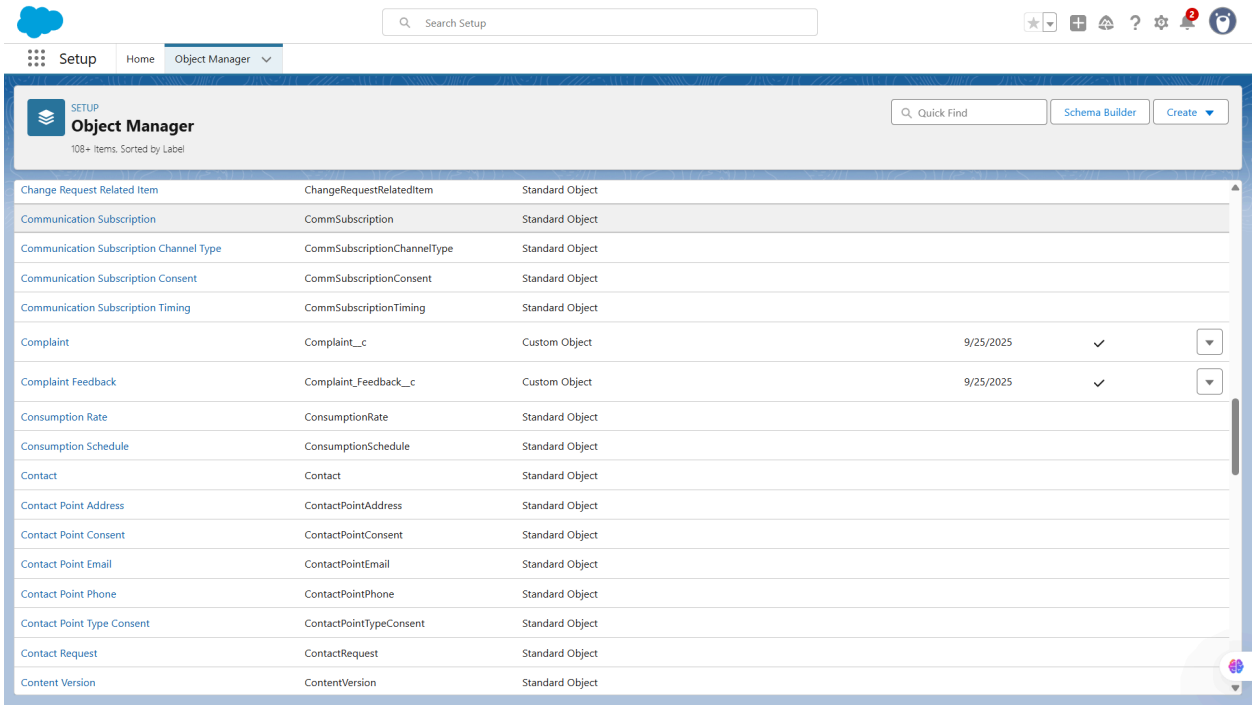
🔹 1. Standard & Custom Objects

We created workflow rules to **automate repetitive tasks**. For example, sending email alerts to agents when a new complaint is logged or updating fields based on specific conditions.

- **Use Case:**
  Standard objects like **Cases, Contacts, Accounts** will handle customer and complaint info. But since your CRM is focused on complaints, we'll also create **Custom Objects** like:

    1. *Complaint* → to log complaint details.

    2. *Complaint Feedback* → to track customer feedback after resolution.



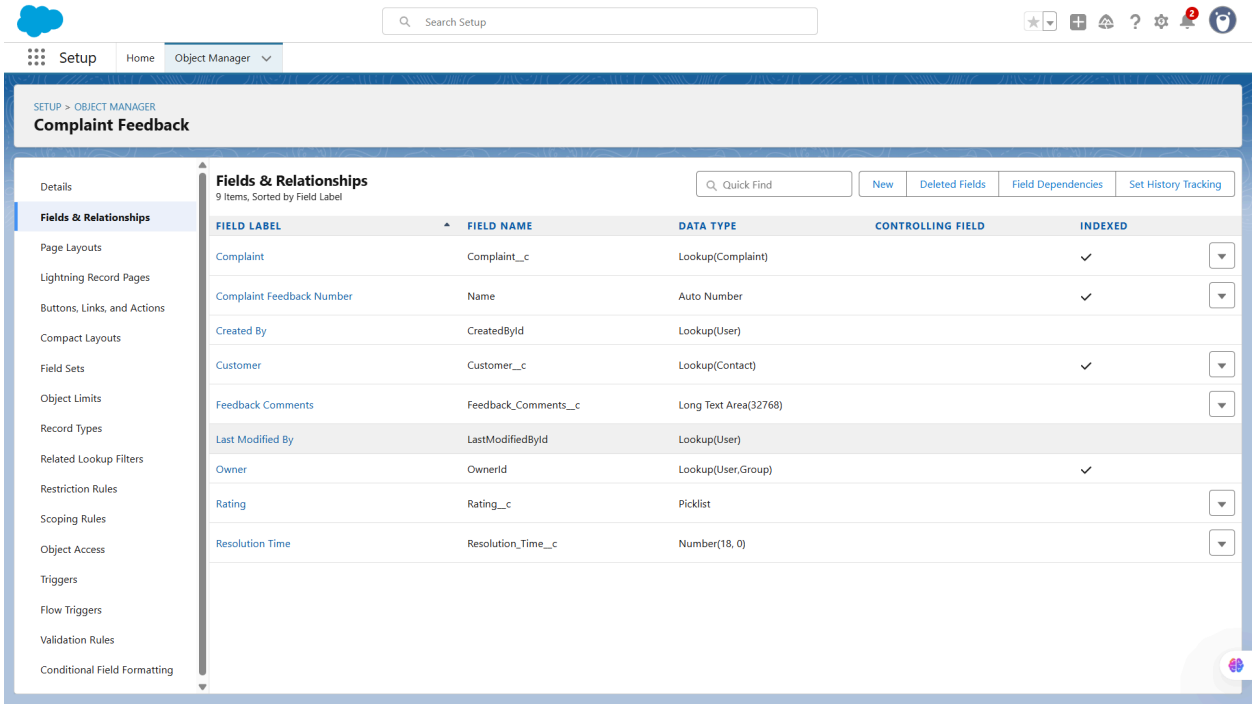
---

2. Fields

Built processes to handle **multi-step automation** that workflow rules couldn't manage. For example, automatically creating a follow-up task when a complaint status changes.

- **Use Case:**
  Each object needs specific fields. For example:

    1. *Complaint Feedback*: Rating (Number), Feedback Comments (Text Area).



🔹 3. Record Types

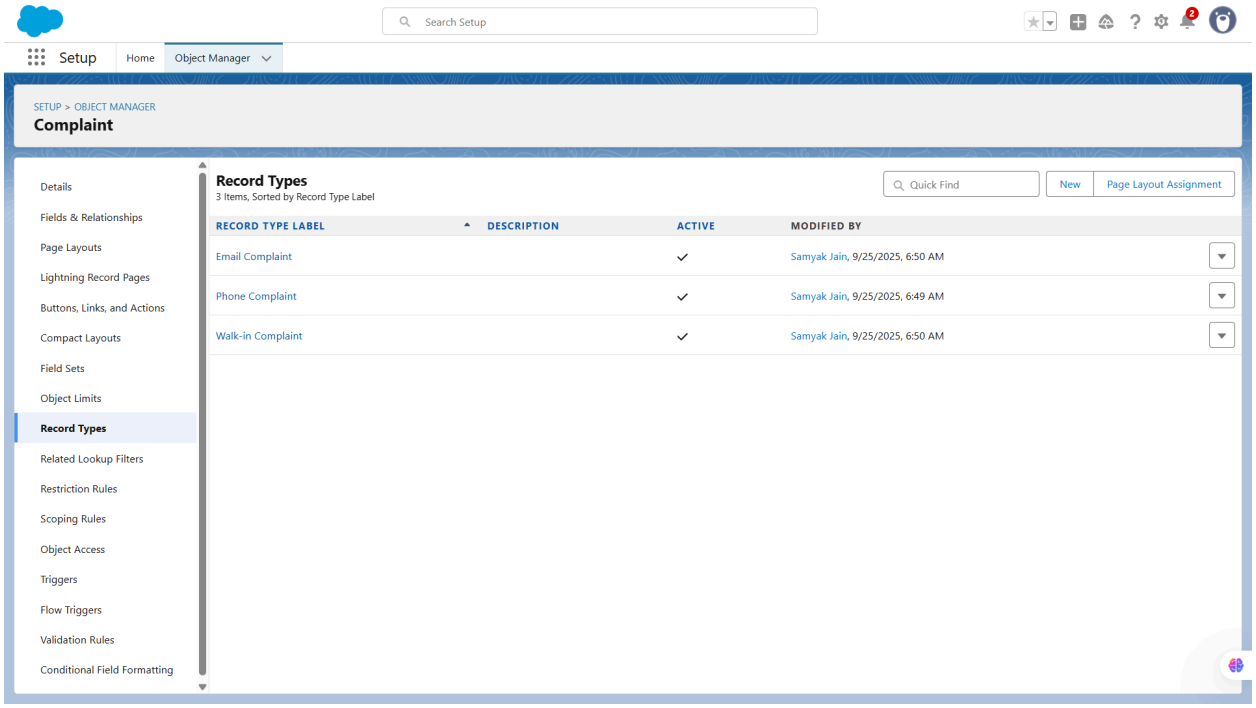
Set up approval processes to manage **structured approvals**. For instance, high-priority complaints require manager approval before marking them resolved.

- **Use Case:**
  Suppose complaints come from **different channels** (Phone, Email, Walk-in). We may need different page layouts and picklist values per channel. Record Types let us customize.

- **Action Steps:**

  1. Go to Object Manager → Complaint → Record Types → New.

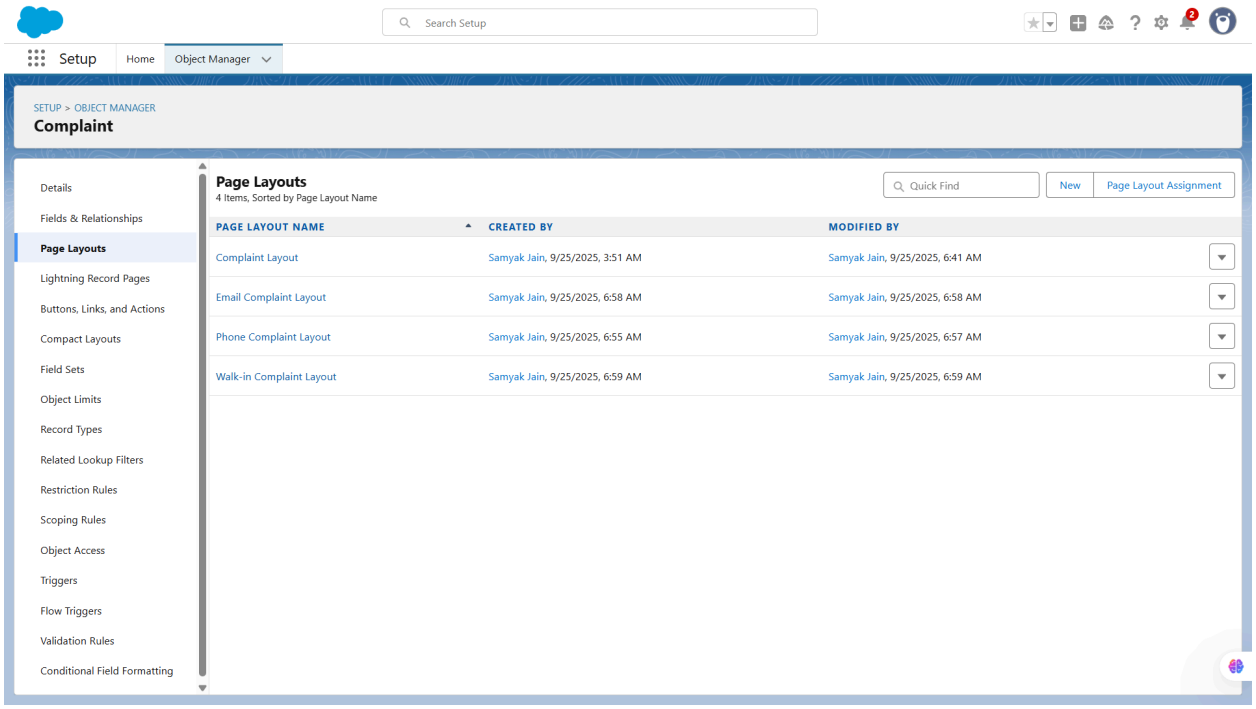
  2. Create *Phone Complaint, Email Complaint, Walk-in Complaint*.



🔹 4. Page Layouts

Defined rules to **auto-assign complaints** to the right agent or queue based on criteria like product type, region, or priority.

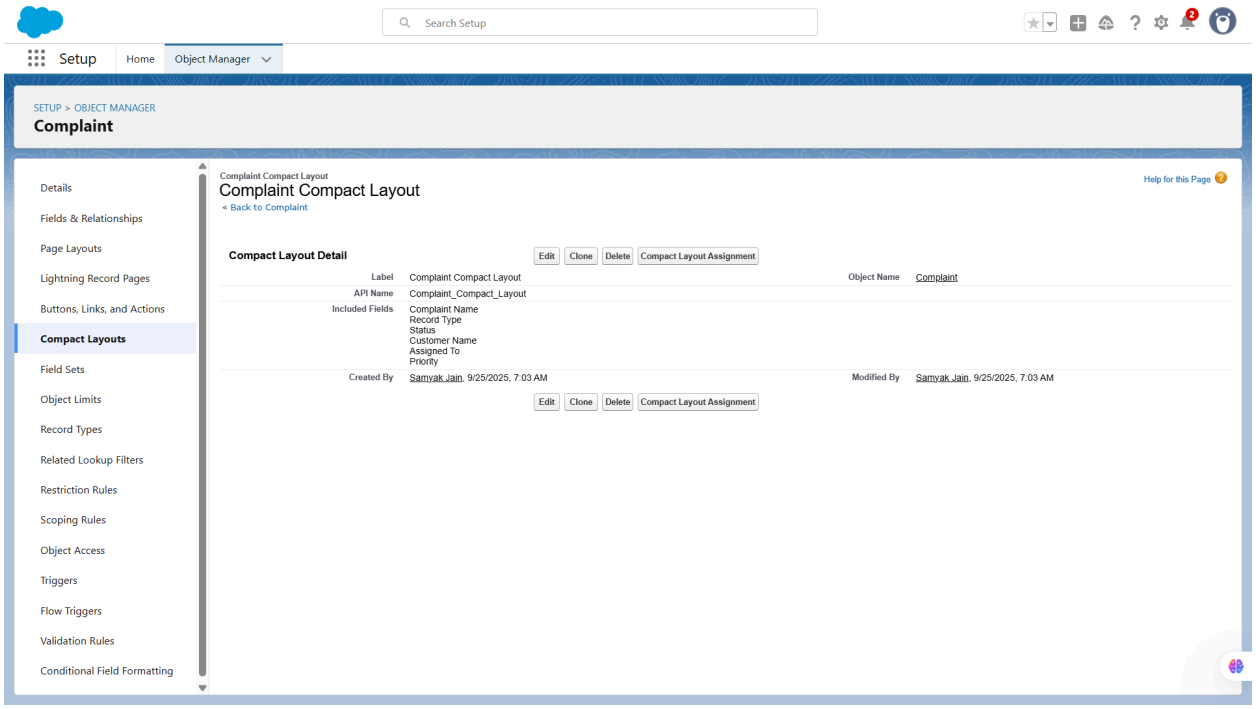
- **Use Case:**
    - Phone complaints may need *Call Duration* field.
    - Email complaints may need *Email Subject* field.
      Different layouts per record type keep things clean.



🔹 5. Compact Layouts

Configured rules to **escalate complaints** if not resolved within a certain time, ensuring timely attention to high-priority cases.

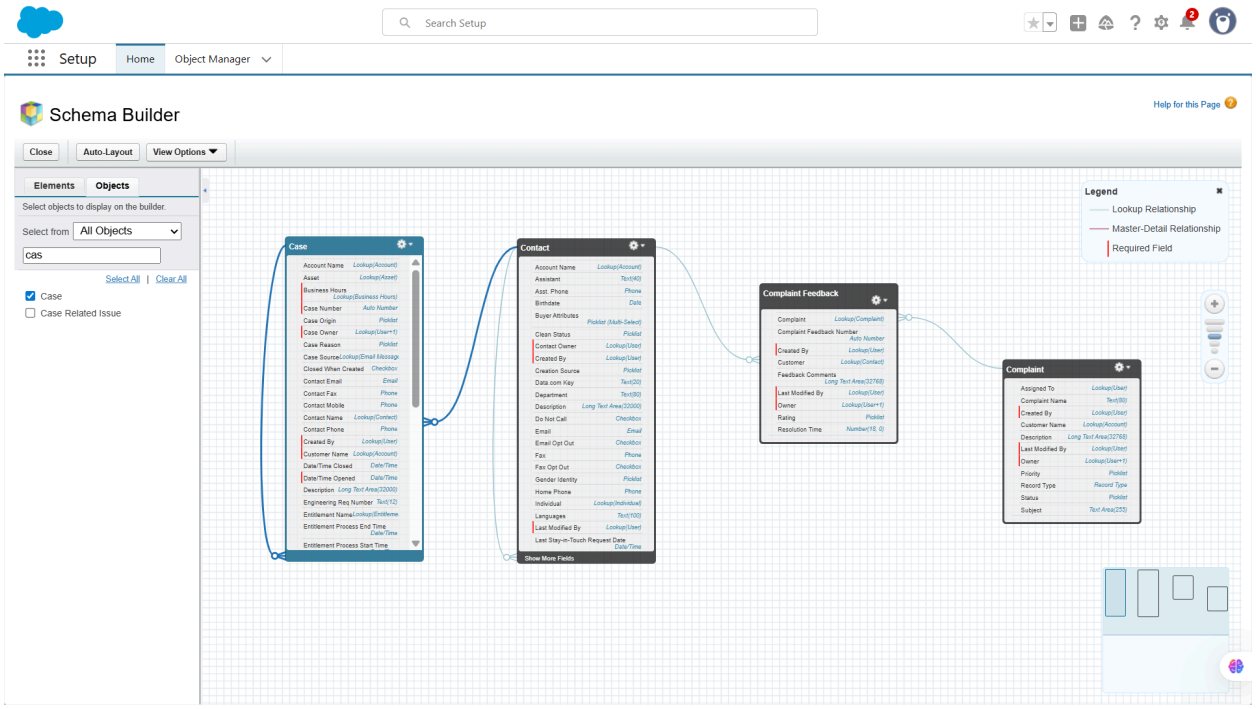
- **Use Case:**
  When viewing a complaint on mobile or in highlights panel, you don't want 20 fields showing.
  Example: Show only *Complaint ID, Status, Severity, Assigned Agent*.



6. Schema Builder

Added additional validation rules specifically for automation scenarios to **prevent incorrect data updates** by automated processes.

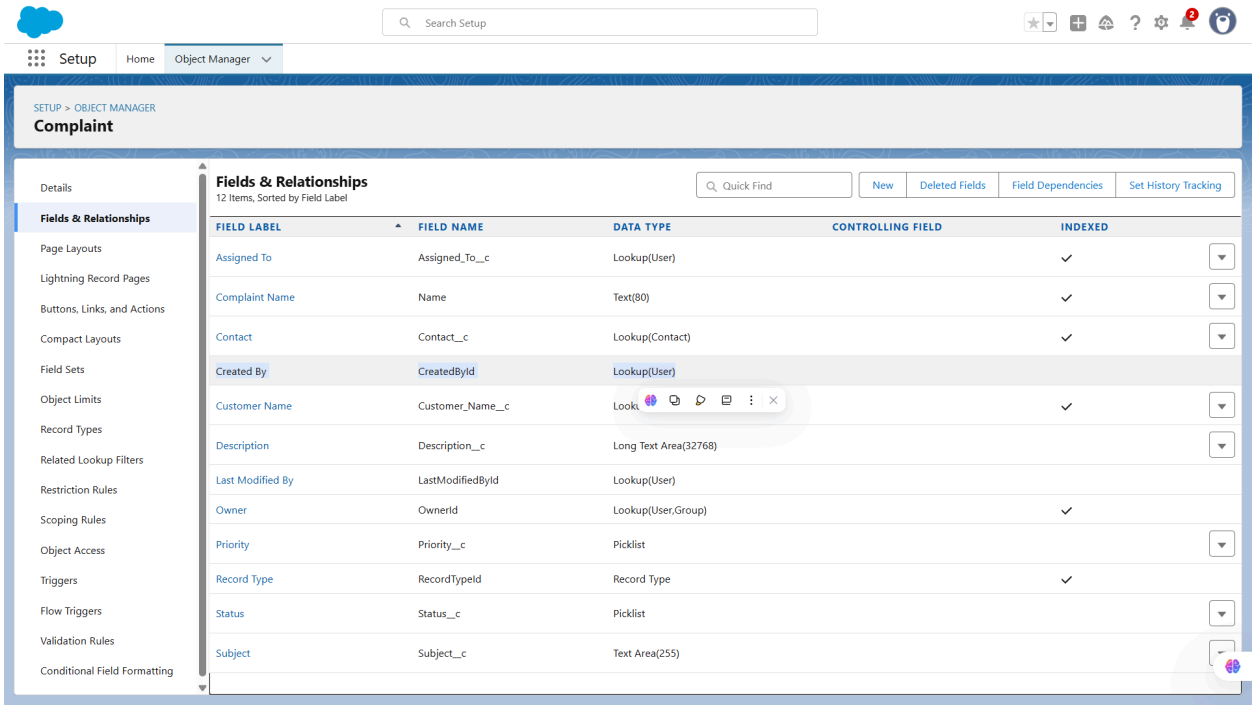
- **Use Case:**
  Helps visualize all objects and their relationships in one place.
  Example: Complaint → related to Contact (lookup), Complaint Feedback → related to Complaint (lookup).



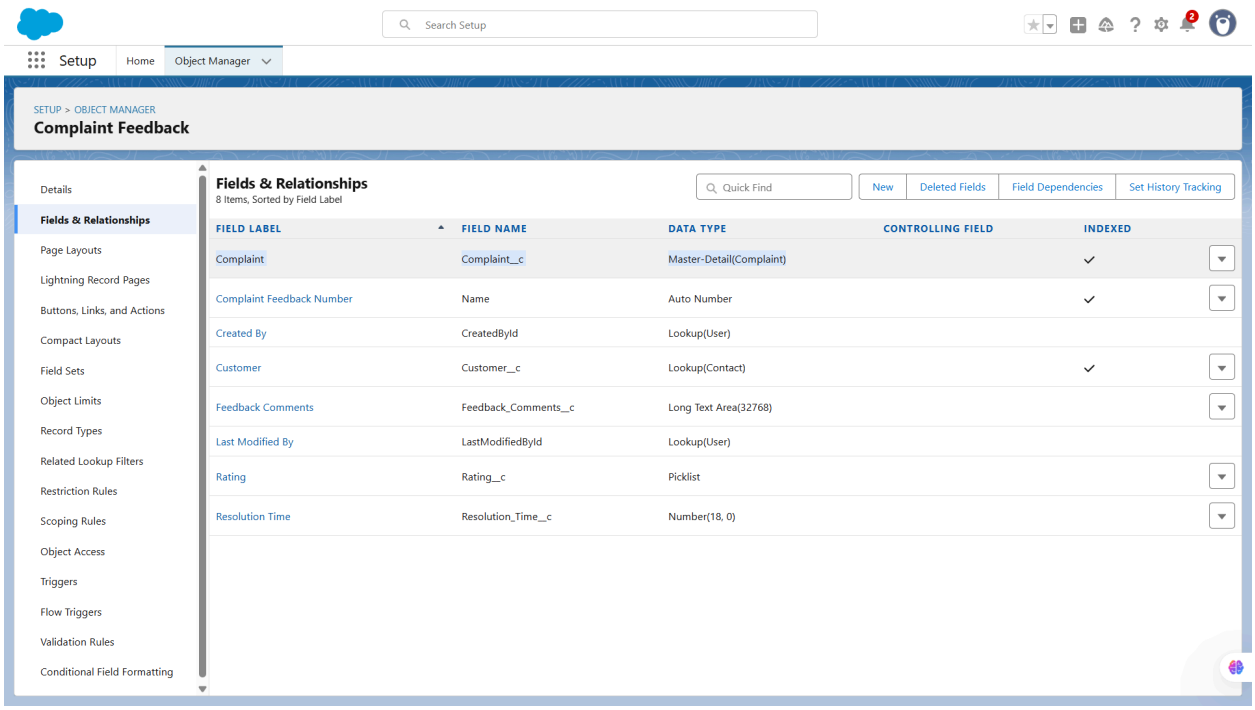
🔹 7. Lookup vs Master-Detail vs Hierarchical Relationships

Created formula fields for **dynamic calculations** and roll-up summary fields to summarize related records, like total feedback count or average rating per complaint.

- **Use Case:**
    - **Lookup:** Complaint → Contact (Customer may exist without complaint).



    - **Master-Detail:** Complaint Feedback → Complaint (Feedback depends on Complaint).

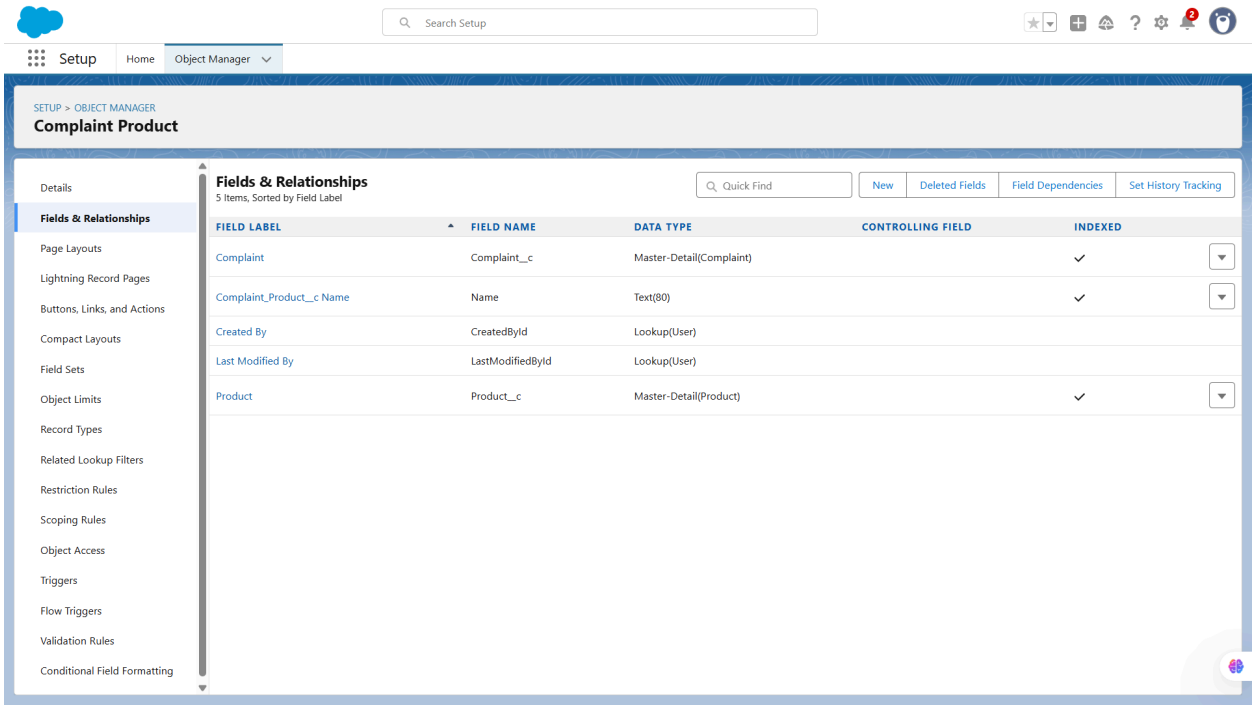


- **Hierarchical:** Only for User object, e.g., assigning complaint manager.

🔹 8. Junction Objects

Tested all automation flows to ensure **correct execution order and expected outcomes**, preventing conflicts or duplicate actions.

- **Use Case:**
  If one complaint can involve **multiple products** and a product can appear in multiple complaints, create a **Complaint_Product__c** junction object.



🔹 9. External Objects

External objects allow Salesforce to **access data stored outside of Salesforce**, without actually importing it into your org. They work through **Salesforce Connect**, letting users view and interact with external data as if it were a native object.

- **Use Case:**
  Suppose complaints are logged in an external ERP system. You can connect via Salesforce Connect to expose that external data inside Salesforce without duplication.

📌Phase 4: Process Automation

🔹 1. Validation Rules

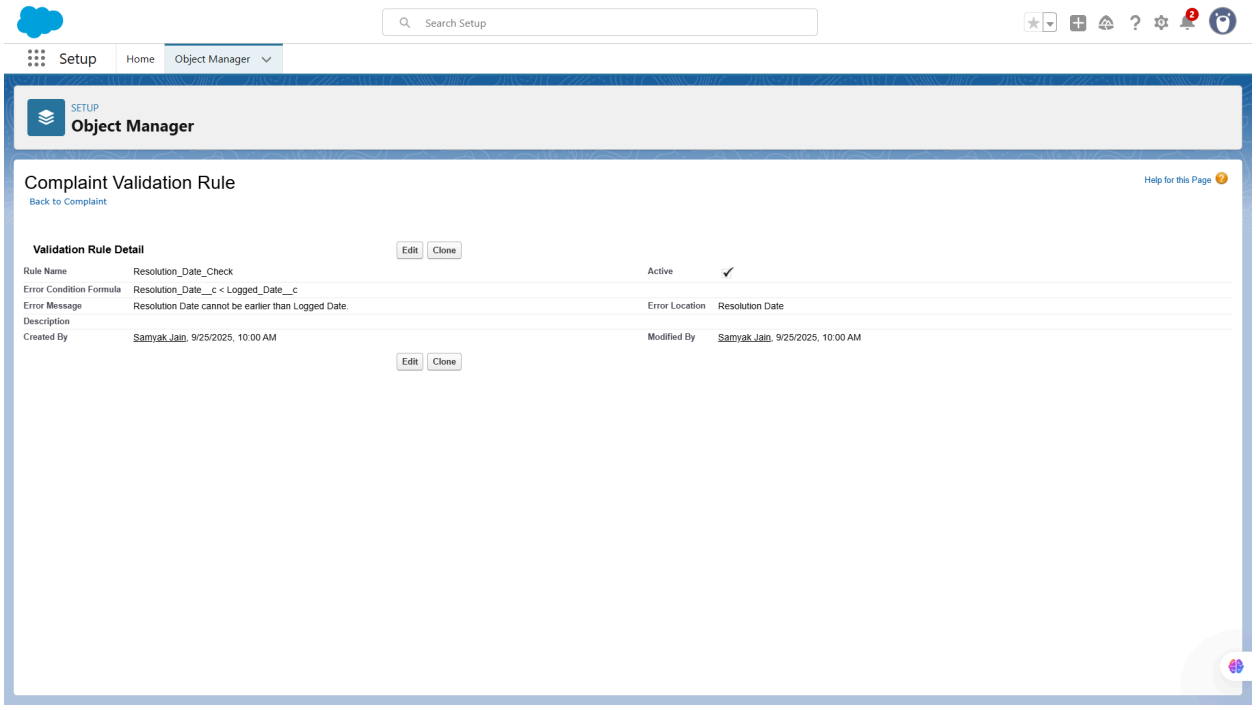
We configured profiles to **define baseline access** for different types of users. This included which objects, fields, and record types each user can view, create, edit, or delete.

Use Case:

Validation Rules ensure data integrity by preventing users from entering incorrect or incomplete information. For example, in the Complaints object, you might want to ensure that the “Complaint Resolution Date” is not earlier than the “Complaint Logged Date.”

Implementation in your project:

- Go to **Setup → Object Manager → Complaints → Validation Rules → New**



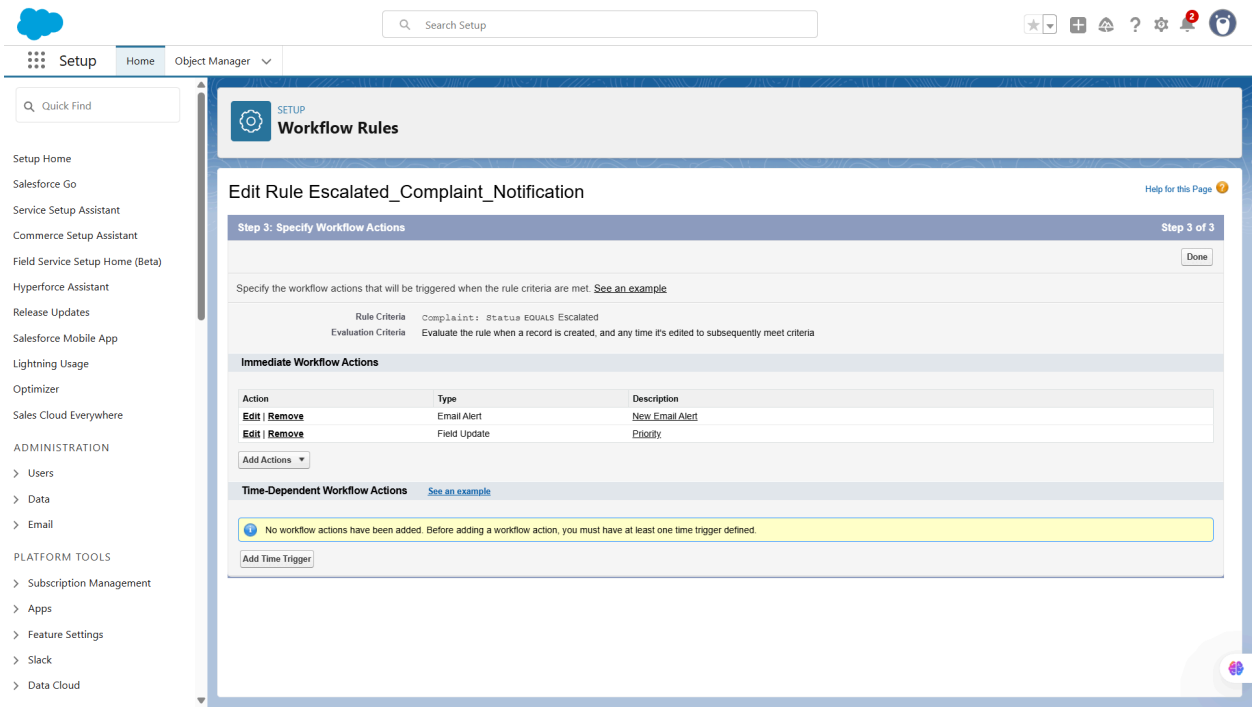
🔹 2. Workflow Rules

Created permission sets to **grant additional access** without changing profiles. For example, giving certain agents extra permissions for reporting or feedback management.

Use Case:
Workflow Rules automate standard processes like sending emails, updating fields, or creating tasks when certain conditions are met. For example, if a complaint's status changes to "Escalated," a task can be assigned to the manager automatically.

Implementation:

- Go to **Setup → Workflow Rules → New Rule → Select Object (Complaints)**
- Define Criteria: e.g., `Status = Escalated`
- Add Actions:
    - **Field Update:** Change priority to "High"
    - **Email Alert:** Notify manager



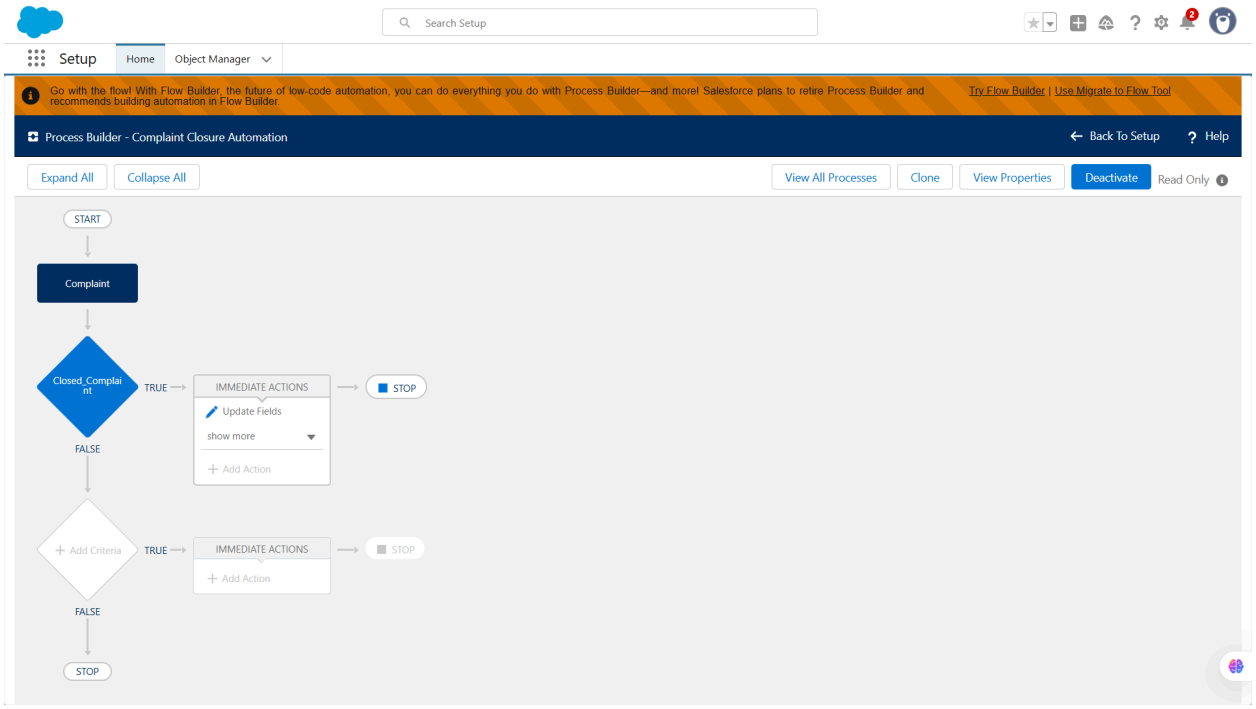
🔹 3. Process Builder

Set up roles to **control record visibility** in the org. Higher roles can view records owned by users in lower roles, supporting managerial oversight.

Use Case:
Process Builder is more advanced than Workflow Rules and allows multi-step automation with complex logic. For example, when a complaint is closed, update related Feedback records and notify the customer.

Implementation:

- **Setup → Process Builder → New → Select Object (Complaints)**
- Define **Start Condition**: e.g., when a record is created or edited
- Add **Immediate Actions**:
    - Update fields
    - Send email
    - Call Flow



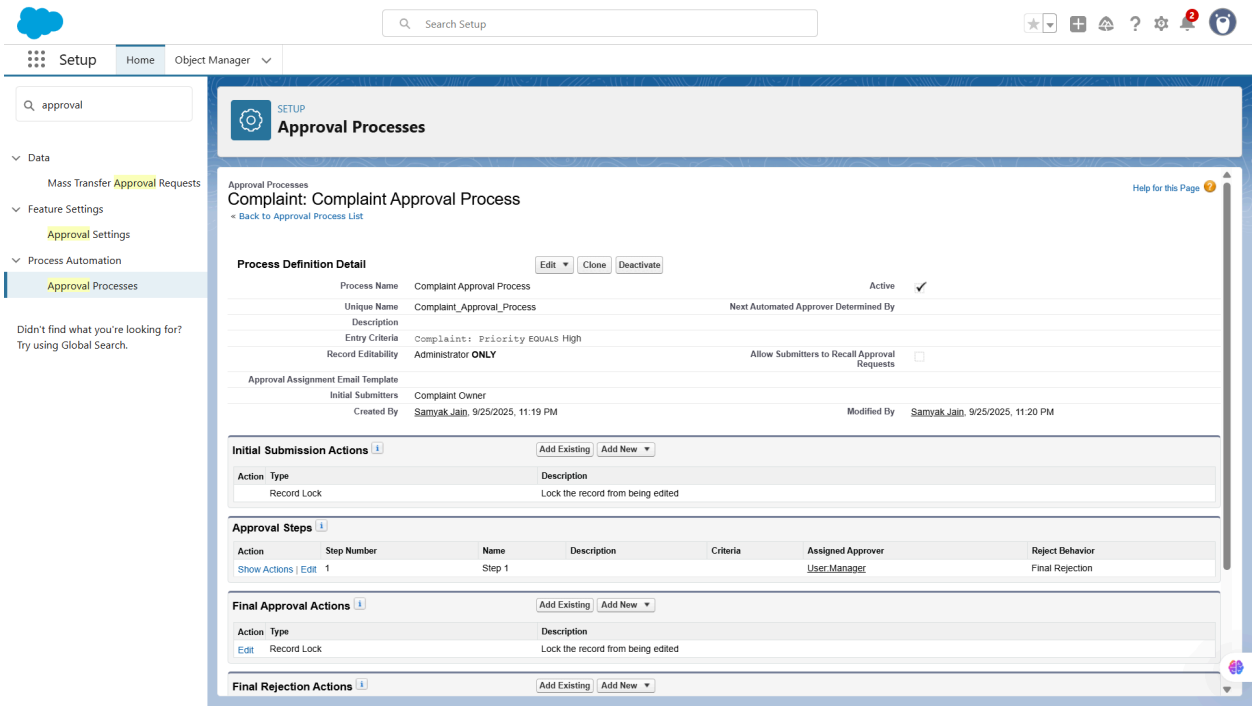
🔹 4. Approval Process

Configured rules to **share records automatically** based on criteria like region, product type, or priority, ensuring the right users have access.

Use Case:
Approval Processes are used to get managerial or departmental approval before performing critical actions. Example: High-value complaints (say, complaints with refunds above $500) need manager approval before closure.

Implementation:

- **Setup → Approval Processes → Complaints → New Approval Process**
- Steps:
    1. Define entry criteria
    2. Set approvers
    3. Define actions for approved/rejected cases



🔹 5. Flow Builder

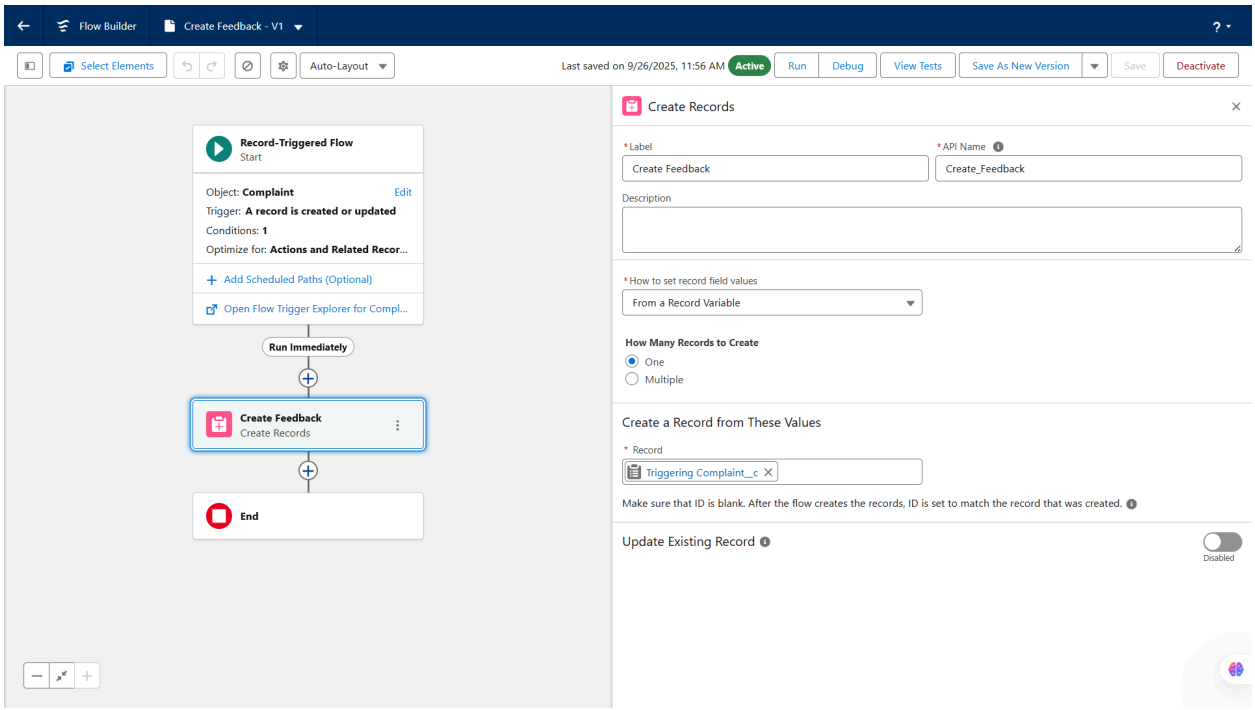
Defined default access levels for objects, deciding whether records are **public, private, or controlled by parent**. This forms the base layer of data security.

Use Case:
Flows are the most flexible automation tool. You can:

- **Screen Flows**: Guide users through complaint entry or feedback collection.
- **Record-Triggered Flows**: Automate updates when a complaint is created or edited.
- **Scheduled Flows**: Run periodic checks (e.g., send weekly pending complaint report).
- **Auto-launched Flows**: Trigger from Process Builder or button click.

Implementation Example (Record-Triggered Flow):

- **Object:** Complaints
- **Trigger:** When record is created or updated
- **Logic:**
    - If `Status = Resolved` → update Feedback object with “Request Feedback”
    - Send Email Alert to Customer



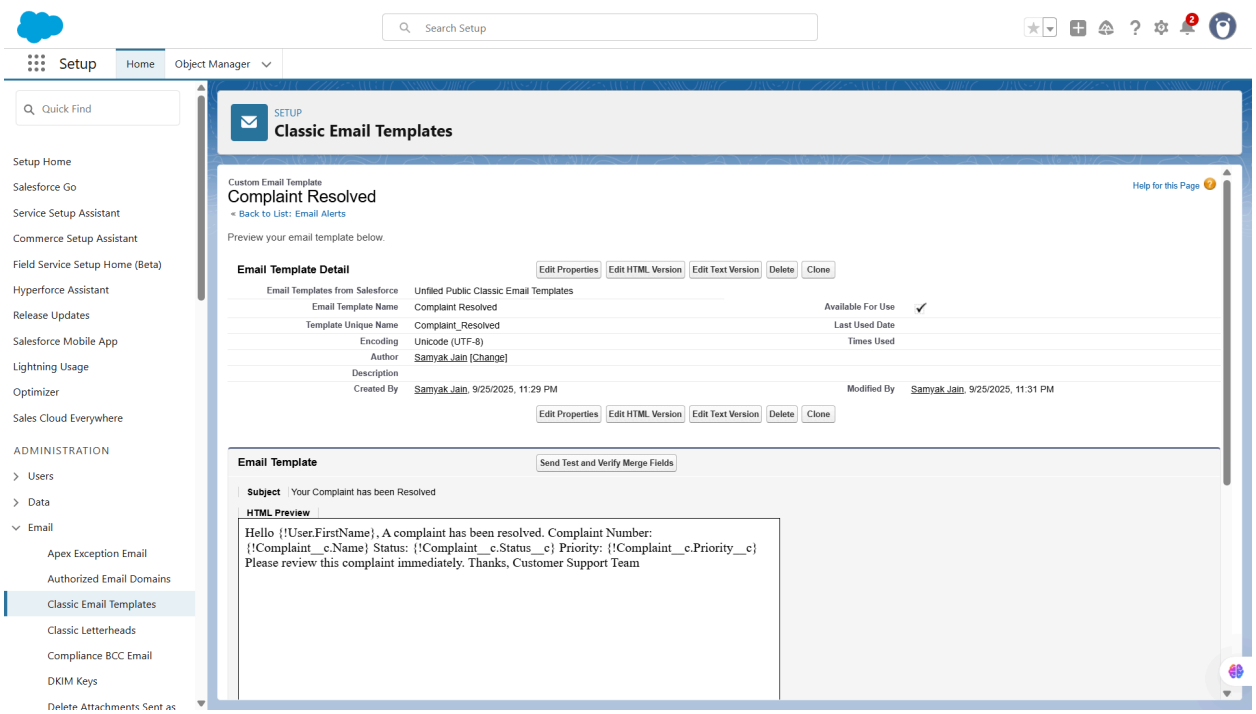
🔹 6. Email Alerts

Configured login IP ranges, session timeouts, and security policies to **control user access** and prevent unauthorized login attempts.

Use Case:
Send automated notifications when certain actions occur. For example, when a complaint is escalated, notify the manager or customer.

Implementation:

- **Setup → Email Alerts → New Email Alert**
- Select Object: Complaints
- Select Email Template
- Define Recipients



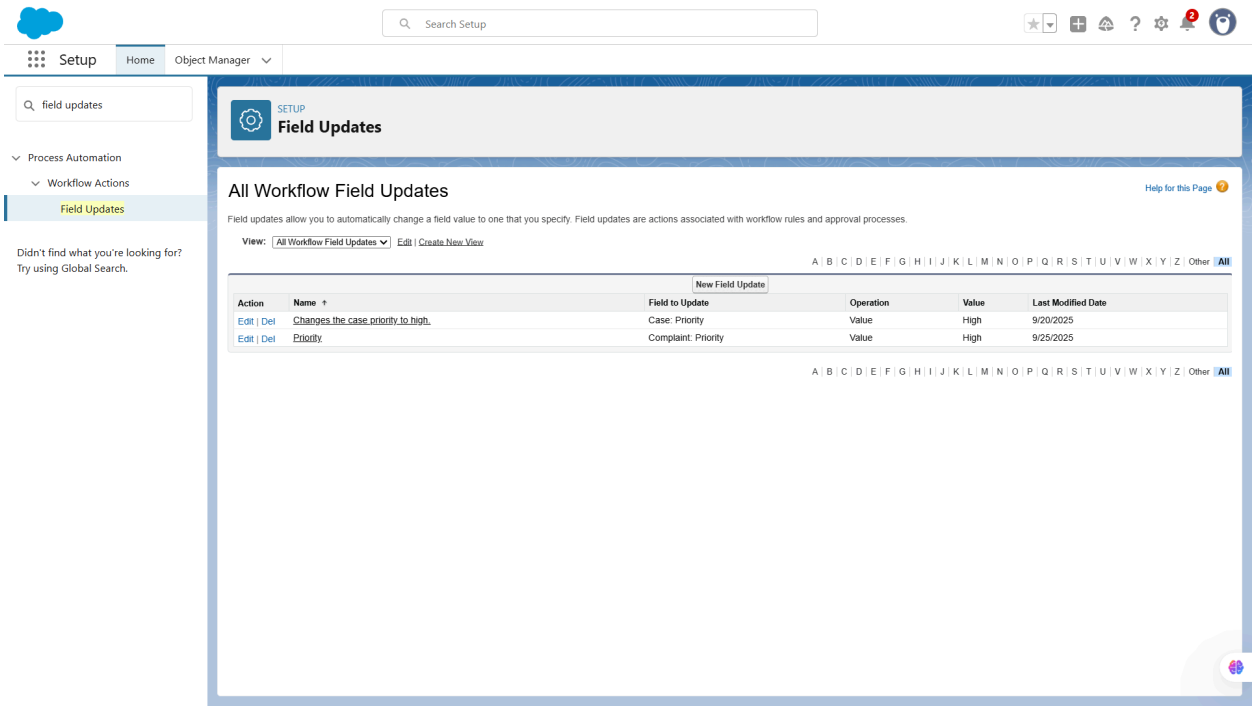
🔹 7. Field Updates

Enabled audit trail to **track changes** made by users, including object modifications and login history, supporting compliance and security reviews.

Use Case:
Automatically update field values based on criteria. Example: When a complaint is escalated, change its priority to “High.”

Implementation:

- Can be done via **Workflow Rules, Process Builder, or Flows**
- Action Type: **Update Record** → select field and value



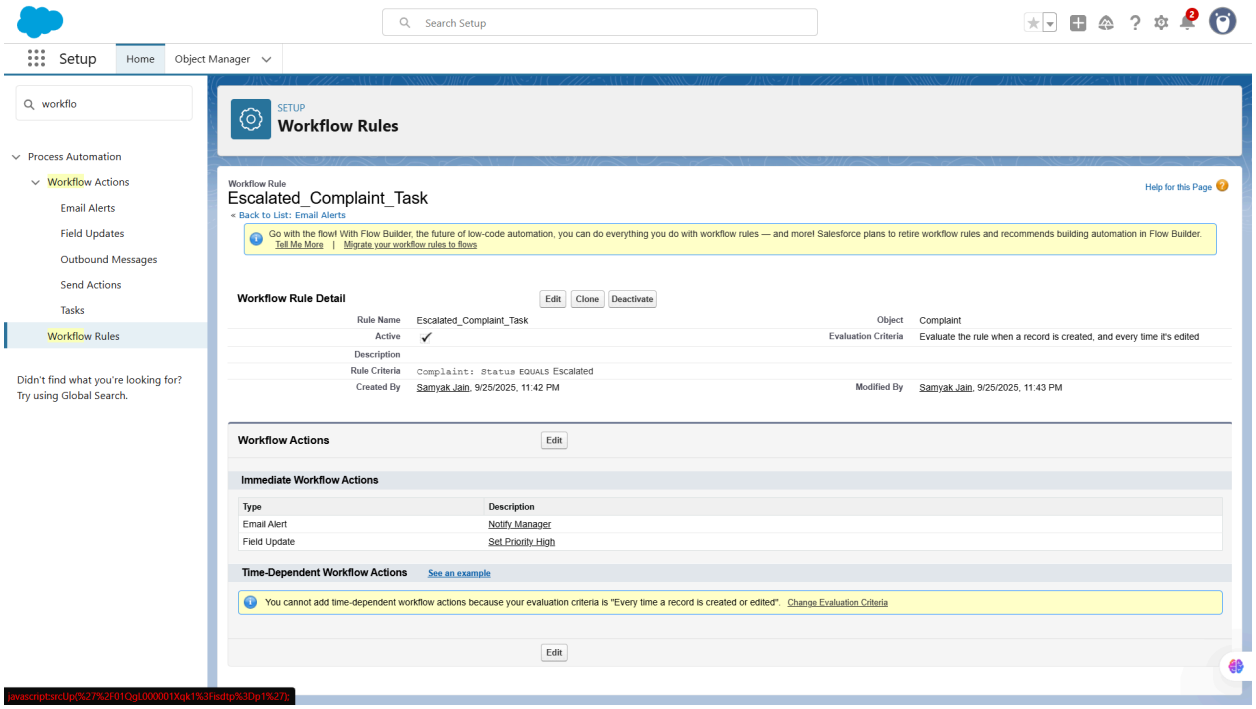
🔹 8. Tasks

We set up automated tasks for complaints and feedback to ensure timely follow-ups. Tasks were assigned to agents with due dates and priorities, helping track responsibilities and avoid missed actions.

Use Case:
Automatically assign tasks to users. Example: Create a task for support staff to follow up a complaint within 24 hours.

Implementation:

- Use **Workflow Rules, Process Builder, or Flow**
- Define Task properties: subject, due date, assignee



🔹 9. Custom Notifications

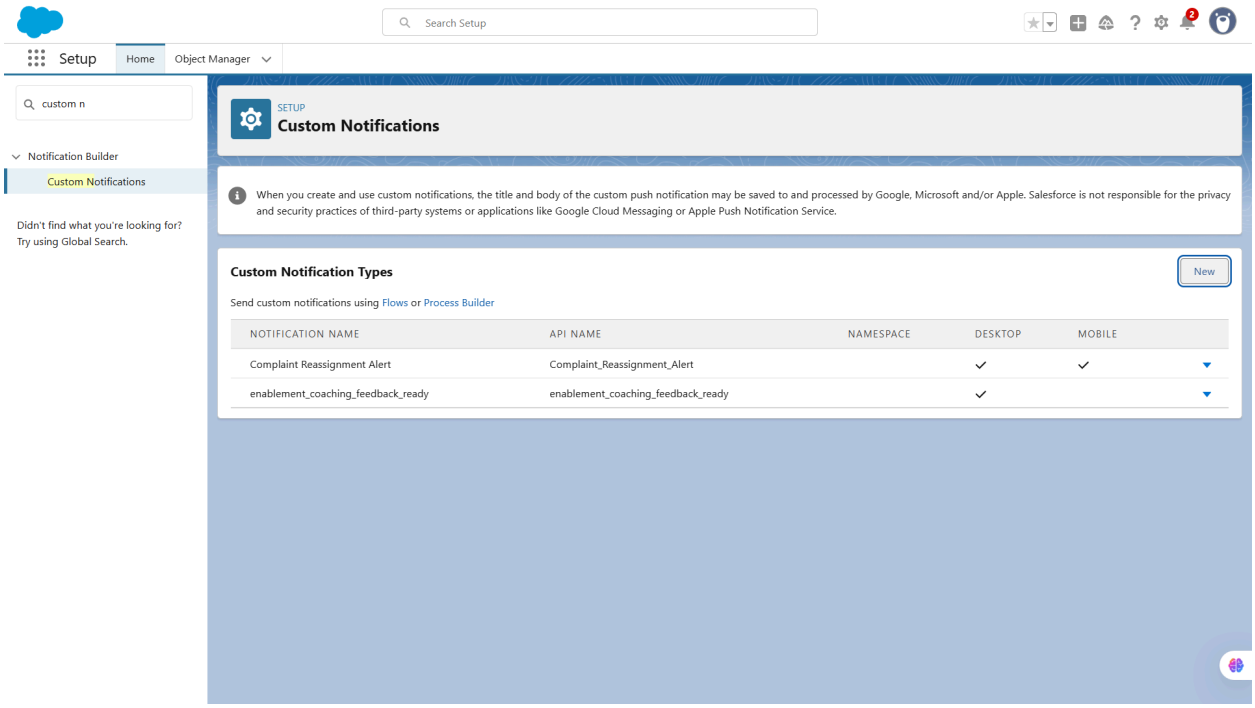
We configured real-time notifications for agents when complaints were assigned, escalated, or feedback required attention. These alerts appeared in Salesforce (desktop and mobile), enabling faster responses and improving overall customer service.

Use Case:

Notify users in Salesforce or mobile app when specific events occur. Example: Notify a support agent when a complaint is reassigned.

Implementation:

- **Setup → Custom Notifications → New**
- Define Notification Type
- Add to **Workflow/Process/Flow**

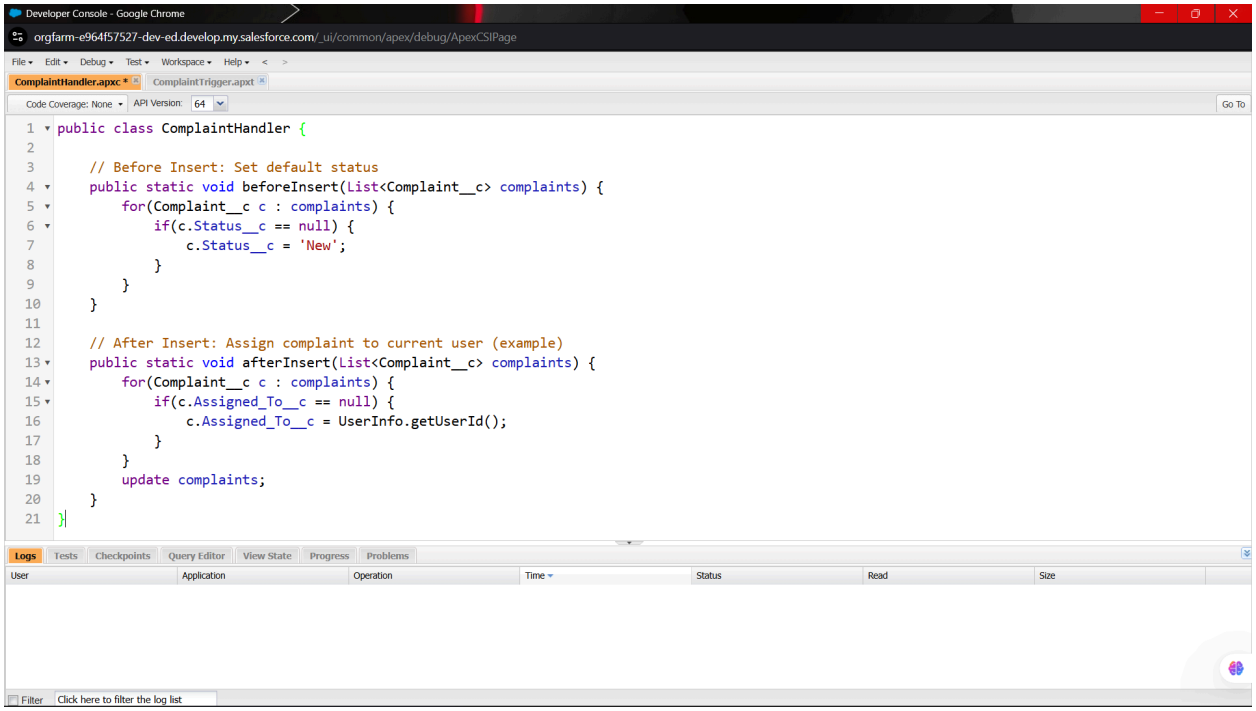

Search Setup
Setup
Home
Object Manager
custom n
Notification Builder
Custom Notifications
Didn't find what you're looking for?
Try using Global Search.
SETUP
Custom Notifications
When you create and use custom notifications, the title and body of the custom push notification may be saved to and processed by Google, Microsoft and/or Apple. Salesforce is not responsible for the privacy and security practices of third-party systems or applications like Google Cloud Messaging or Apple Push Notification Service.
Custom Notification Types
New
Send custom notifications using Flows or Process Builder
NOTIFICATION NAME
API NAME
NAMESPACE
DESKTOP
MOBILE
Complaint Reassignment Alert
Complaint_Reassignment_Alert
enablement_coaching_feedback_ready
enablement_coaching_feedback_ready

📌Phase-5: Apex Programming (Developer)

🔹 1. Classes & Objects

Created Apex classes to **encapsulate business logic** for complaints, feedback, and related processes. Objects were used to define instances of these classes and handle data operations efficiently.

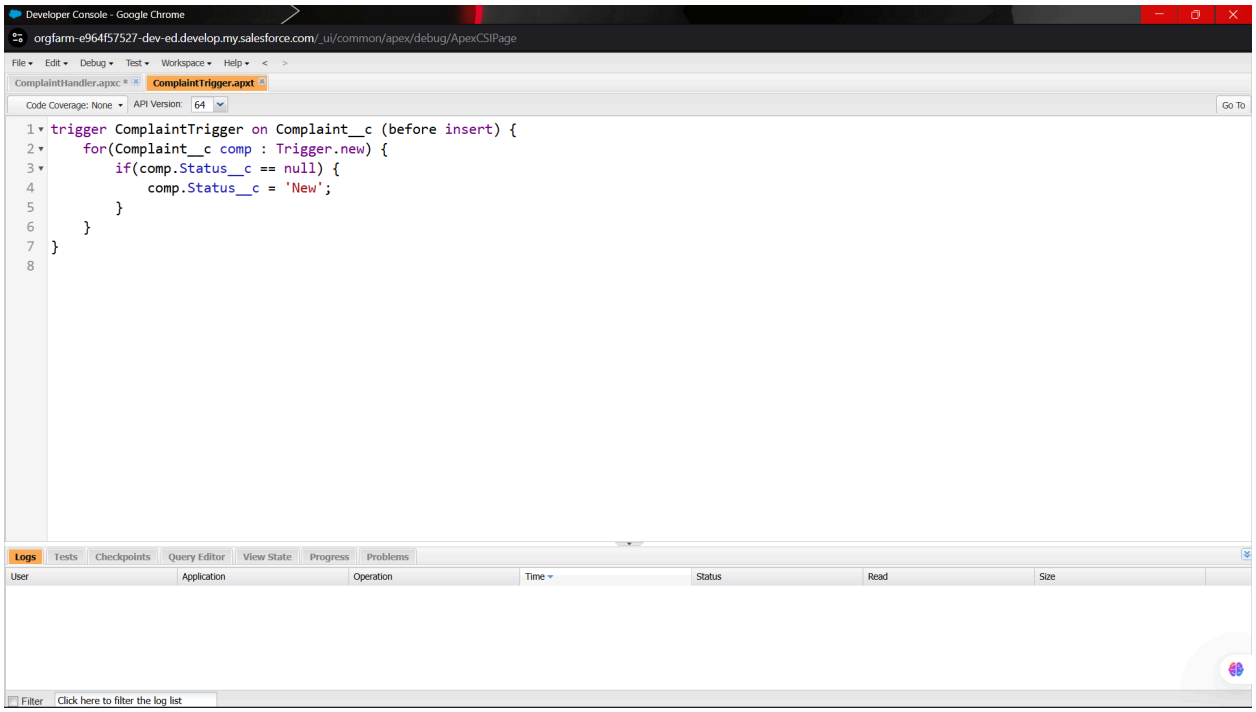
Use Case:
Apex **classes** are reusable blocks of code that can contain variables, methods, and logic. They help you organize code efficiently and encapsulate functionality. For example, you might have a `ComplaintHandler` class that processes complaints, validates data, and assigns tasks to support agents.



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

Implemented triggers to **automate actions on record changes**. For example, before inserting a complaint, we validated certain fields; after insert, we created related follow-up tasks.

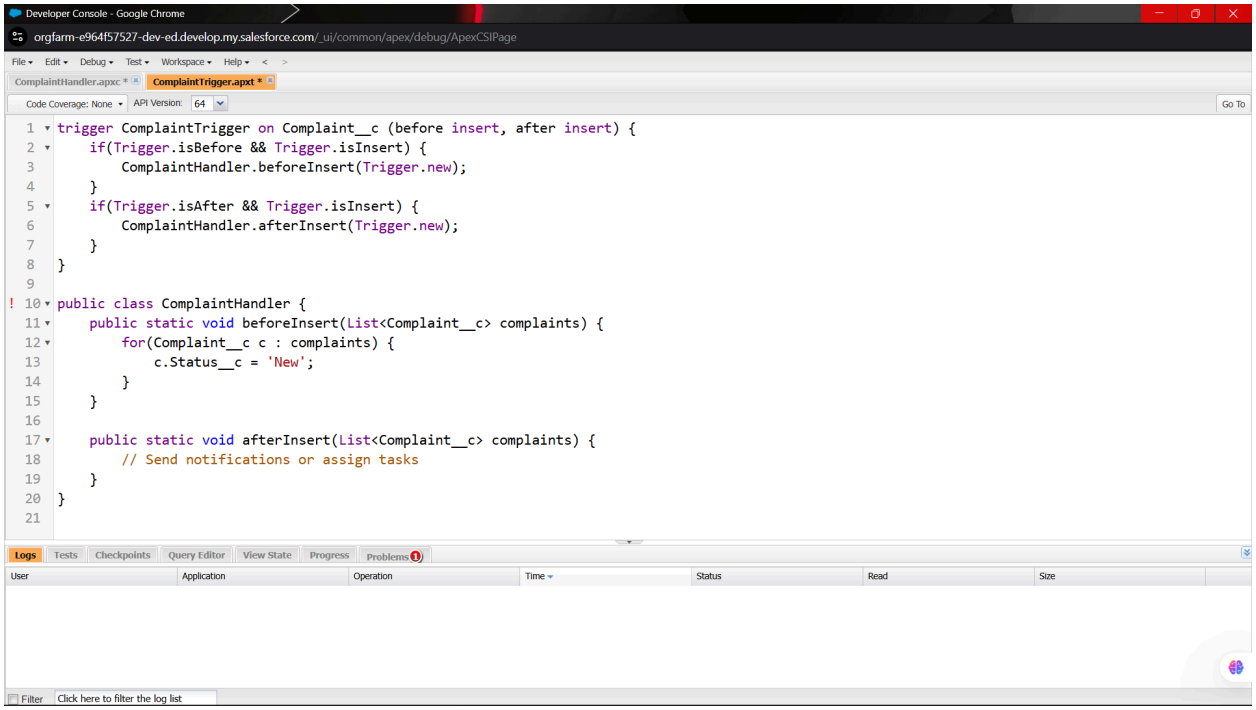
Use Case:
Triggers automatically execute Apex code before or after a record is **inserted, updated, deleted, or undeleted**.
Example: When a complaint is created, you want to **auto-assign it** to a queue or agent and send a notification.



🔹 3. Trigger Design Pattern

Used **trigger frameworks** (like one trigger per object and handler classes) to keep triggers modular, reusable, and easy to maintain, preventing code duplication and recursion issues.

Use Case:
Complex triggers can become messy. The **Trigger Handler Pattern** separates logic from the trigger to make it **clean, reusable, and testable**.



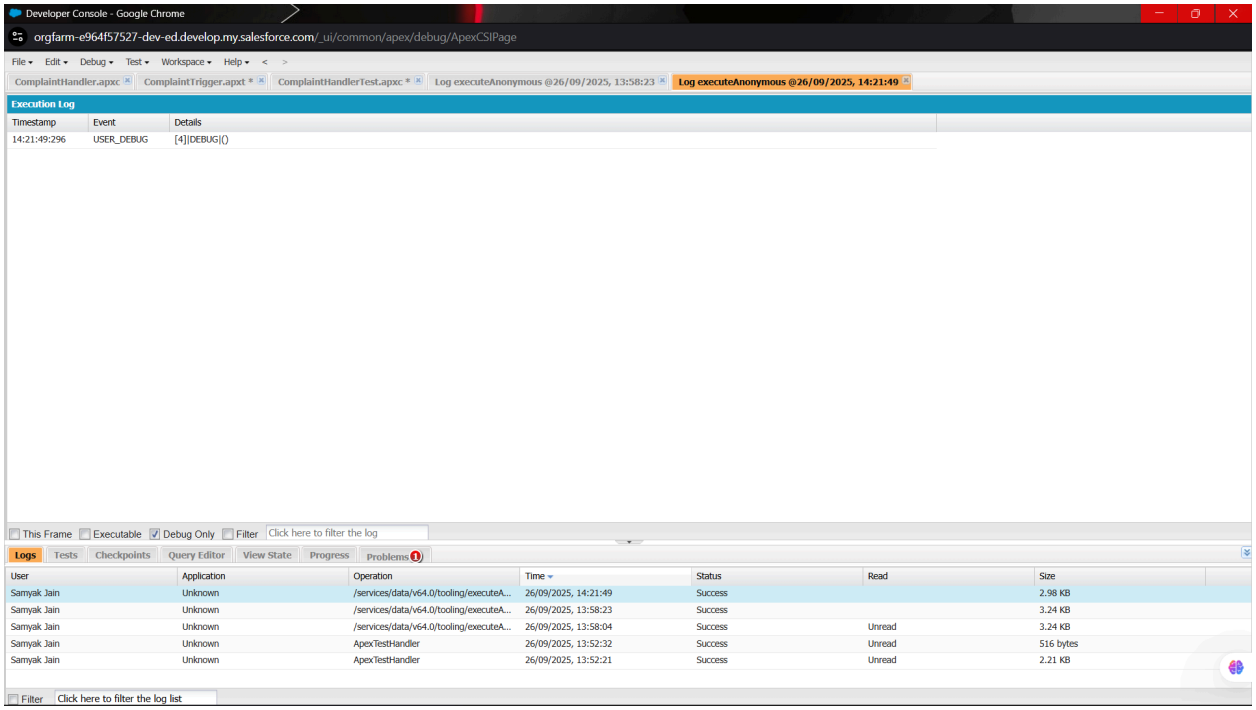
🔹 4. SOQL & SOSL

Wrote **SOQL queries** to retrieve records efficiently and **SOSL searches** to find records across multiple objects, supporting dynamic reporting and automation logic..

Use Case:

- **SOQL (Salesforce Object Query Language)**: Fetch specific records from Salesforce objects.
  Example: Find all complaints assigned to a particular agent.

- **SOSL (Salesforce Object Search Language)**: Search for text across multiple objects and fields.

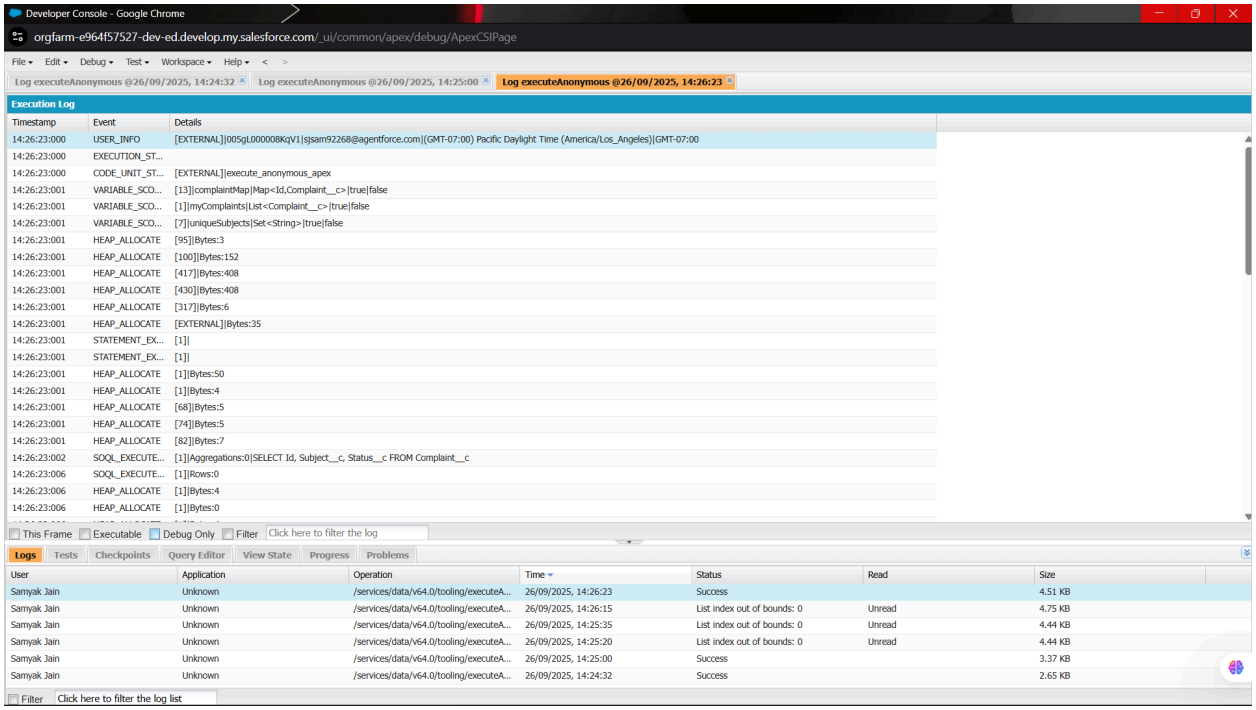


🔹 5. Collections: List, Set, Map

Used **Lists, Sets, and Maps** to store and process multiple records efficiently. For example, Maps were used to link complaint IDs with feedback records for bulk processing.

Use Case:
Collections store multiple values and are heavily used in Apex logic:

- **List:** Ordered collection of records.
- **Set:** Unique values, no duplicates.
- **Map:** Key-value pairs for fast lookups.



🔹 6. Control Statements

Used **Lists, Sets, and Maps** to store and process multiple records efficiently. For example, Maps were used to link complaint IDs with feedback records for bulk processing.

Use Case:
Used to implement decision-making and loops. Common in triggers, classes, batch jobs.
Example: Auto-escalate complaints based on priority.

```
for(Complaint__c c : complaints) {
    if(c.Priority__c == 'High') {
        c.Status__c = 'Escalated';
    } else {
        c.Status__c = 'In Progress';
    }
}
```

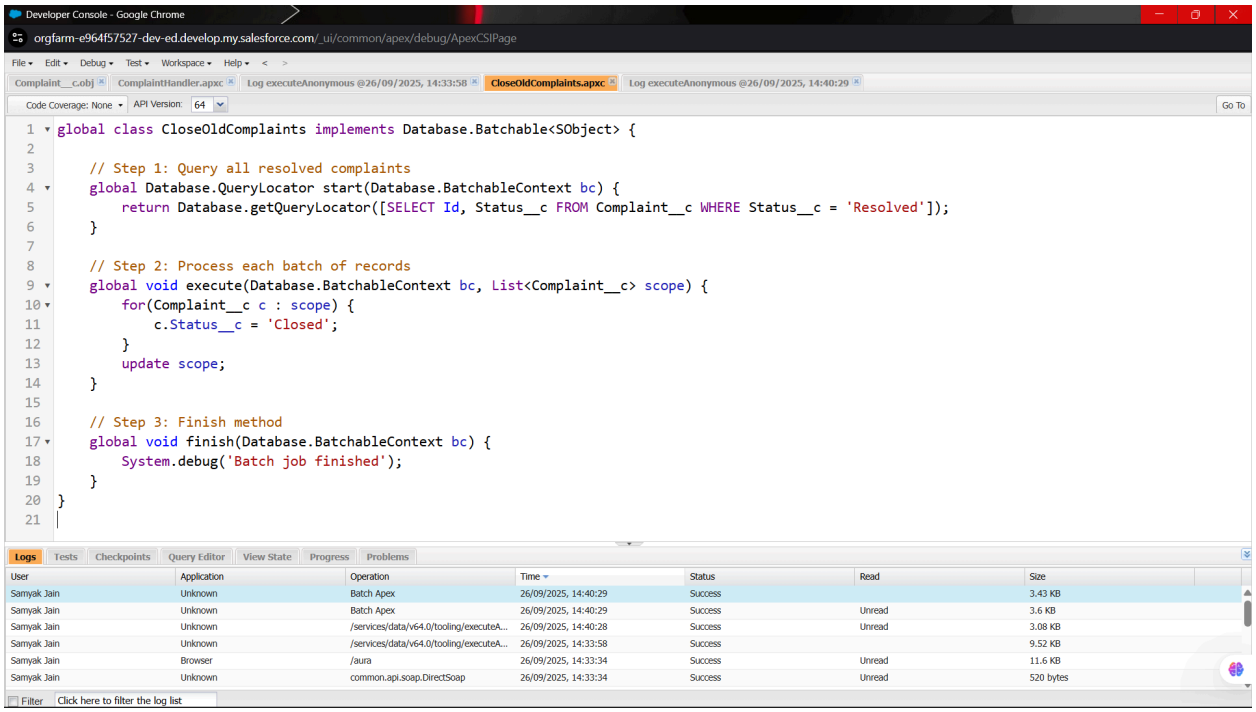
Status	Status
In Progress	Escalated
Priority	Priority
Medium	High
Assigned To	Assigned To
Samyak Jain	Samyak Jain

---

7 Batch Apex

Use Case:

Handles **large data volumes (over 50k records)** asynchronously. Example: Close all resolved complaints older than 90 days every night.



Enter Apex Code

```apex
CloseOldComplaints batch = new CloseOldComplaints();
Database.executeBatch(batch, 200);


global class ScheduleCloseComplaints implements Schedulable {
    global void execute(SchedulableContext sc) {
        CloseOldComplaints batch = new CloseOldComplaints();
        Database.executeBatch(batch, 200);
    }
}

```

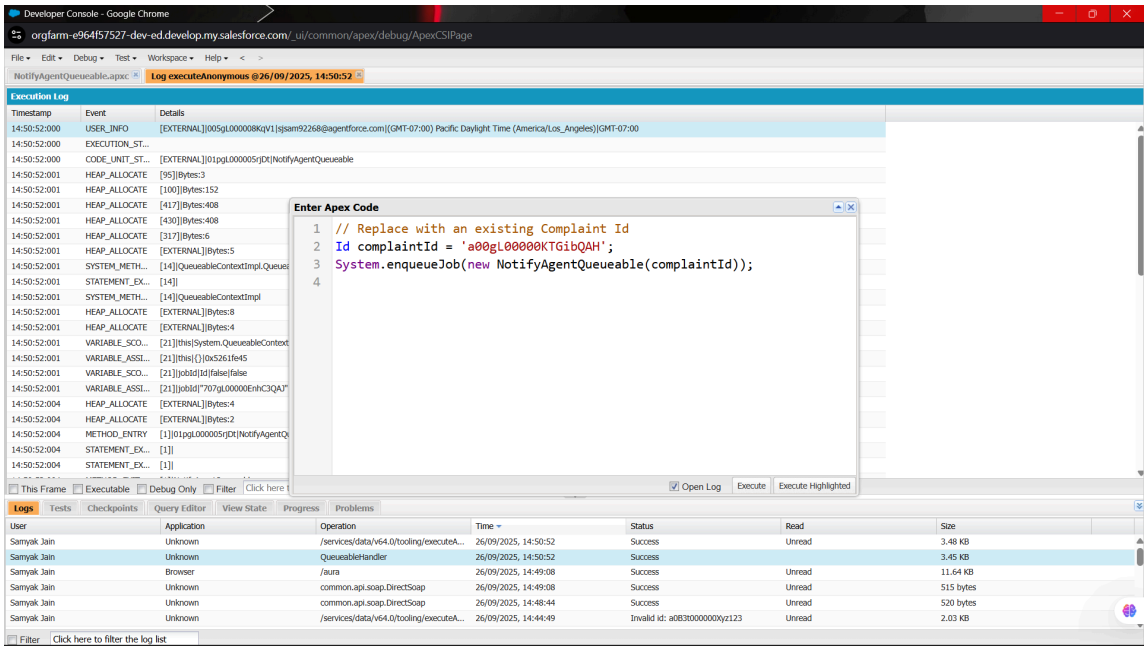
Open Log | Execute | Execute Highlighted

🔹 7. Queueable Apex

Implemented **Queueable Apex** for asynchronous processing of long-running tasks, like updating multiple related records without hitting governor limits.

Use Case:

Similar to Batch Apex but used for **smaller jobs** that need to run asynchronously. Example: Send notifications after complaint processing.

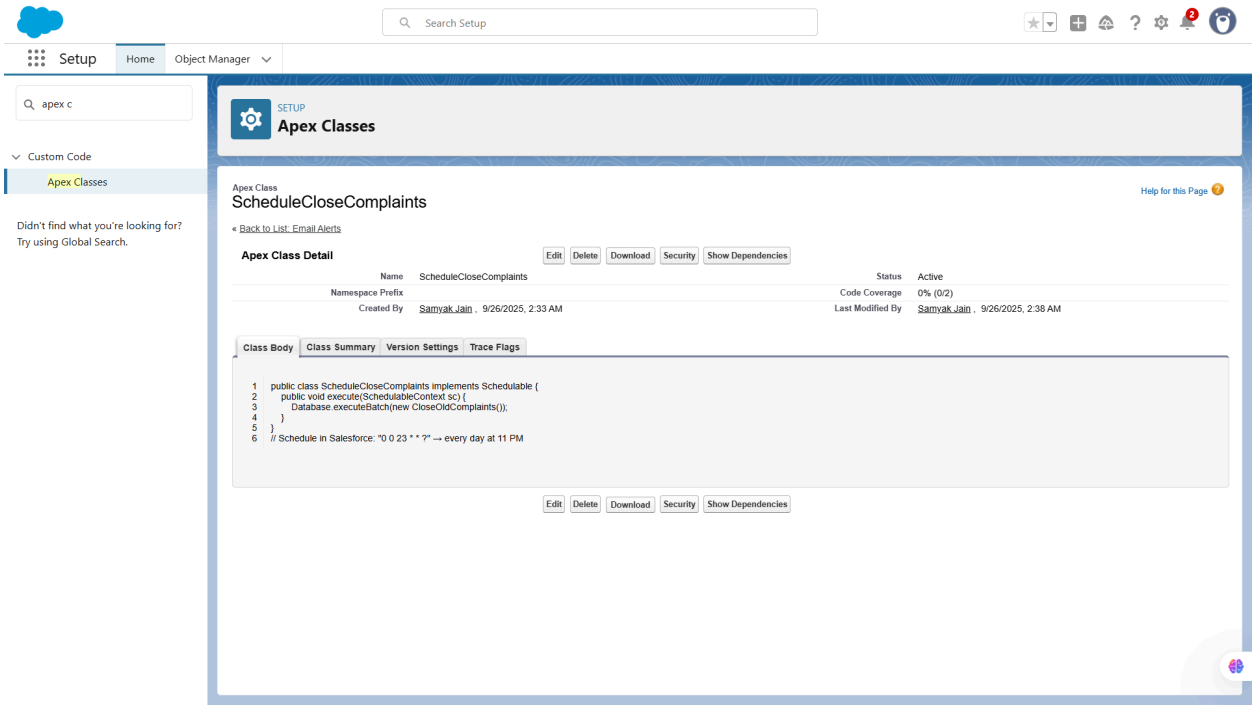


🔹 8. Scheduled Apex

Created **scheduled Apex jobs** to automate periodic processes, such as sending weekly complaint summary emails or cleaning old feedback records.

Use Case:
Run Apex at a scheduled time or interval. Example: Run the batch job to close old complaints every night at 11 PM.

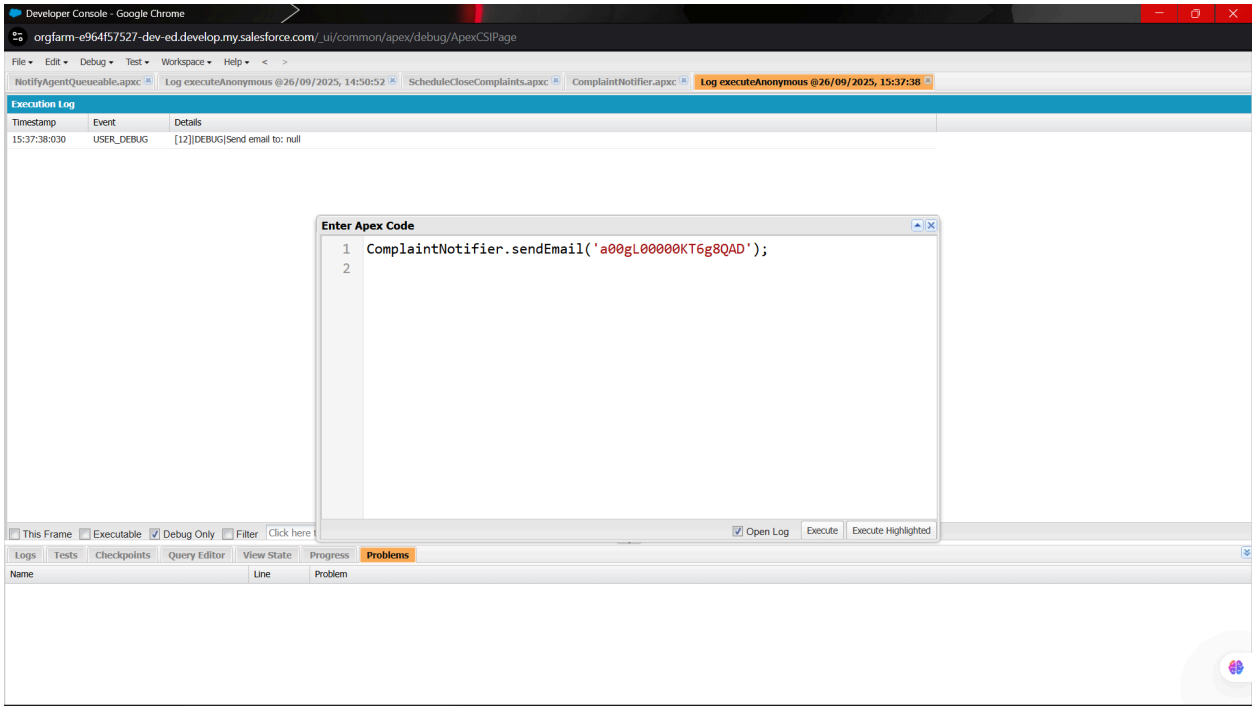
```
public class ScheduleCloseComplaints implements Schedulable {
    public void execute(SchedulableContext sc) {
        Database.executeBatch(new CloseOldComplaints());
    }
}
// Schedule in Salesforce: "0 0 23 * * ?" → every day at 11 PM
```



🔹 9. Future Methods

Used **future methods** for asynchronous callouts and operations that didn't require immediate execution, ensuring smooth user experience and system performance.

Use Case:
Run code asynchronously **without blocking the user**. Often used to call external APIs. Example: Call an external system to log complaint updates.

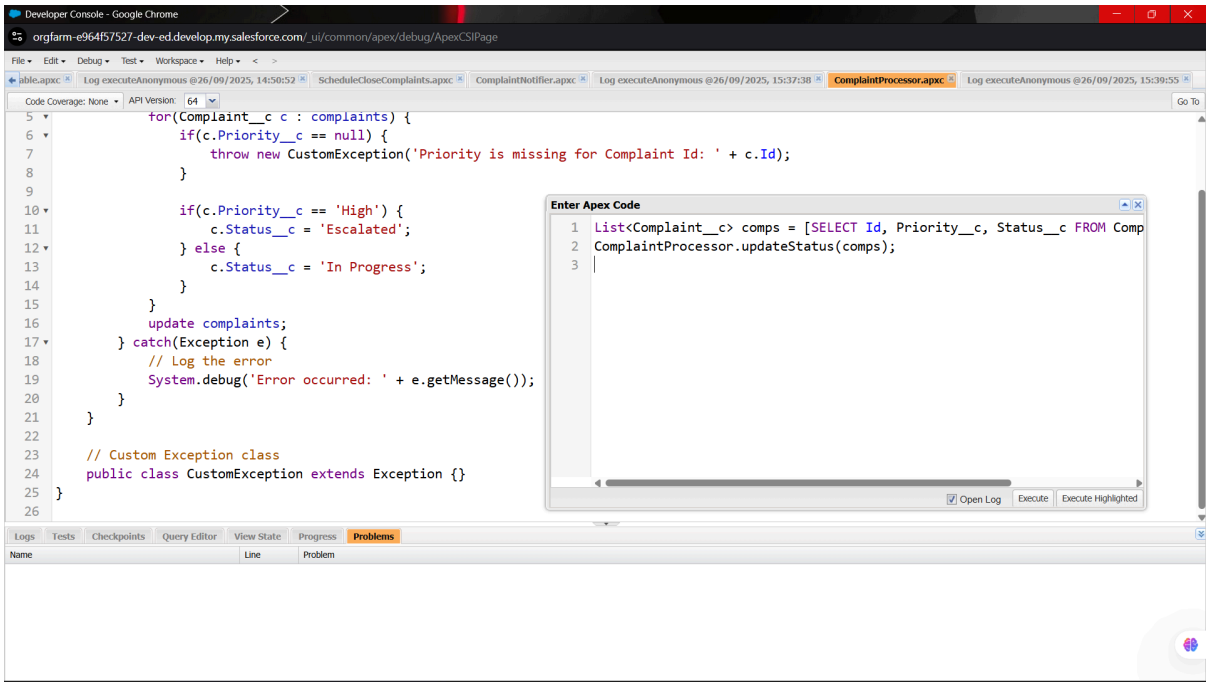


🔹 10. Exception Handling

Added **try-catch blocks** to gracefully handle errors in Apex code, ensuring that exceptions were logged and users received meaningful error messages instead of system failures.

Use Case:

Prevent runtime errors from crashing code. Example: Catch an error while updating complaints.

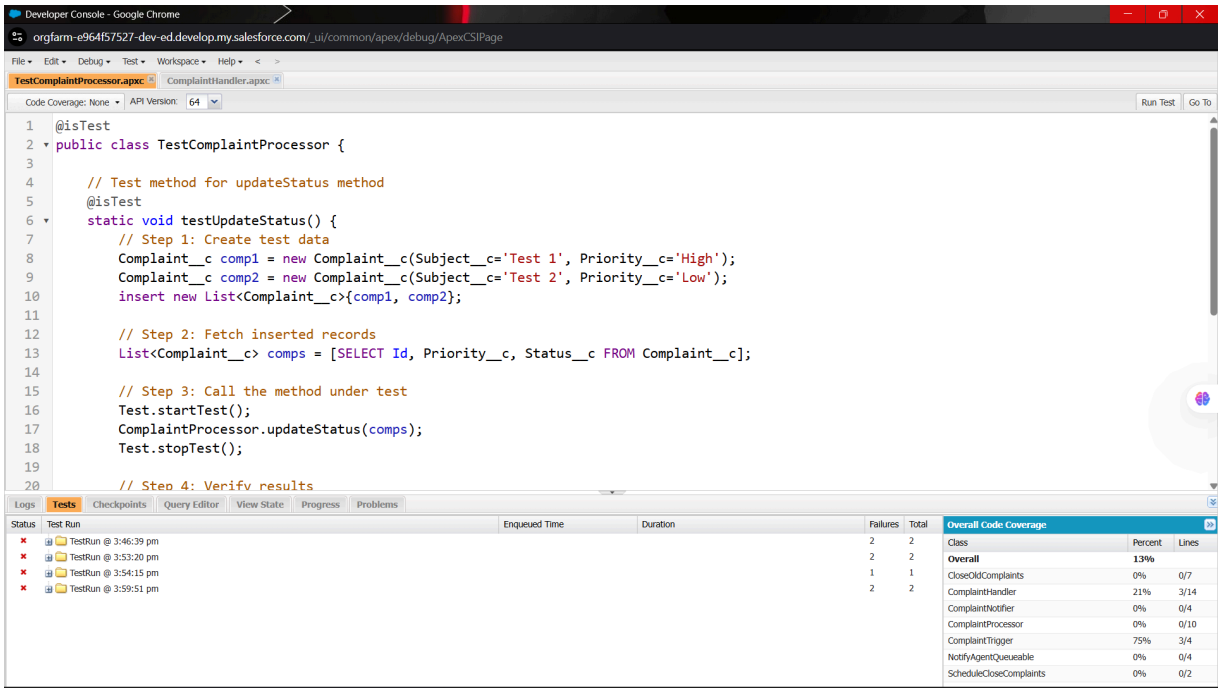


🔹 11. Test Classes

Developed **Apex test classes** to cover triggers and classes with positive and negative test scenarios. This ensured **code coverage compliance** and verified business logic correctness before deployment.

Use Case:

Ensure code coverage and validate functionality. Required for deploying to production.



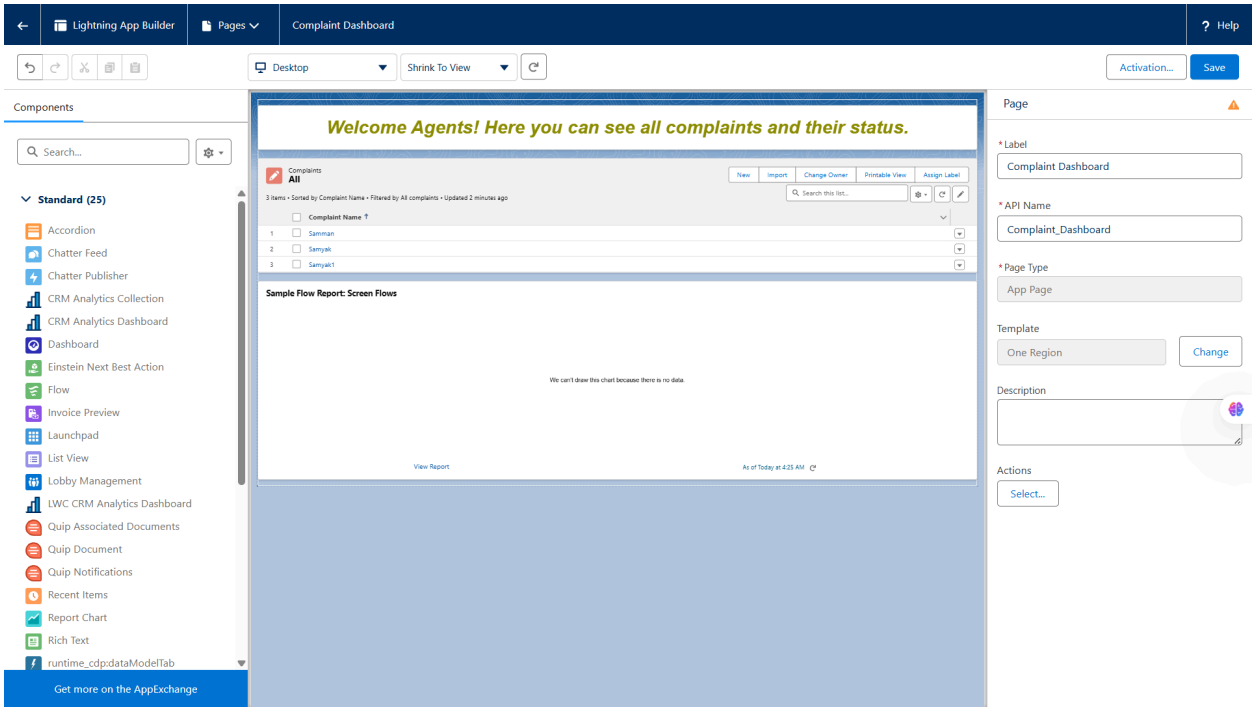
📌Phase-6: User Interface Development

🔹 1. Lightning App Builder

Use Case:
Lightning App Builder allows you to create **custom pages** for Salesforce Lightning Experience using a drag-and-drop interface. You can build **App Pages, Home Pages, and Record Pages** without coding. For example, in your Customer Complaint CRM, you can design a **custom complaint dashboard page** that shows all complaints, their statuses, and assigned agents.

Implementation:

1. Go to **Setup → Lightning App Builder → New**.
2. Choose **Record Page** or **App Page**.
3. Drag components like **Report Chart, Related Lists, Rich Text** onto the page.
4. Assign it to the **Complaint__c object**.
5. Activate the page for **Org Default, App Default, or Specific Profiles**.

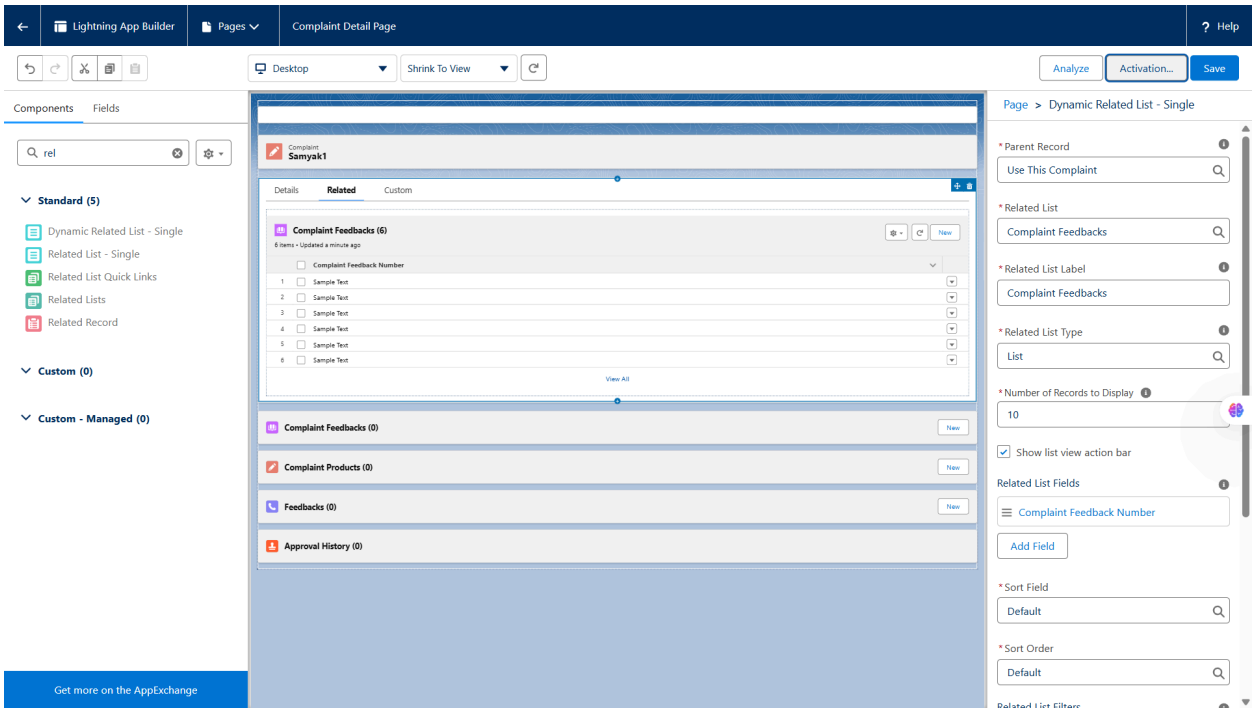


🔹 2. Record Pages

Use Case:
Record Pages control how **individual object records** (e.g., Complaint) are displayed. This allows you to customize the layout depending on **profile or role**. For instance, a **Support Agent Record Page** might show case details and assigned tasks, while a **Manager Record Page** shows reports and analytics.

Implementation:

- Use **Lightning App Builder** to create a **Record Page**.
- Add **Related Lists, Highlights Panel, Tabs, and Custom Components**.
- Assign it to **profiles** so that users see only relevant information.

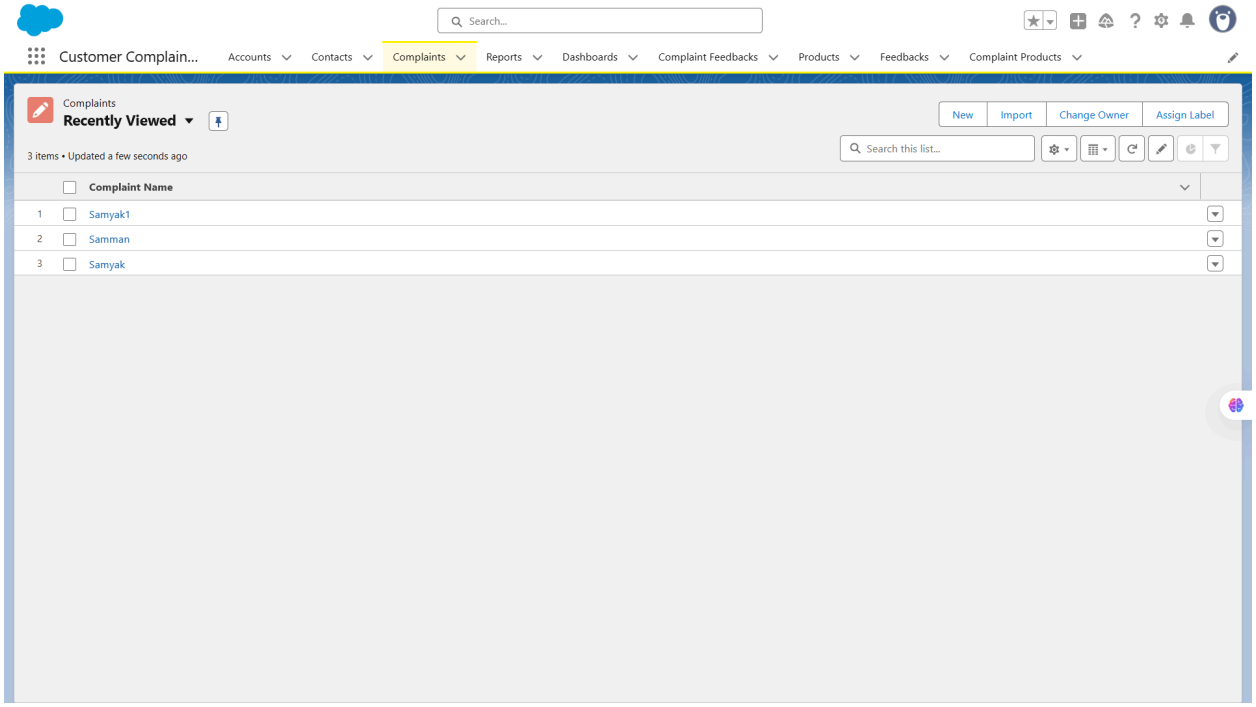


🔹 3. Tabs

Use Case:
Tabs let users **navigate to different objects or components** in Salesforce. In your CRM, you might create tabs for **Complaints, Customers, Products, and Dashboard** so agents can quickly switch between them.

Implementation:

1. Go to **Setup → Tabs → New**.
2. Select the object or Lightning Component you want to create a tab for.
3. Add it to your **Lightning App**.

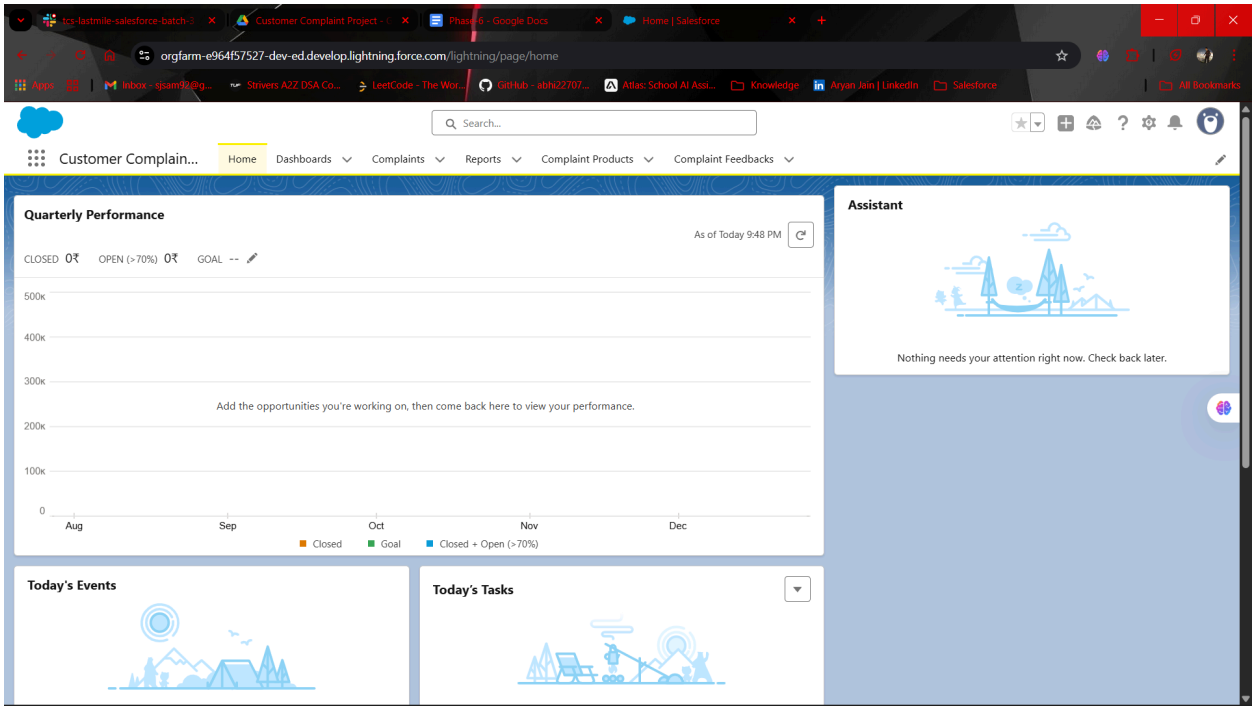


🔹 4. Home Page Layouts

Use Case:
Home Pages display **dashboard-like information** to users when they log in. For a Customer Complaint CRM, you can show **total complaints, pending complaints, tasks, and alerts** on the Home Page.

Implementation:

- Use **Lightning App Builder → Home Page**.
- Drag **components** like **List Views, Report Charts, News, and Custom Components**.
- Assign the layout to **profiles** so agents/managers have different dashboards.

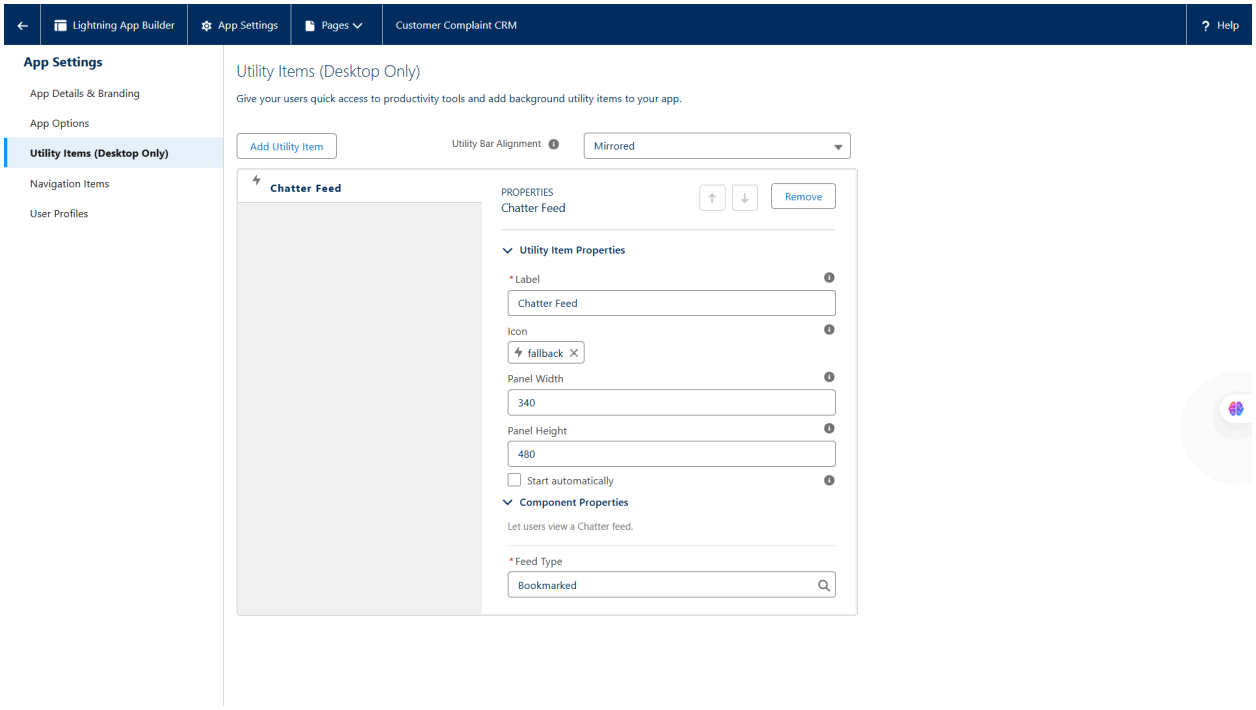


🔹 5. Utility Bar

Use Case:
Utility Bar provides **quick-access tools** at the bottom of Salesforce. For example, a Support Agent can have a **Quick Create Complaint, Chat, or Notes component** for faster operations.

Implementation:

1. Go to **Setup → App Manager → Edit Lightning App → Utility Bar**.
2. Add tools like **Quick Actions, Rich Text, or Custom LWC**.
3. Save and test.

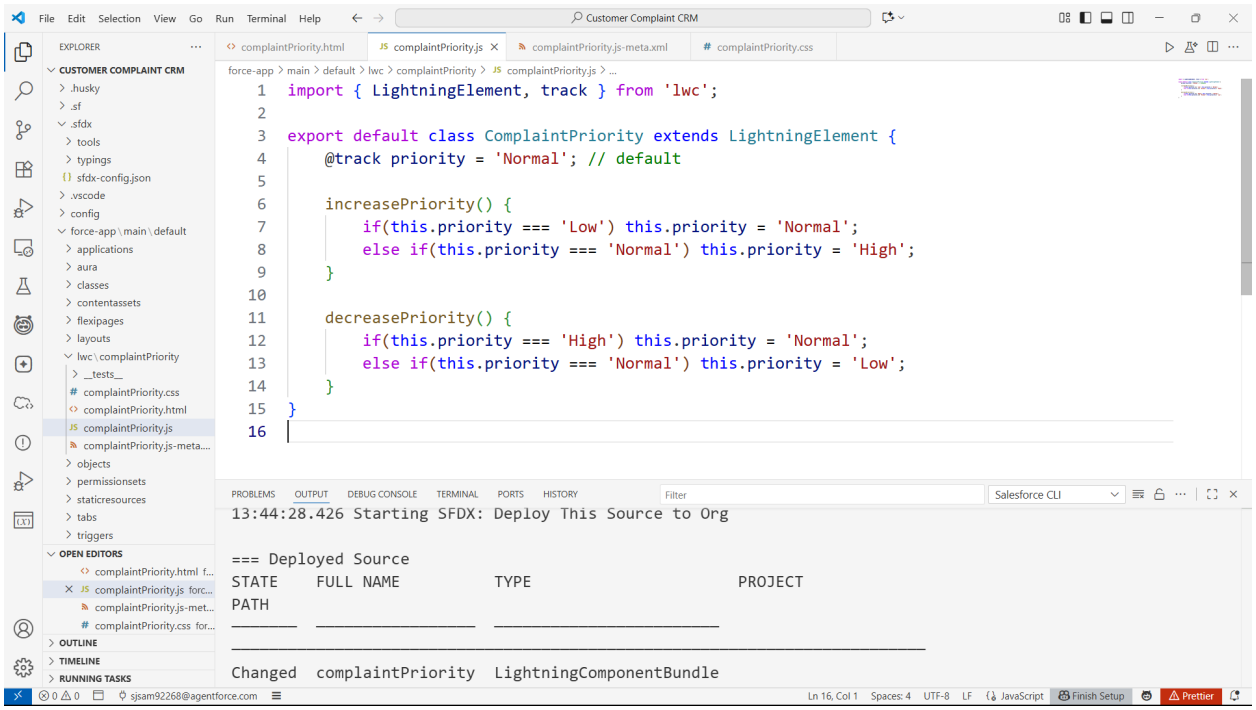


🔹 6. LWC (Lightning Web Components)

Use Case:
LWCs are **modern, reusable UI components** built with JavaScript, HTML, and CSS. They let you create **custom functionality** beyond standard Salesforce components. For example, a dynamic **complaint priority widget** that updates in real-time.

Implementation:

1. Use **VS Code with Salesforce Extension Pack**.
2. Create LWC: `sfdx force:lightning:component:create --type lwc --componentname ComplaintPriority`.
3. Add HTML, CSS, JS logic.
4. Deploy to org and use it on **Record Pages or App Pages**.

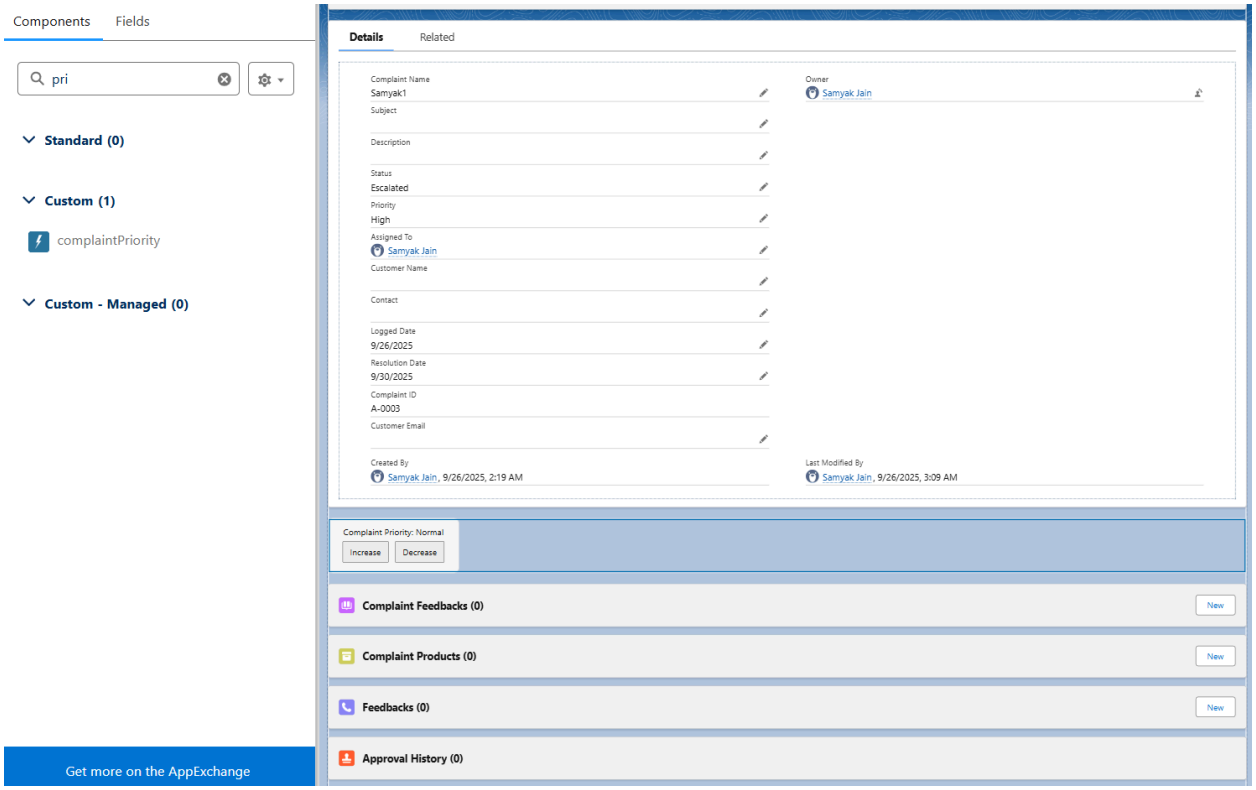


🔹 7. Apex with LWC

Use Case:
Sometimes, LWC needs **server-side logic** for data that cannot be handled on the client. For example, **fetching complaints with filters and aggregations** via Apex.

Implementation:

1. Create **Apex class** with `@AuraEnabled(cacheable=true)` method.
2. Call this Apex method in your **LWC JS file** using `import methodName from '@salesforce/apex/YourClass.methodName';`.
3. Handle the response in LWC and display data.

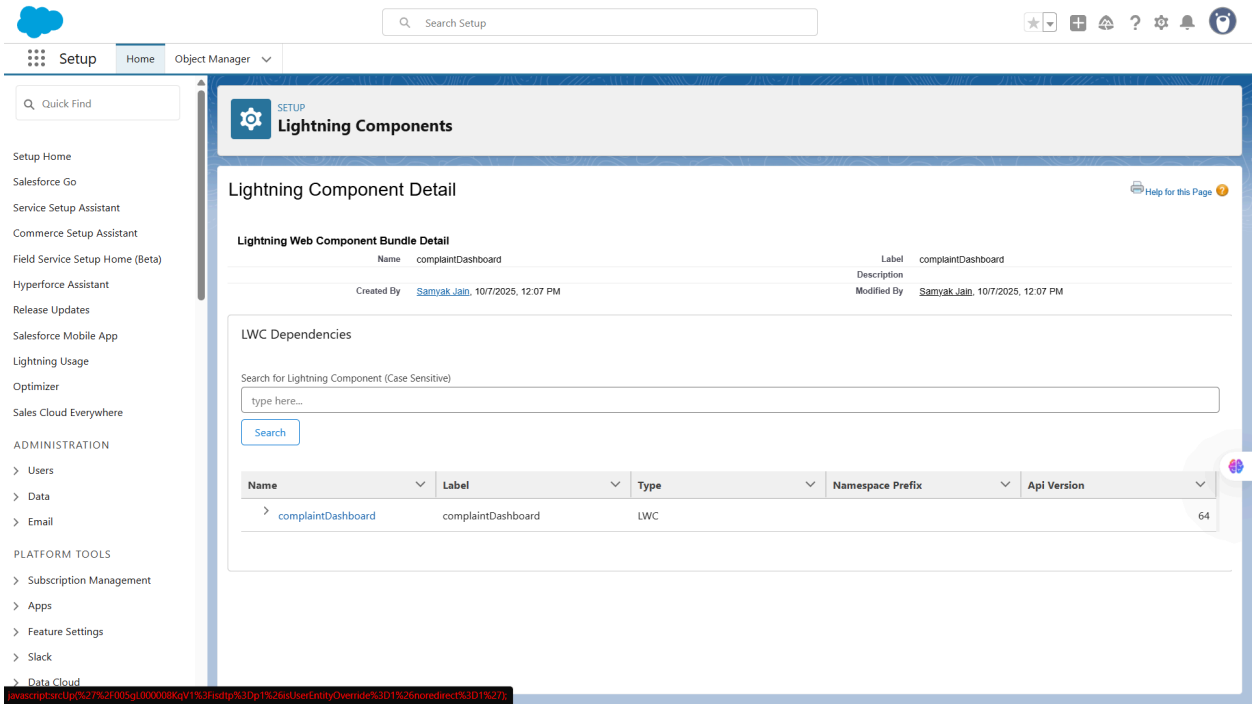


🔹 8. Events in LWC

Use Case:
Events allow **communication between components**. For instance, when a complaint is closed in one component, another component (like a dashboard) should update automatically.

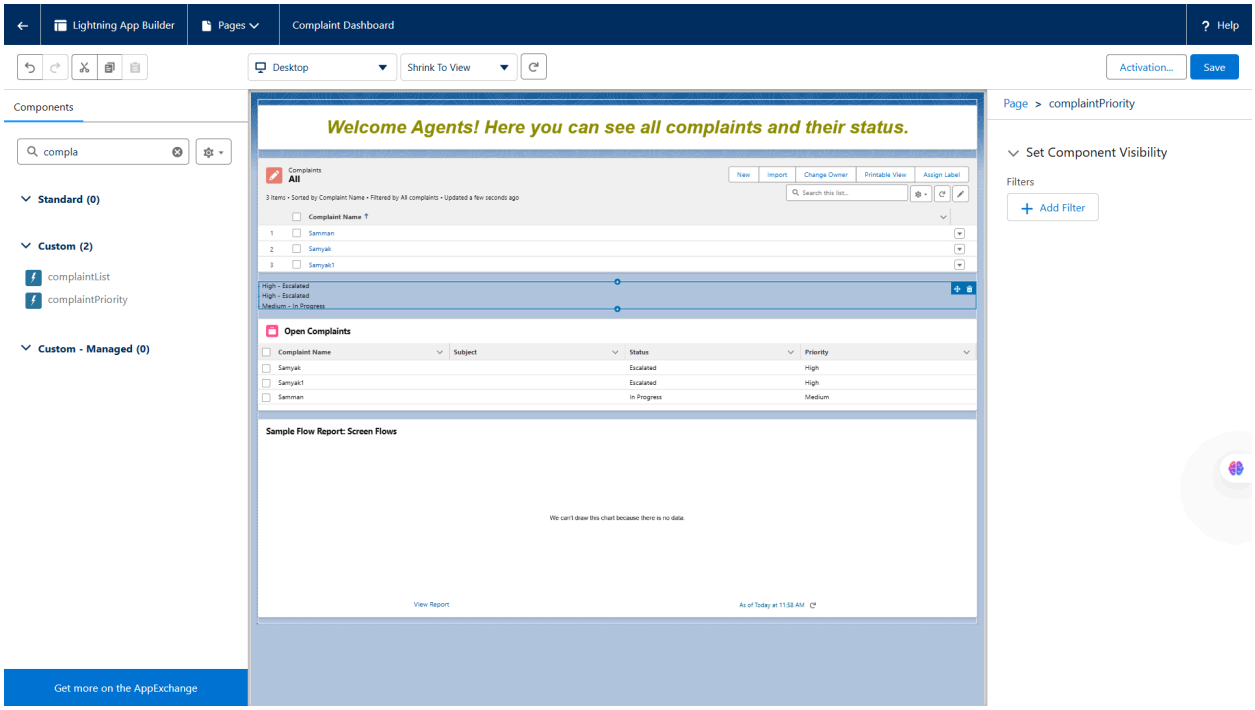
Implementation:

- Use **Custom Events**: `this.dispatchEvent(new CustomEvent('complaintclosed', { detail: { id: recordId } }))`.
- The parent component listens: `<c-child-component oncomplaintclosed={handleEvent}></c-child-component>`.



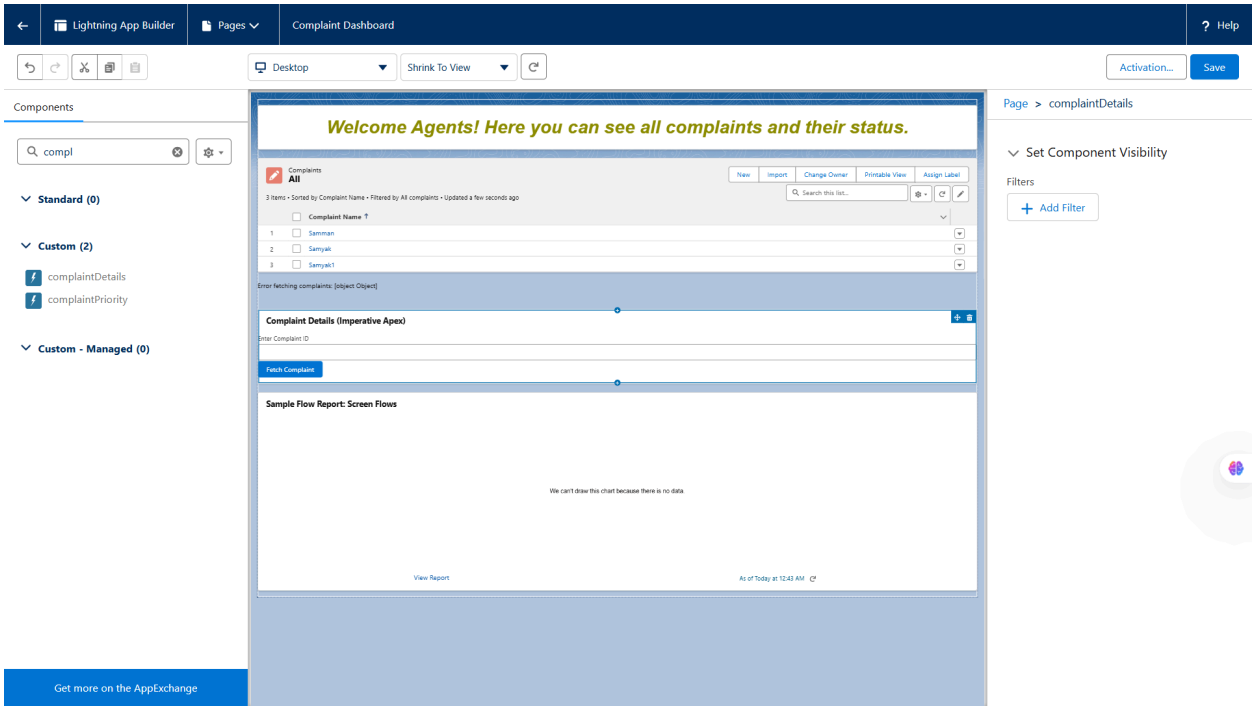
🔹 9. Wire Adapters

Use Case:
Wire Adapters let you **reactively get Salesforce data** in LWC. For example, automatically displaying **all open complaints** in a list component that updates whenever records change.



🔹 10. Imperative Apex Calls

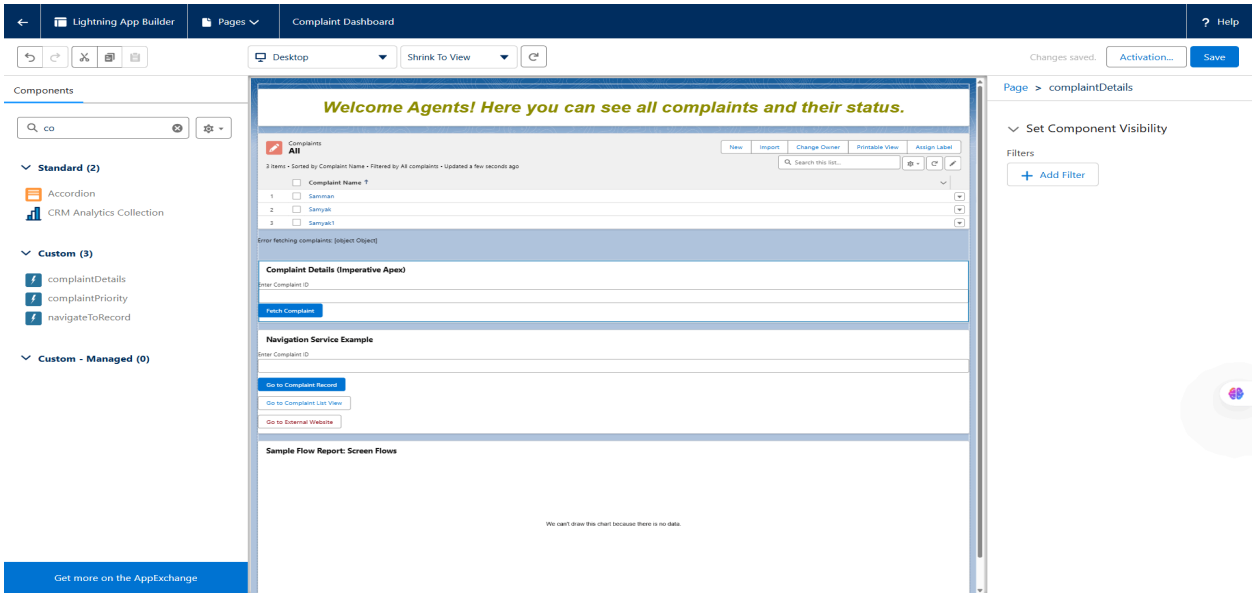
Use Case:
Imperative calls allow **manual invocation of Apex** instead of reactive wiring. Useful for **actions triggered by buttons or form submissions**.



🔹 11. Navigation Service

Use Case:

Navigation Service lets you **programmatically navigate** users to **records, lists, or external URLs**. For example, after creating a complaint, the agent is automatically redirected to its detail page.



📌 Phase 7: Integration & External Access

🔹 1. Named Credentials

Use Case:
Simplify and secure authentication for external callouts (e.g., sending complaint details to an external service).
Implementation Steps:

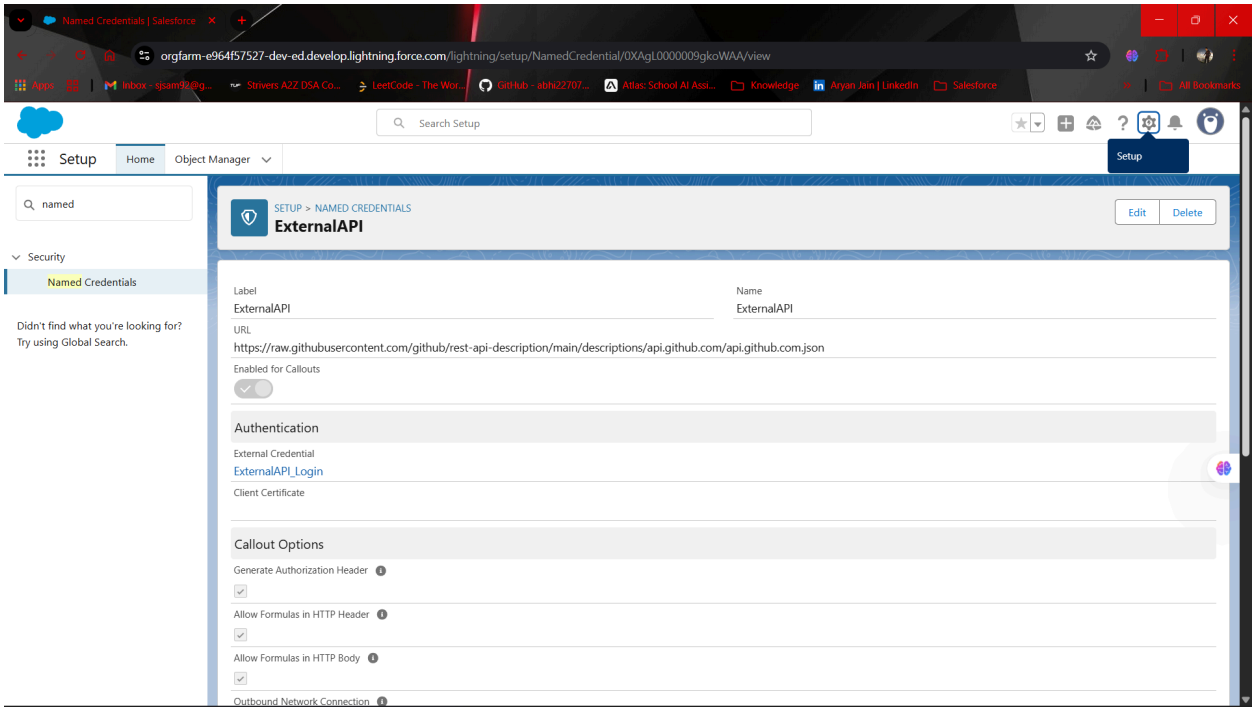
1. Setup → *Named Credentials* → *New*.
2. Provide external endpoint URL and authentication method.
3. Reference it in Apex

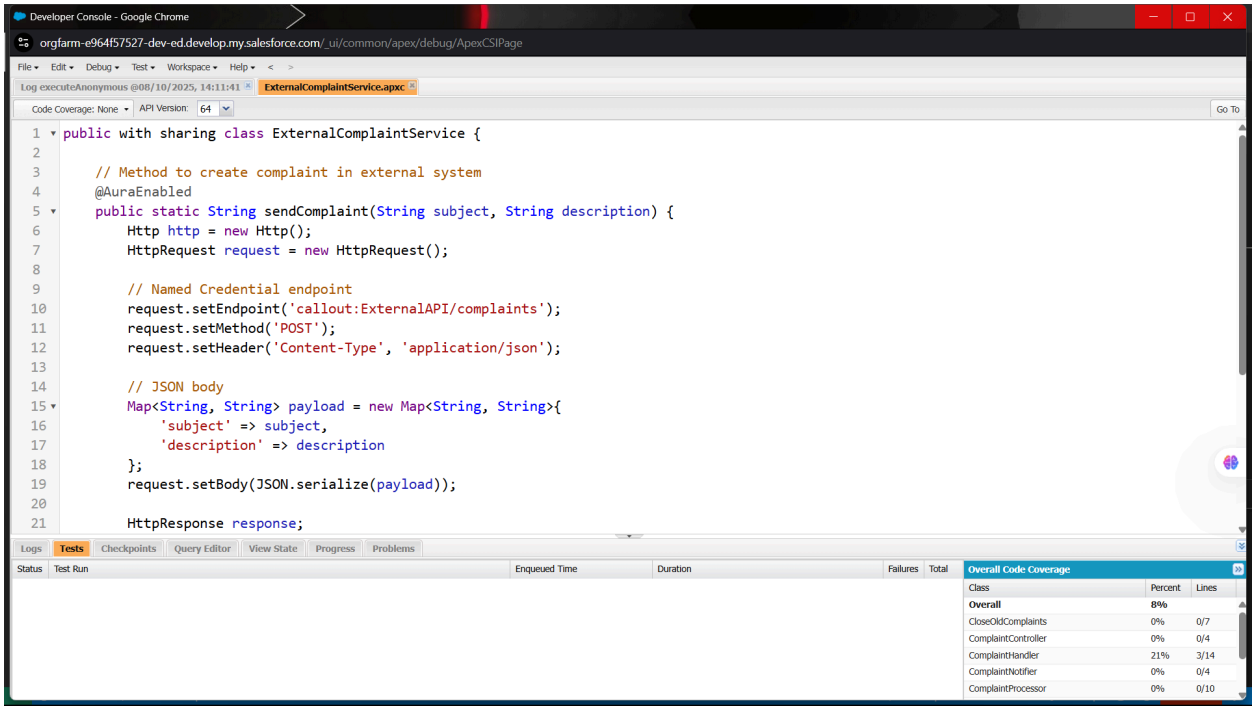
🔹 2. External Services

Use Case:
Integrate Salesforce with REST APIs without writing Apex — automatically create invocable actions.

Implementation Steps:

1. Setup → *External Services* → *New*.
2. Register schema via Swagger/OpenAPI definition.
3. Use the generated actions directly in Flow or Process Builder.

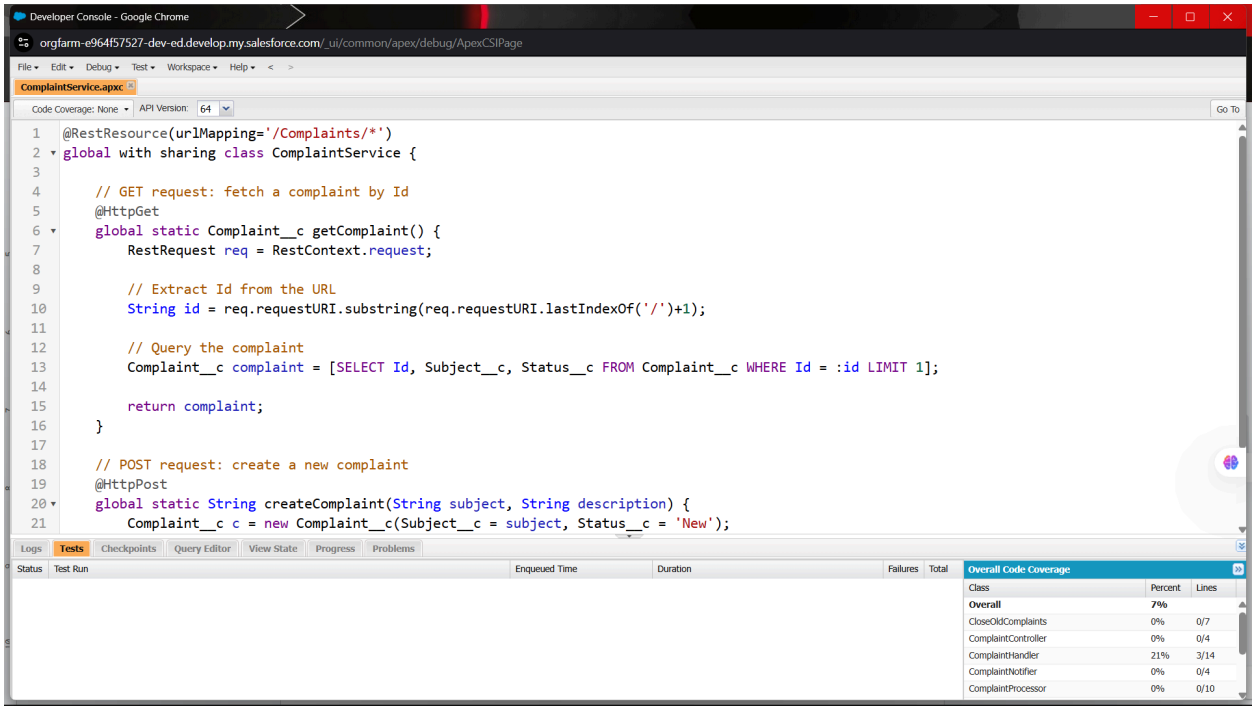




🔹 3. Web Services (REST/SOAP)

Use Case:
Expose your Complaint object data to external systems.
Implementation Steps:

- **REST:** Create an `@RestResource` Apex class.

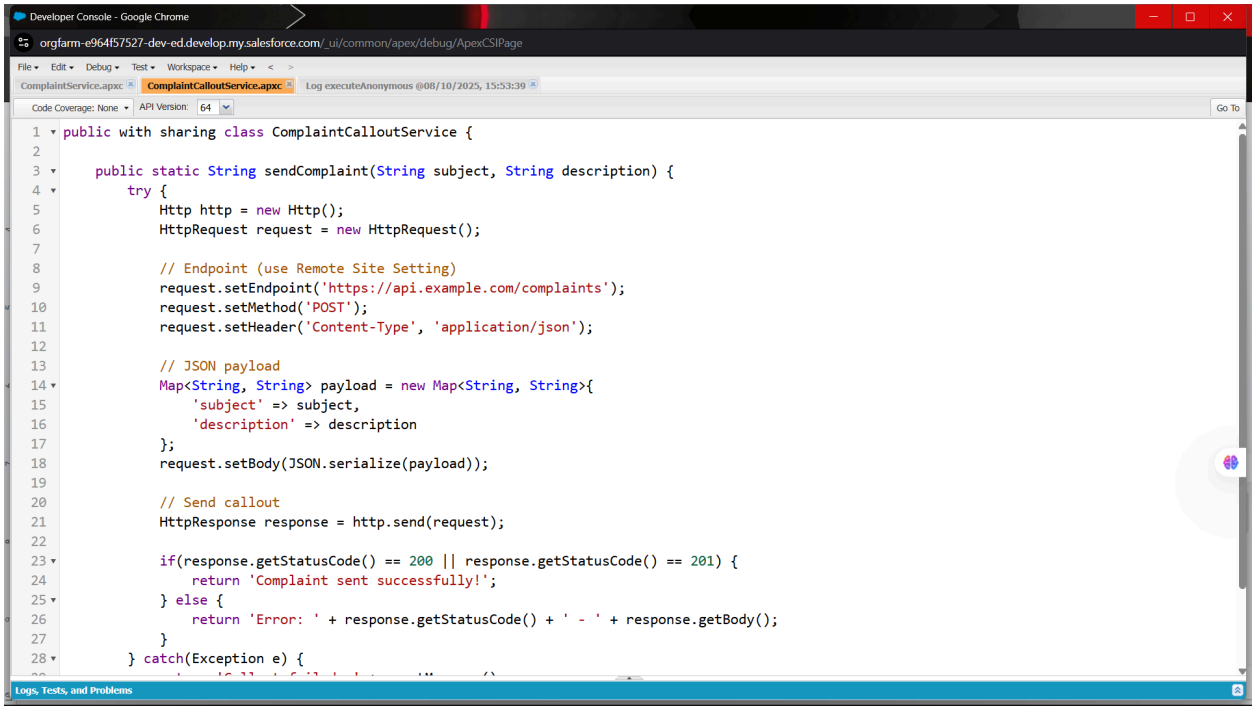


🔹 4. Platform Events

Use Case:
Notify external or internal systems in real time when complaints change (publish–subscribe model).

Implementation Steps:

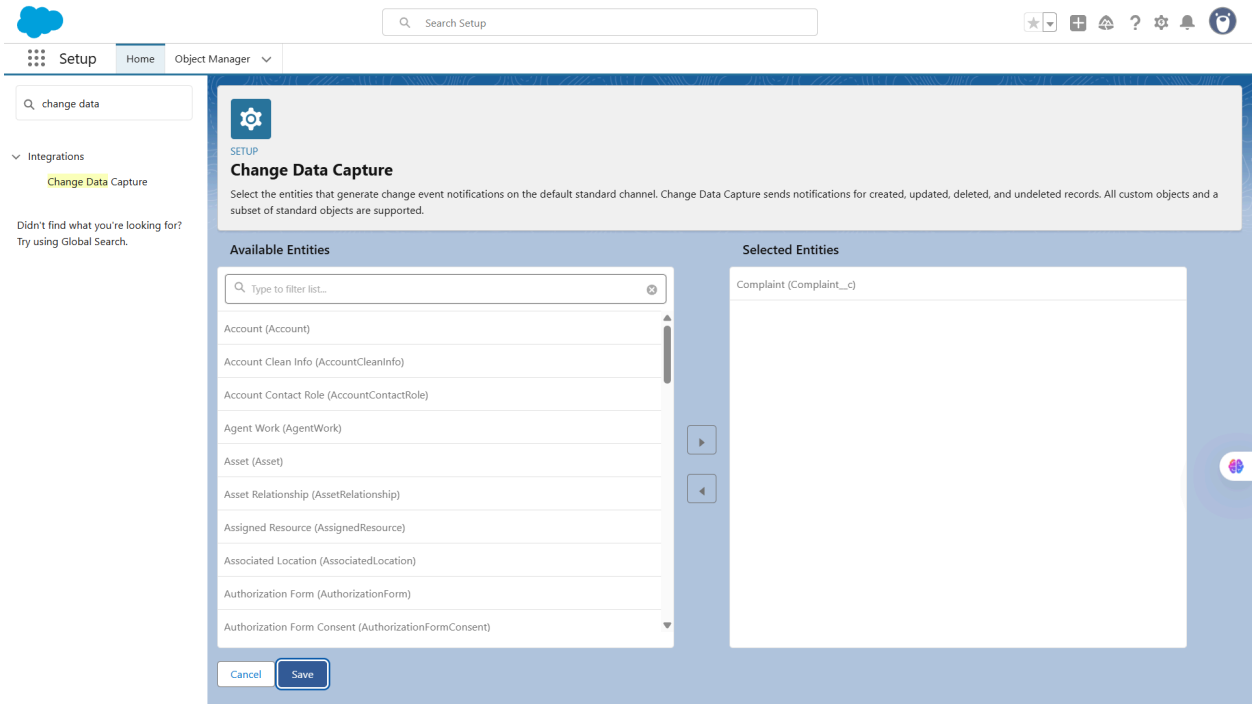
- Create a *Platform Event* like `Complaint_Updated__e`.
- Publish it via Apex or Flow.
- Subscribe using external systems or internal triggers.



🔹 5. Change Data Capture (CDC)

Use Case:
Stream changes of Complaint records to external systems automatically.
Implementation Steps:

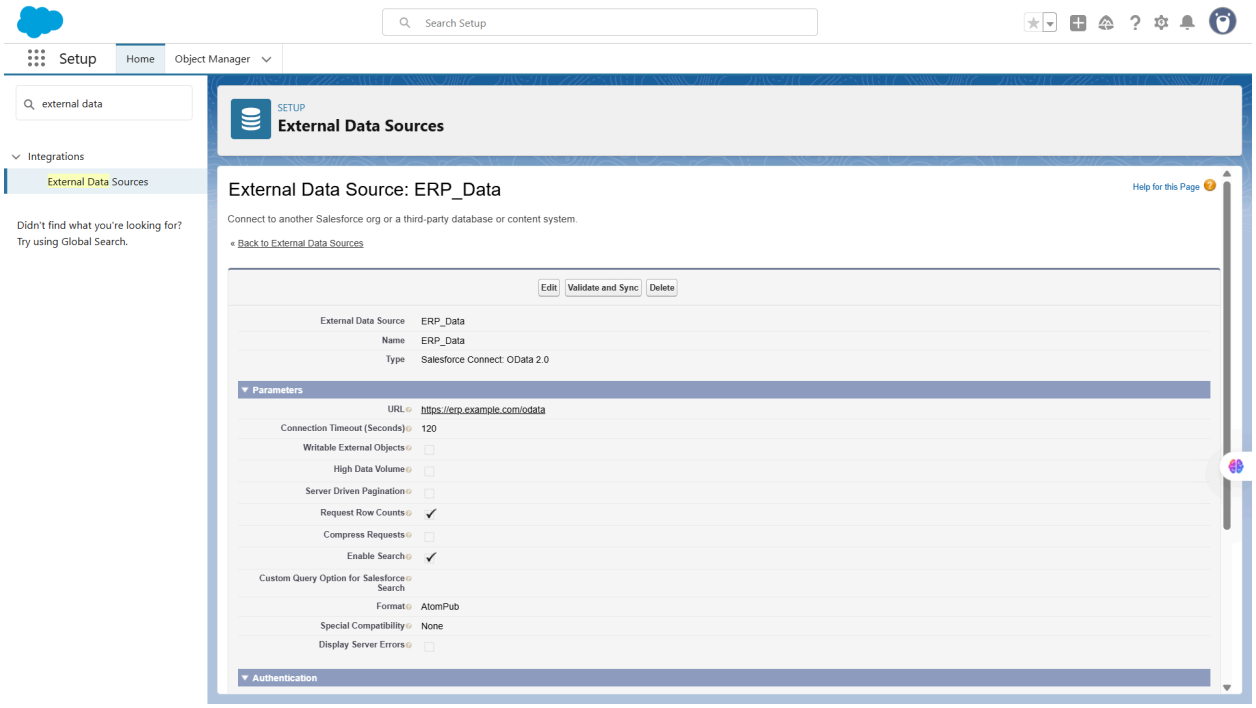
- Enable CDC for `Complaint__c` object in Setup.
- External system subscribes to CDC events via CometD API.



🔹 6. Salesforce Connect

Use Case:
Access external data (e.g., product or customer info from ERP) in real-time *without storing it in Salesforce*.
Implementation Steps:

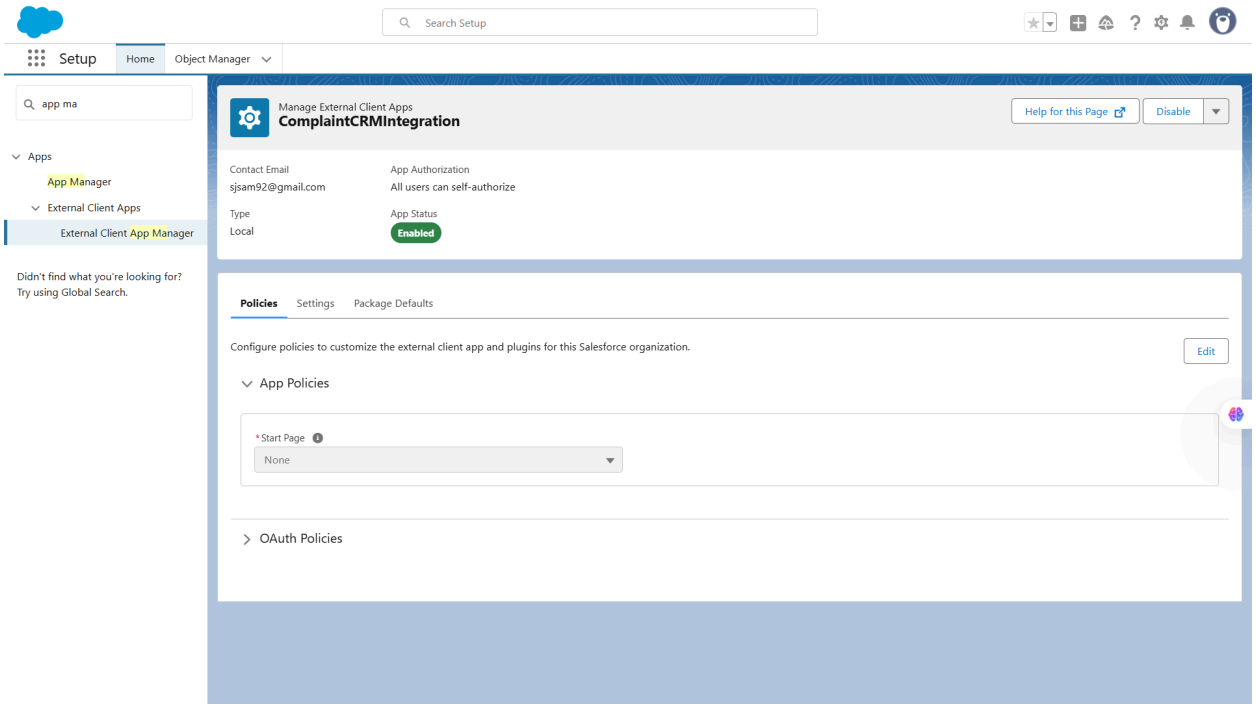
- Setup → *Salesforce Connect* → *New External Data Source*.
- Choose adapter (OData 2.0 / 4.0).
- Validate and sync external objects.



🔹 7. OAuth & Authentication

Use Case:
Secure API access between Salesforce and external apps.
Implementation Steps:

- Setup → *App Manager* → *New Connected App*.
- Configure OAuth scopes (e.g., `api`, `refresh_token`).
- Use consumer key & secret in the external system for authentication.



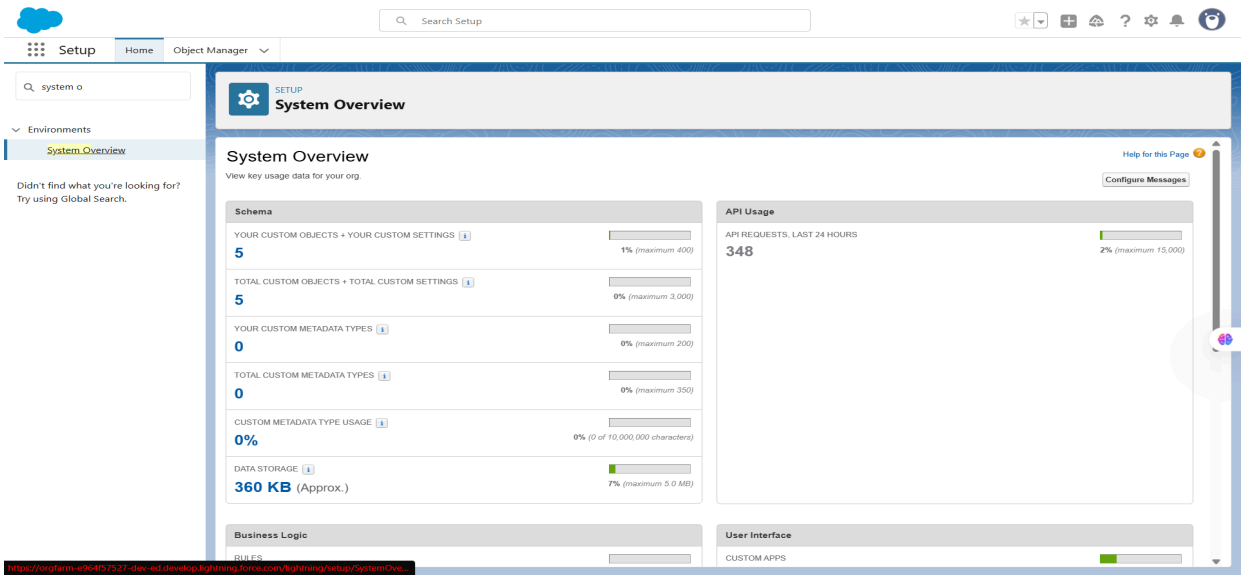
⚙️ 8. API Limits

Use Case:

Ensure integrations stay within Salesforce governor and daily API call limits.

Implementation Steps:

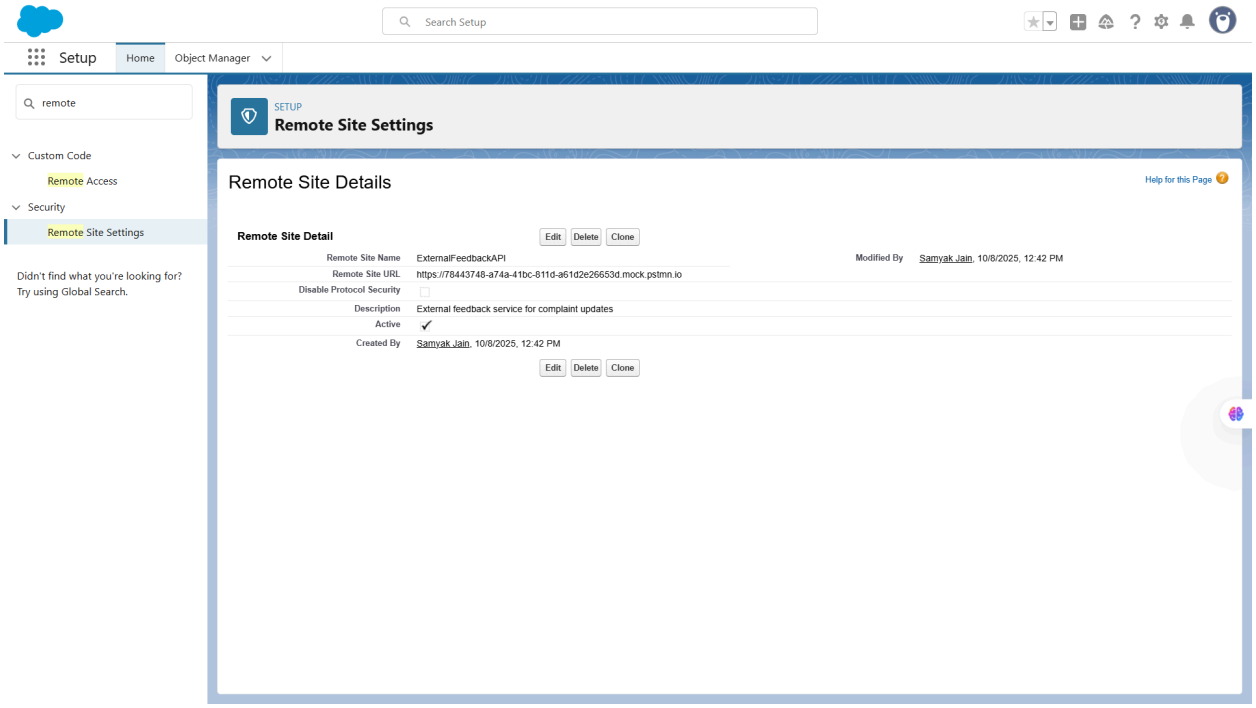
- Monitor usage in *Setup → System Overview → API Requests*.



🔹 9. Remote Site Settings

Use Case:
Authorize Salesforce to make callouts to external URLs.
Implementation Steps:

- Setup → Security → *Remote Site Settings* → *New*.
- Add domain URL of the external API (e.g., `https://api.myservice.com`).



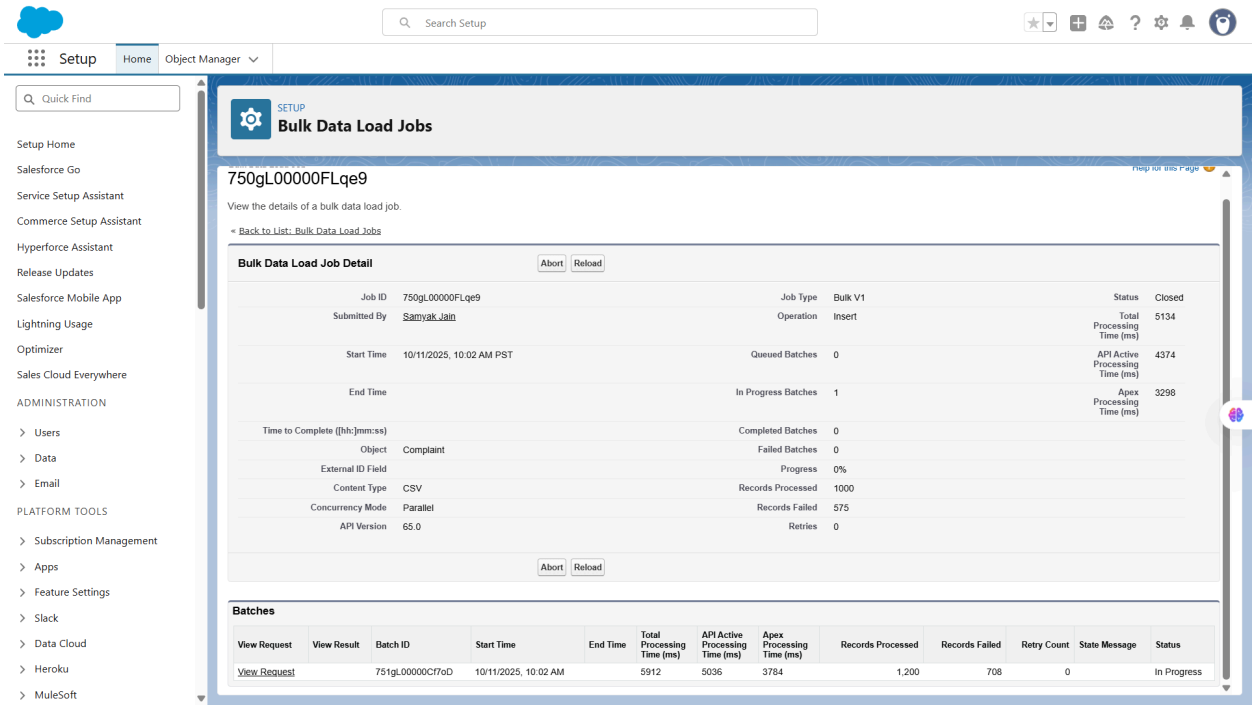
📌Phase-8: Data Management & Deployment

🔹 1. Data Import Wizard

Use Case:
The Data Import Wizard allows admins to **easily import small to medium-sized datasets** (up to 50,000 records) into Salesforce without using any external tools. It's ideal for uploading complaints, customers, or feedback records from spreadsheets into custom objects.

Implementation Steps:

1. Go to **Setup → Data Import Wizard**.
2. Click **Launch Wizard**.
3. Choose the object (e.g., `Complaint__c` or `Customer__c`).
4. Upload a `.CSV` file containing your data.
5. Map CSV columns to Salesforce fields.
6. Click **Start Import** and monitor progress under **Bulk Data Load Jobs**.

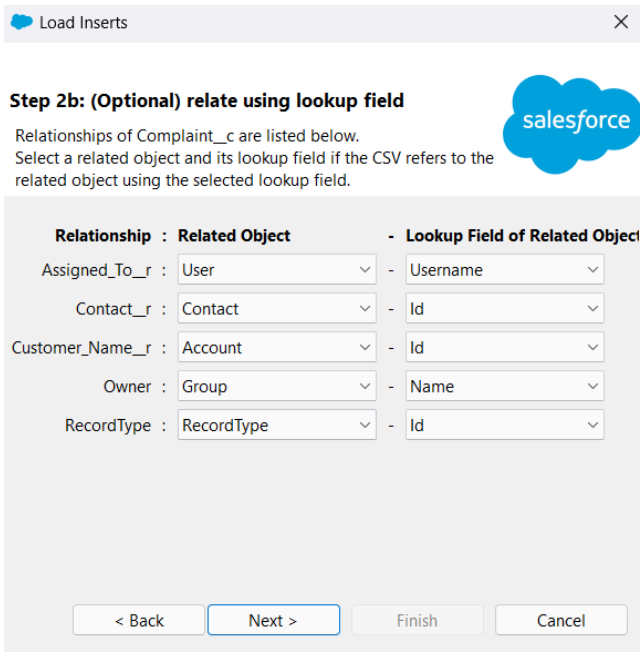


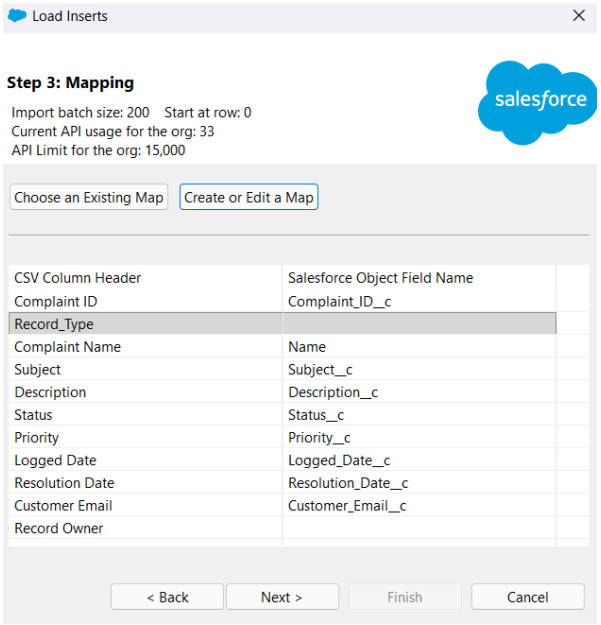
🔹 2. Data Loader

Use Case:
The Data Loader is used for **large-scale data operations** (up to 5 million records). It's perfect for admins or developers who need to **insert, update, upsert, delete, or export** records using CSV files.

Implementation Steps:

1. Install **Salesforce Data Loader** from Setup or Salesforce Help.
2. Log in using your Salesforce credentials (use **Security Token** if required).
3. Select the operation (Insert / Update / Upsert / Delete / Export).
4. Choose the target object and CSV file.
5. Map the fields and execute.
6. Review **Success** and **Error** logs after completion.



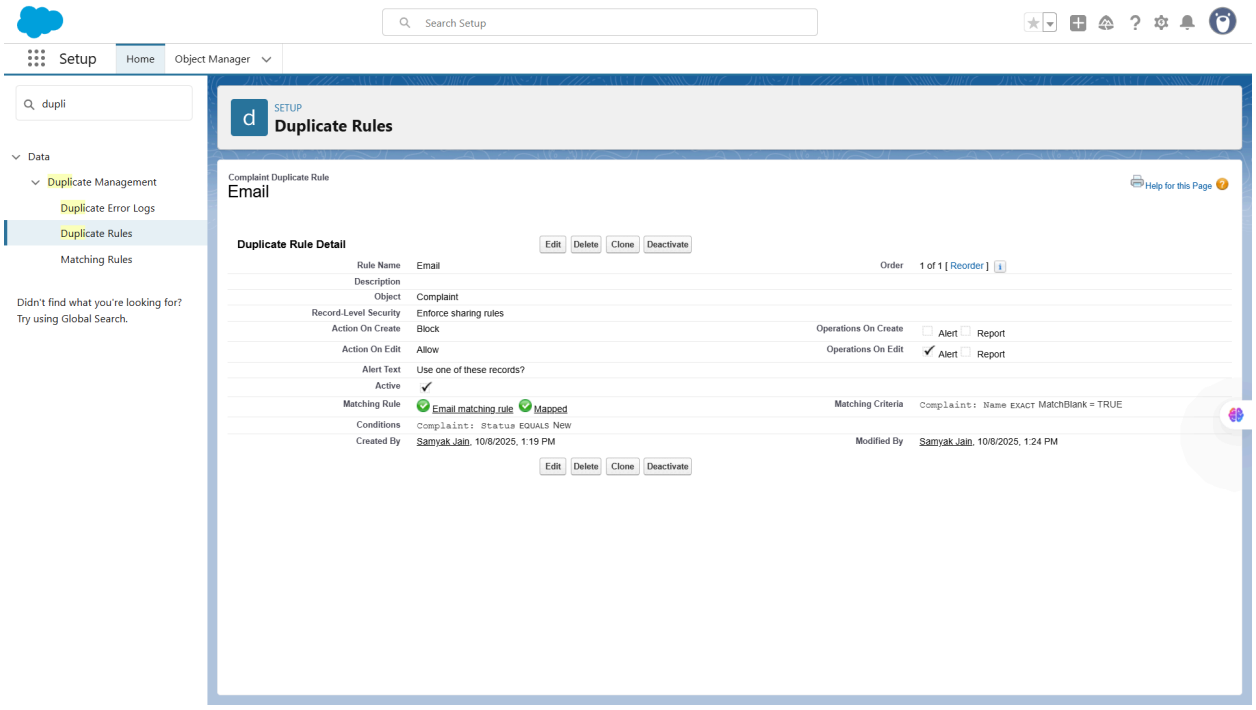


🔹 3. Duplicate Rules

Use Case:
Duplicate Rules help prevent **data redundancy** by identifying and blocking or alerting users when duplicate records are created — for example, when two complaints are logged for the same customer.

Implementation Steps:

1. Go to **Setup** → **Duplicate Rules**.
2. Click **New Rule** → select an object (e.g., `Complaint__c`).
3. Choose a **Matching Rule** (e.g., match by Email or Phone).
4. Set the **Action**: Allow / Block / Alert.
5. Activate the rule and test by creating duplicate data.

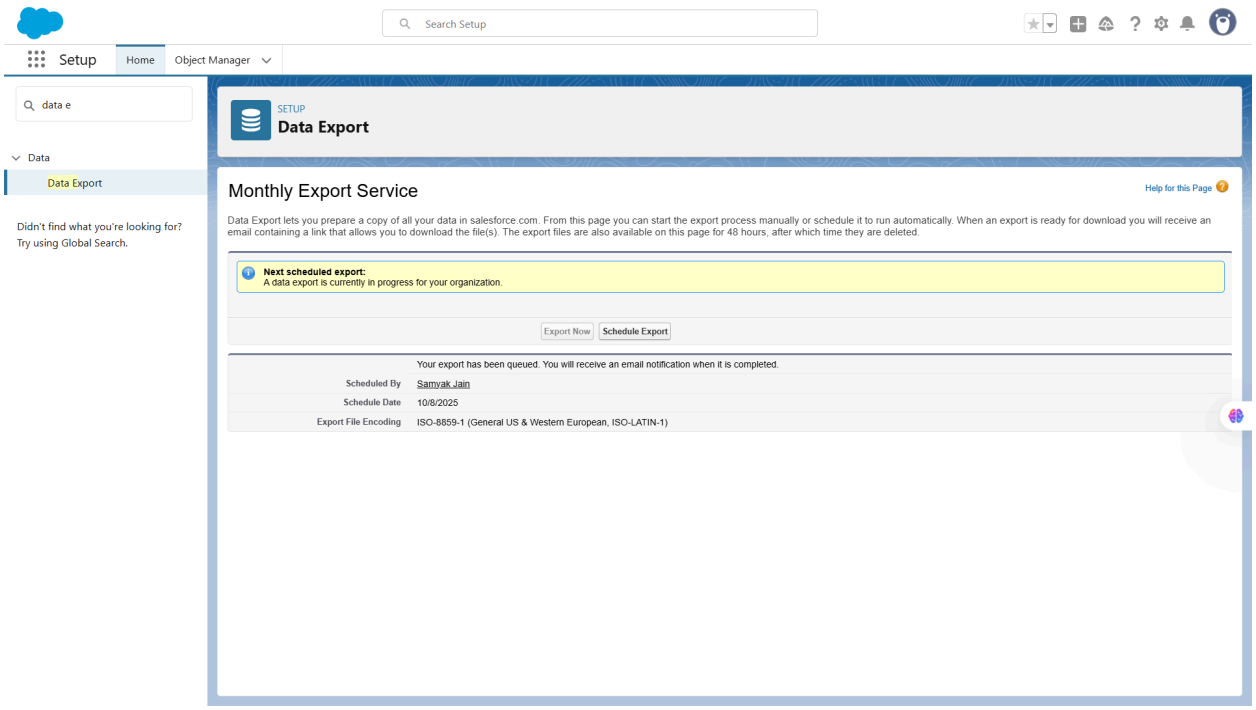


🔹 4. Data Export & Backup

Use Case:
Data Export ensures **regular backups of Salesforce data**, either manually or automatically, to prevent data loss. You can export all records, attachments, and files into `.zip` archives.

Implementation Steps:

1. Navigate to **Setup** → **Data Export**.
2. Choose **Export Now** or schedule **weekly/monthly exports**.
3. Select the objects (e.g., Complaints, Customers, Feedback).
4. Salesforce generates a downloadable `.zip` file.



🔹 5. Change Sets

Use Case:
Change Sets allow admins to **deploy customizations** (like objects, fields, validation rules, flows, LWCs, etc.) between **related orgs** (e.g., Sandbox → Production) without coding.

Change Sets are normally used to deploy components between Salesforce orgs. Since my org does not have outbound change sets enabled, I used VS Code to deploy my project components to the target org.

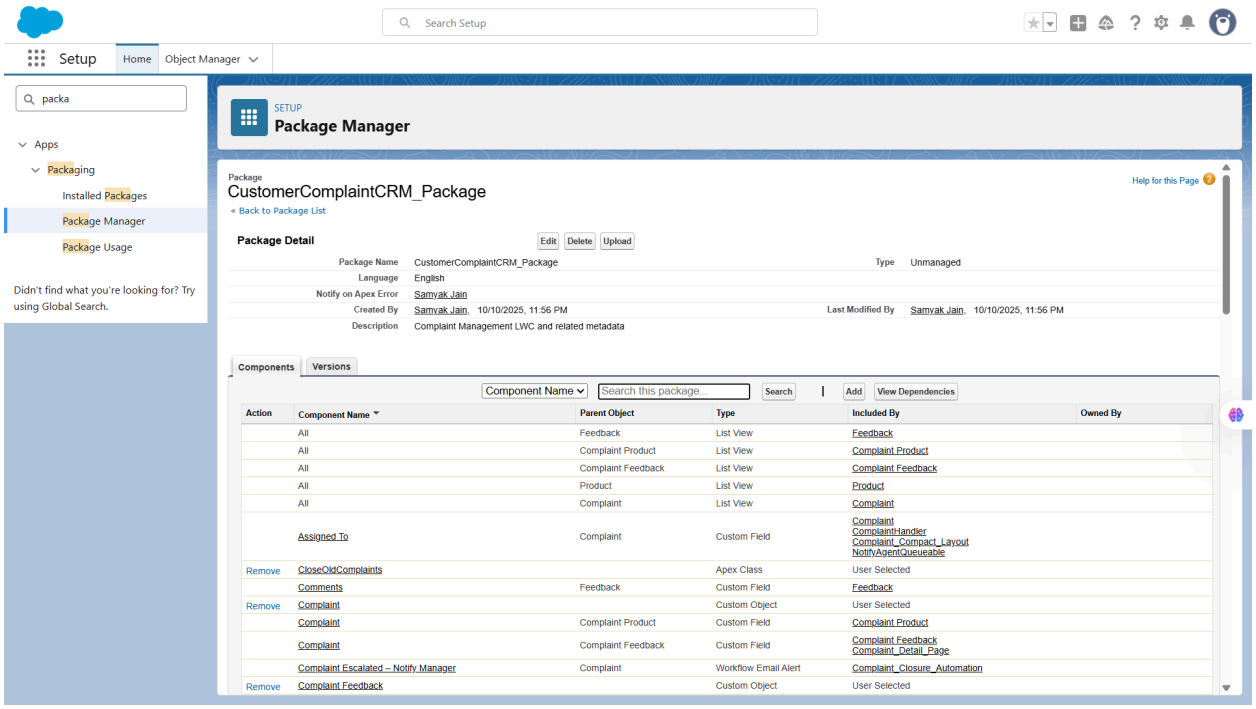
🔹 6. Unmanaged vs Managed Packages

Use Case:
Salesforce Packages group metadata and components for distribution.

- **Managed Packages** are for **commercial apps** (protected, upgradable).
- **Unmanaged Packages** are for **open or internal sharing** (editable).

Implementation Steps:

1. In **Setup → Package Manager**, create a new package.
2. Add components (objects, fields, LWCs, Apex).
3. Choose **Managed** (for AppExchange apps) or **Unmanaged** (for internal projects).

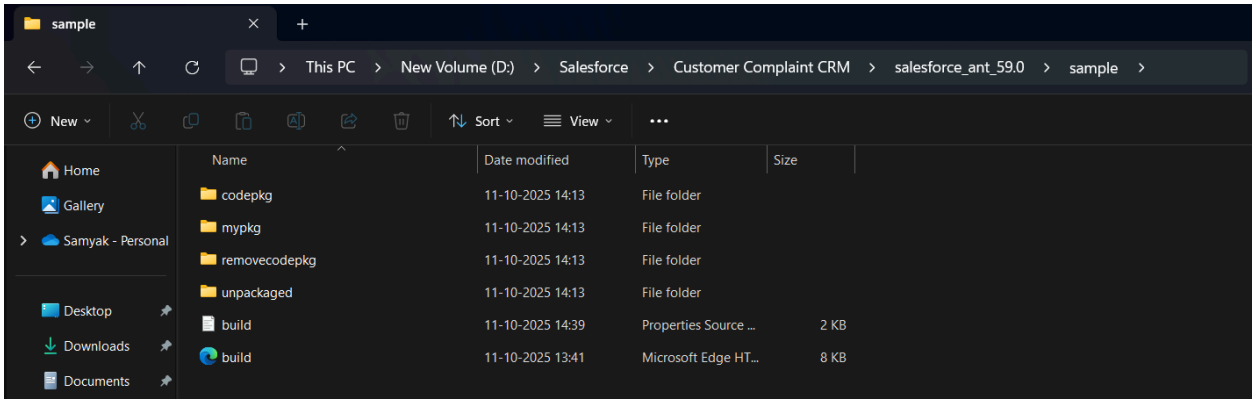


🔹 7. ANT Migration Tool

Use Case:
The **ANT Migration Tool** is a command-line utility for **automated deployments** between Salesforce orgs — best suited for developers managing large projects with version control.

Implementation Steps:

1. Install **Apache ANT** and **Salesforce Migration Tool (zip)**.
2. Create `build.xml` and `package.xml` to define metadata.
3. Run commands like `ant retrieve` and `ant deploy`.



🔹 8. VS Code & SFDX

Use Case:
Salesforce DX (SFDX) and **VS Code** integration streamline **development and deployment** using modern CLI-based tools. It's ideal for source-driven development, version control, and CI/CD automation.

Implementation Steps:

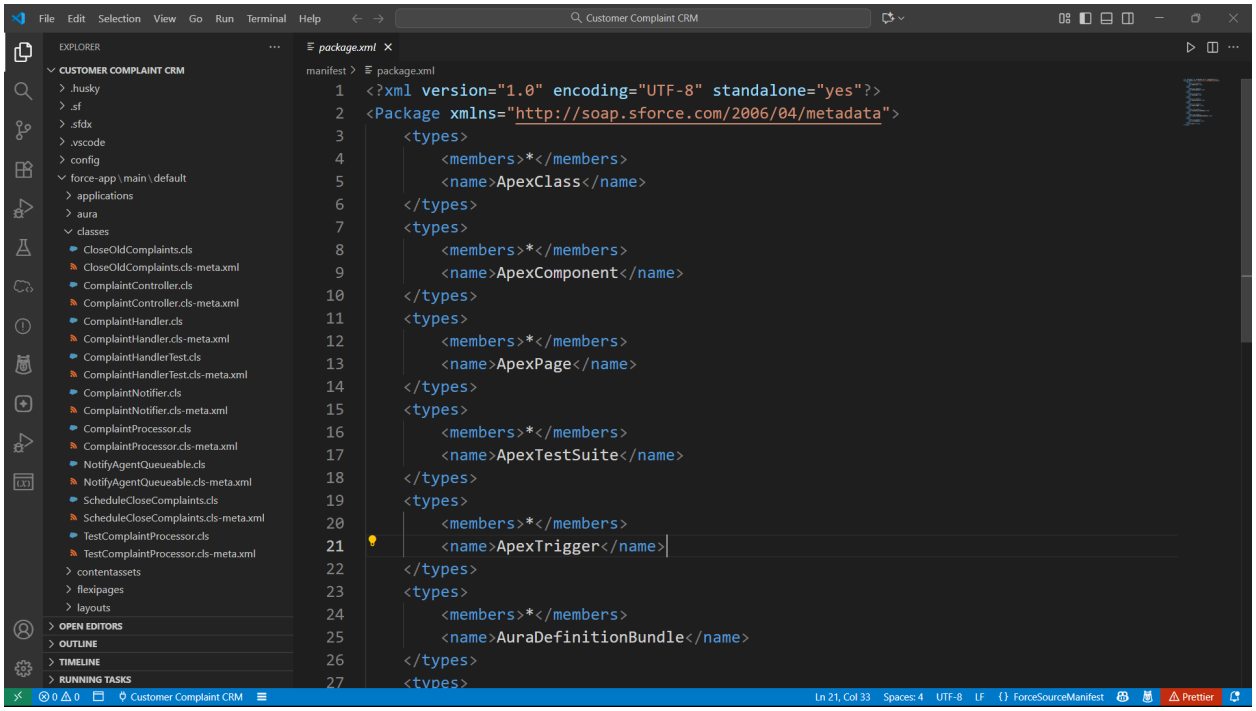
1. Install **VS Code** and **Salesforce Extensions Pack**.

   Connect to your org using: sfdx auth:web:login -a DevHub

2. Create or open an SFDX project:
   sfdx force:project:create -n ComplaintCRM

3. Retrieve or deploy metadata:

   sfdx force:source:retrieve -m ApexClass sfdx force:source:deploy -m LightningComponentBundle



📌Phase 9: Reporting, Dashboards & Security Review

🔹 1. Reports

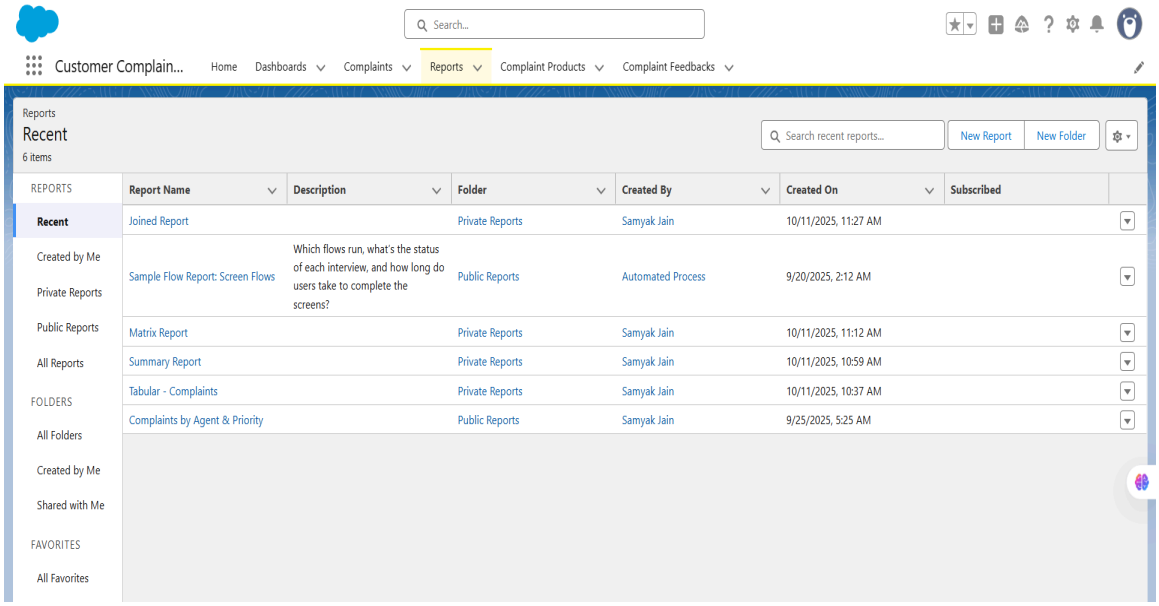
🔐 Goal:

Reports allow Salesforce users to analyze, organize, and visualize data stored in the system. Each report type serves a specific purpose:

- **Tabular Reports:** Provide a simple list of records and are useful for exports or quick data views.
- **Summary Reports:** Group records by a field and show subtotals, enabling managers to see counts or sums by category.
- **Matrix Reports:** Allow grouping of data by rows and columns simultaneously, ideal for cross-tab analysis, like complaints by status vs. assigned agent.
- **Joined Reports:** Combine multiple report blocks from related or unrelated objects, useful for comparing different datasets side-by-side, e.g., complaints and their feedback.

Reports are the foundation for informed decision-making, operational oversight, and team performance tracking.

Use Case:
Reports help customer service managers analyze complaint trends, agent performance, and resolution times. For example, a “Complaints by Status” report shows how many complaints are New, In Progress, or Resolved.



🔹 2. Report Types

🔐 Goal:

Report Types define **which objects and fields are available** when creating reports. They act as templates that determine what data can be accessed and how objects are related.

- Standard report types are provided by Salesforce for common objects.
- Custom report types can be created to include related custom objects or specific fields not available in standard types.

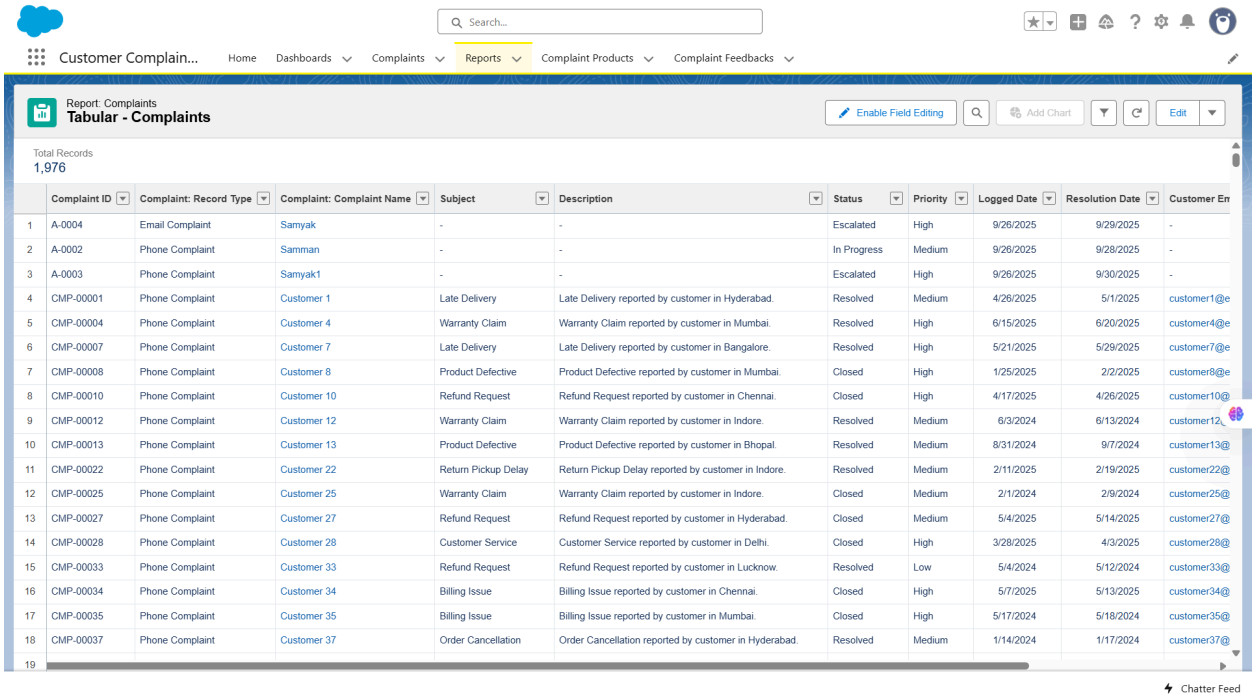
Report Types ensure that users can build meaningful reports while maintaining data relevance and integrity.

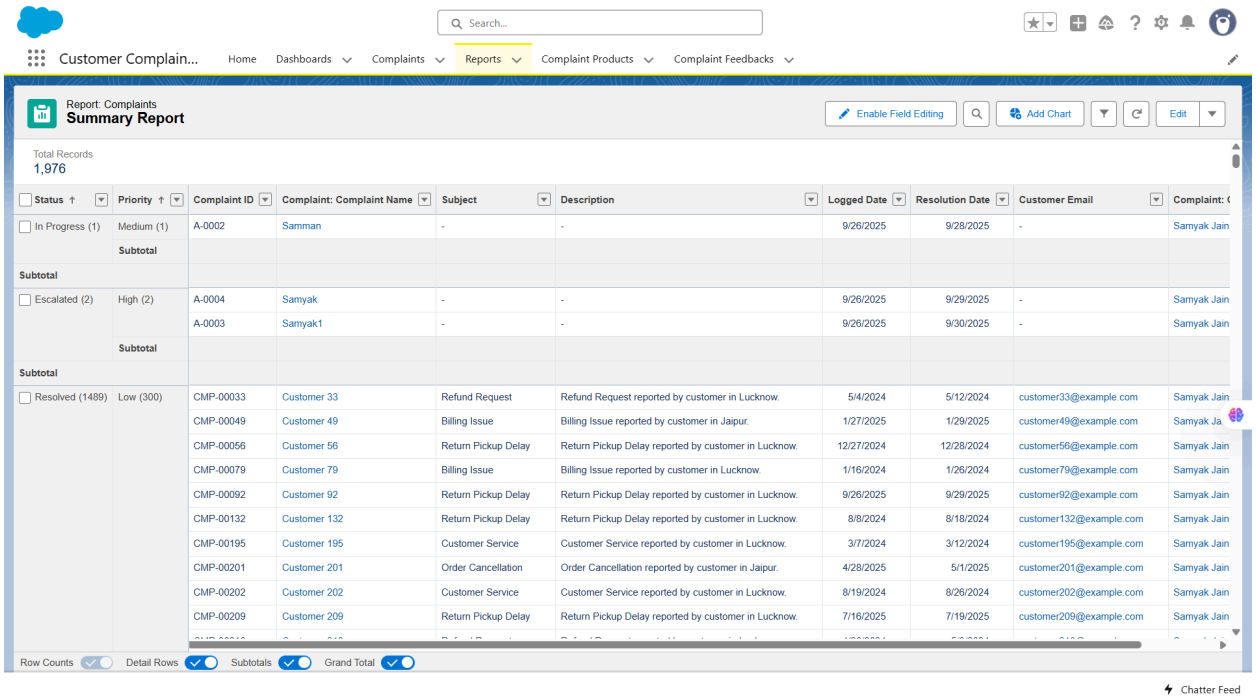
Use Case:
Defines which objects and relationships are available for reporting.
Example: To combine “Complaints” and “Feedback,” you need a **custom report type**.

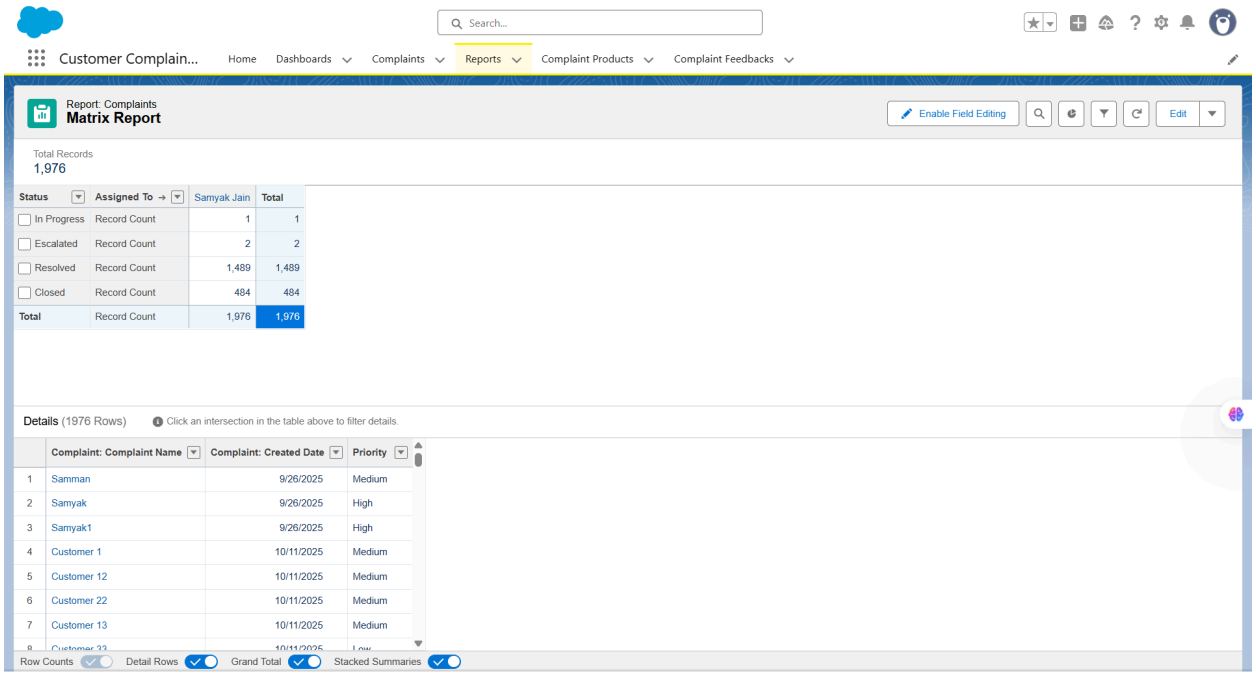
Types:

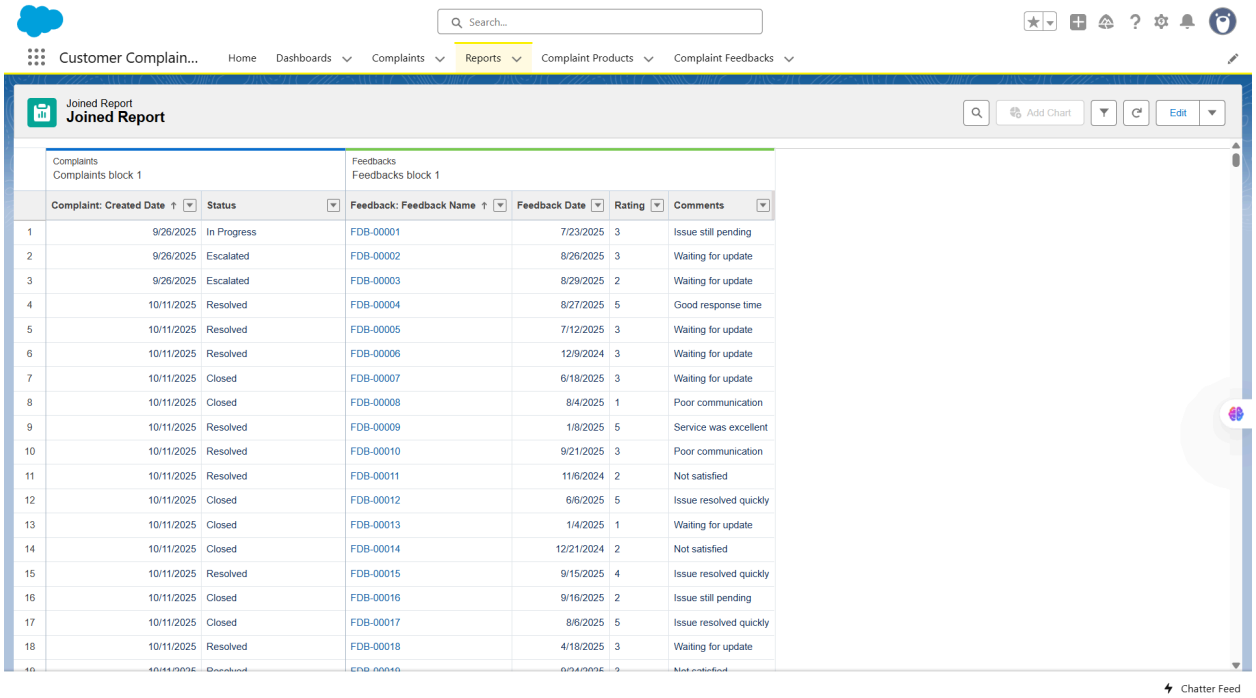
- **Tabular:** Simple list of records (e.g., all complaints created this week).
- **Summary:** Groups records by field (e.g., complaints grouped by Status or Priority).

- **Matrix:** Groups both rows and columns (e.g., complaints by Status *and* Agent).
- **Joined:** Combines multiple report types (e.g., Complaints + Feedback).









🔹 3. Dashboards

🔐 Goal:

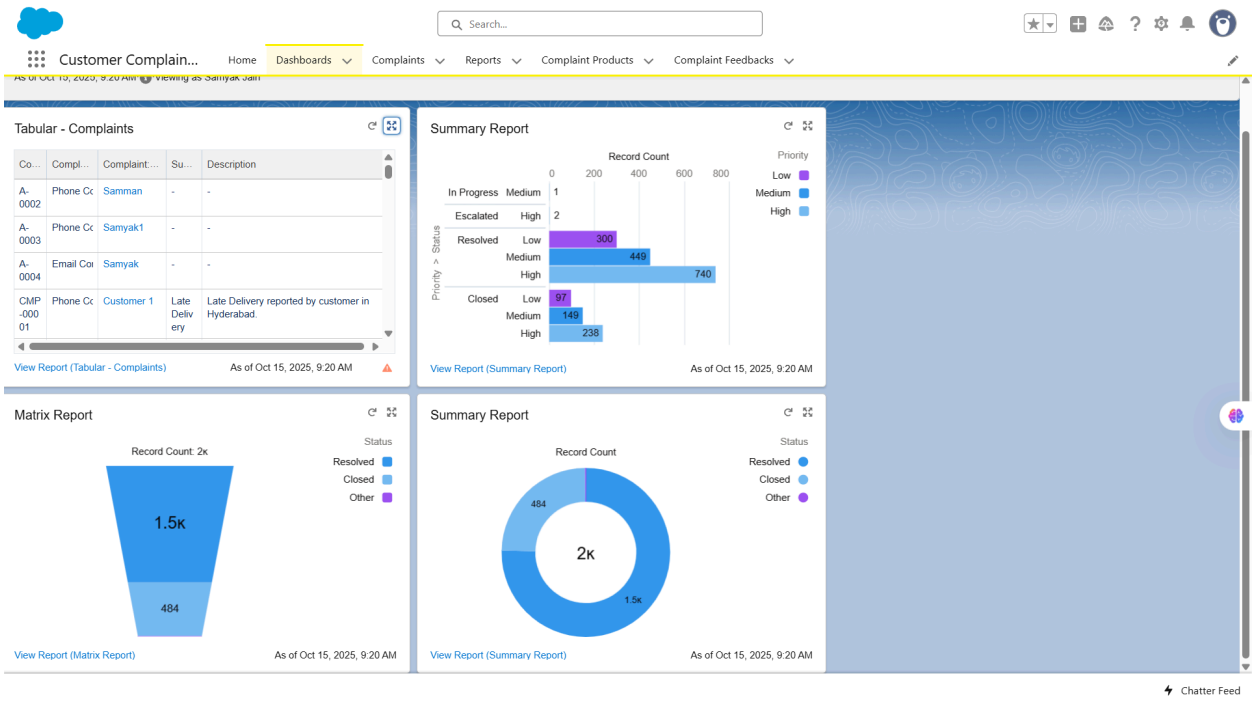
Dashboards provide a **visual representation of report data** using charts, tables, and metrics in one consolidated view.

- They allow managers and executives to monitor KPIs, team performance, and operational metrics at a glance.
- Dashboards are composed of **dashboard components** (e.g., pie charts, bar charts, tables) that are linked to underlying reports.

Dashboards improve data-driven decision-making and enable real-time tracking of business processes.

Use Case:
Dashboards visualize key metrics from multiple reports. For example, a **Complaint Management Dashboard** can show:

- Tablar Complaint (Lightning Table)
- Summary Report (Horizontal Bar Chart)
- Matrix Report (Funnel Chart)
- Summary Report (Donut Chart)



🔹 4. Dynamic Dashboards

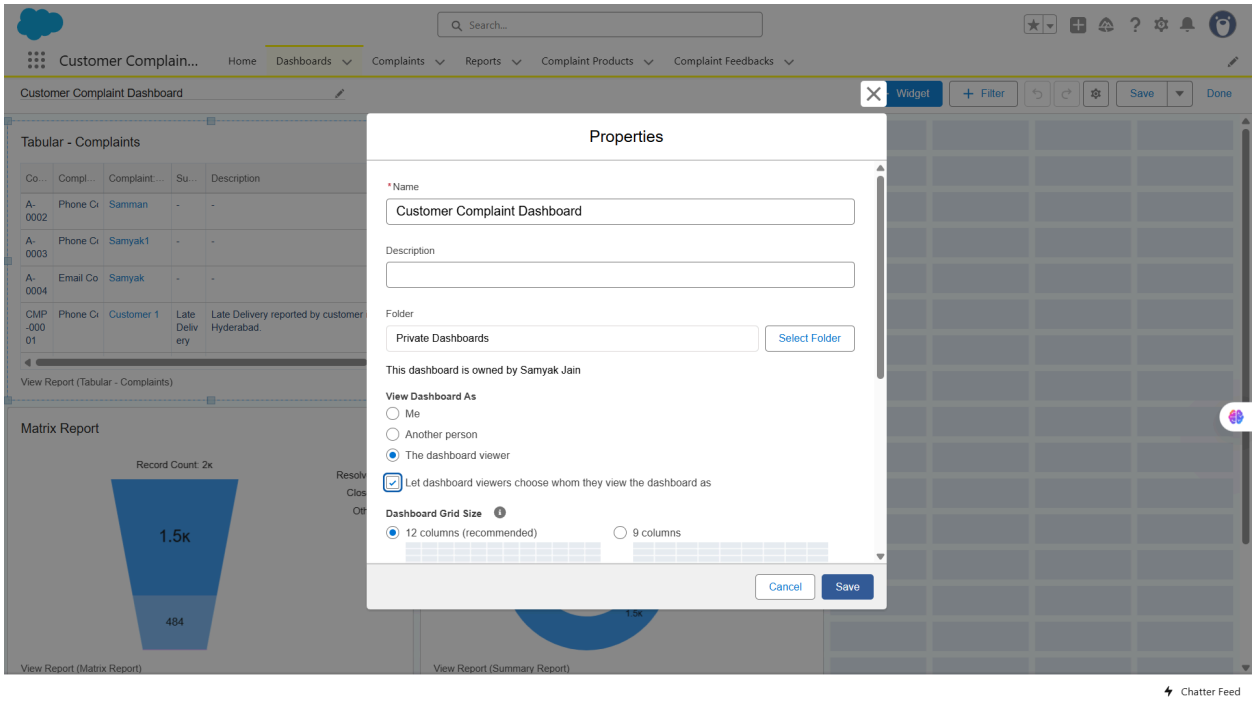
🔐 Goal:

Dynamic Dashboards allow users to view the **same dashboard with data filtered according to their own access level**.

- Unlike standard dashboards, which display data based on a fixed running user, dynamic dashboards adapt to the viewer's permissions and role hierarchy.
- This is especially useful for managers and agents who need personalized visibility without creating multiple copies of the same dashboard.

Dynamic dashboards enhance security and usability while providing tailored insights for each user.

Use Case:
Allows users to view dashboard data **as themselves** (based on their permissions), without needing multiple copies for each user.
Example: Each agent can see *only their own complaints* in the same dashboard.



🔹 5. Profiles

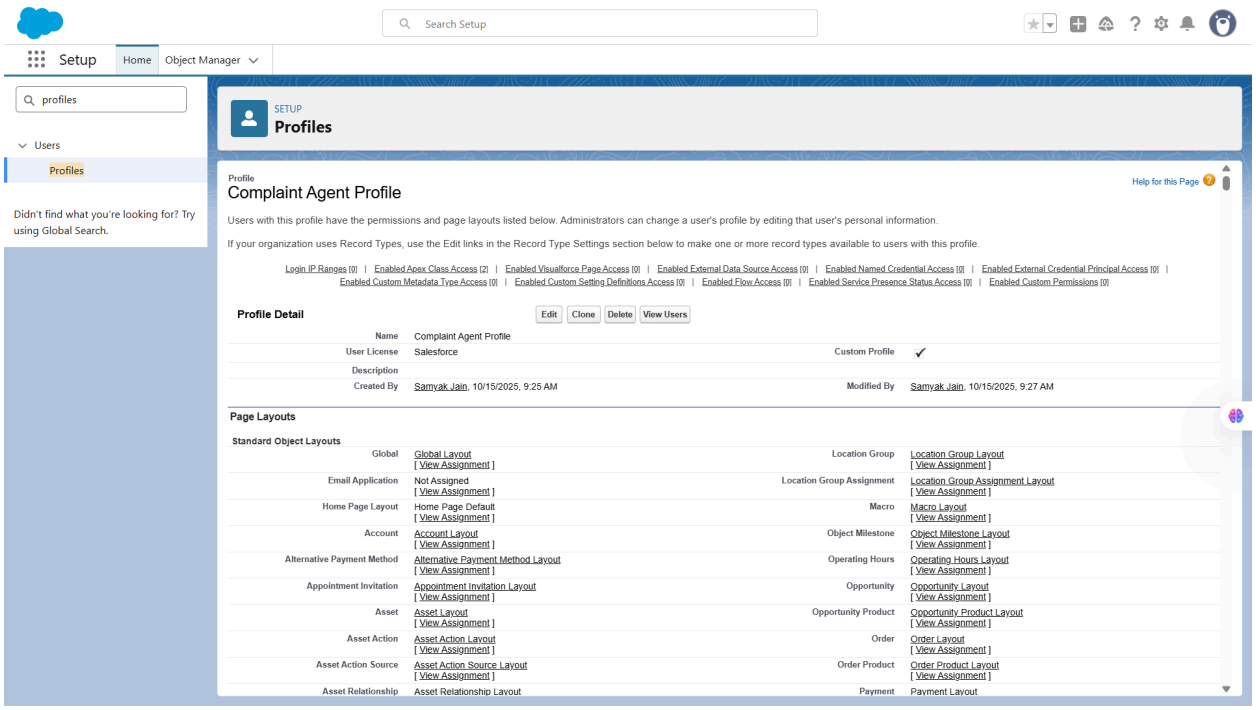
🔐 Goal:

Assign the custom Complaint Agent Profile (or any profile) to your Support Agent users so they get the correct permissions.

Use Case:

Profiles define **base-level access** — which objects, fields, tabs, and apps a user can see or edit.

Example: Agents can edit Complaints but not modify Feedback.



🔹 6. Roles

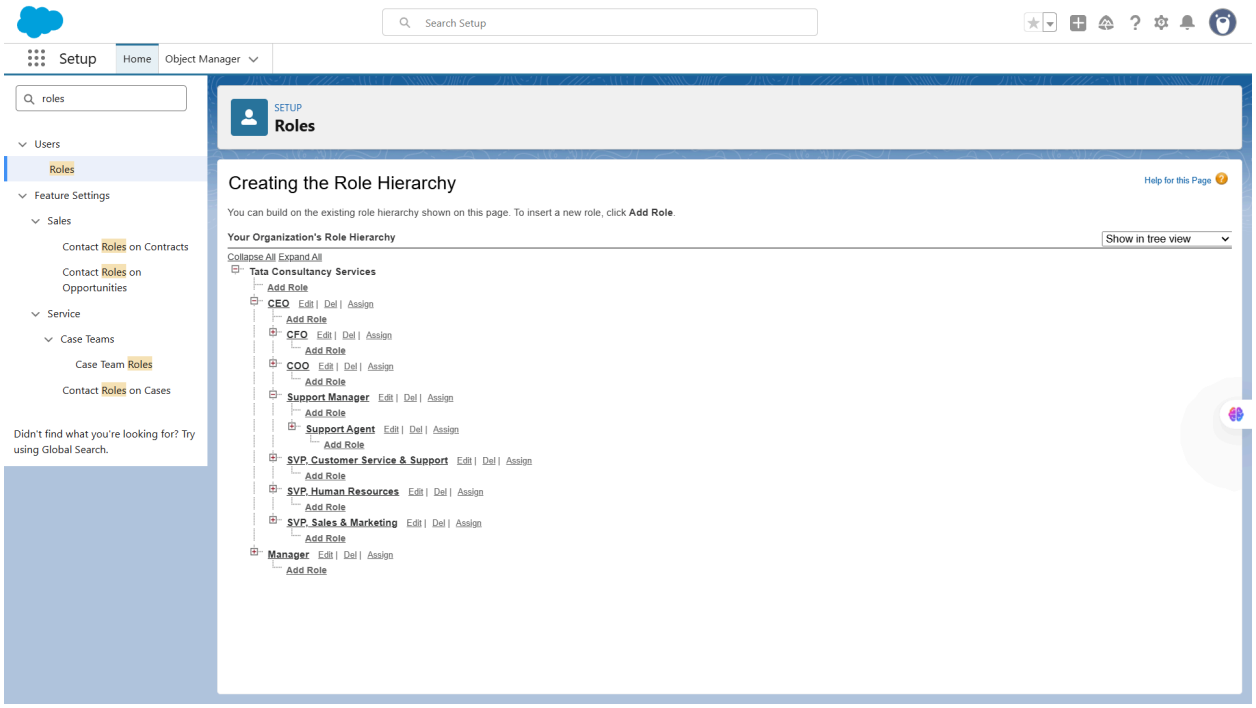
🔐 Goal:

Create a clear role hierarchy so managers can see the records owned by users below them (for example, Support Managers can view all Support Agents' complaints).

Use Case:

Roles determine **record visibility** in the role hierarchy (who can see whose records).

Example: Support Manager can see complaints of all agents under them.



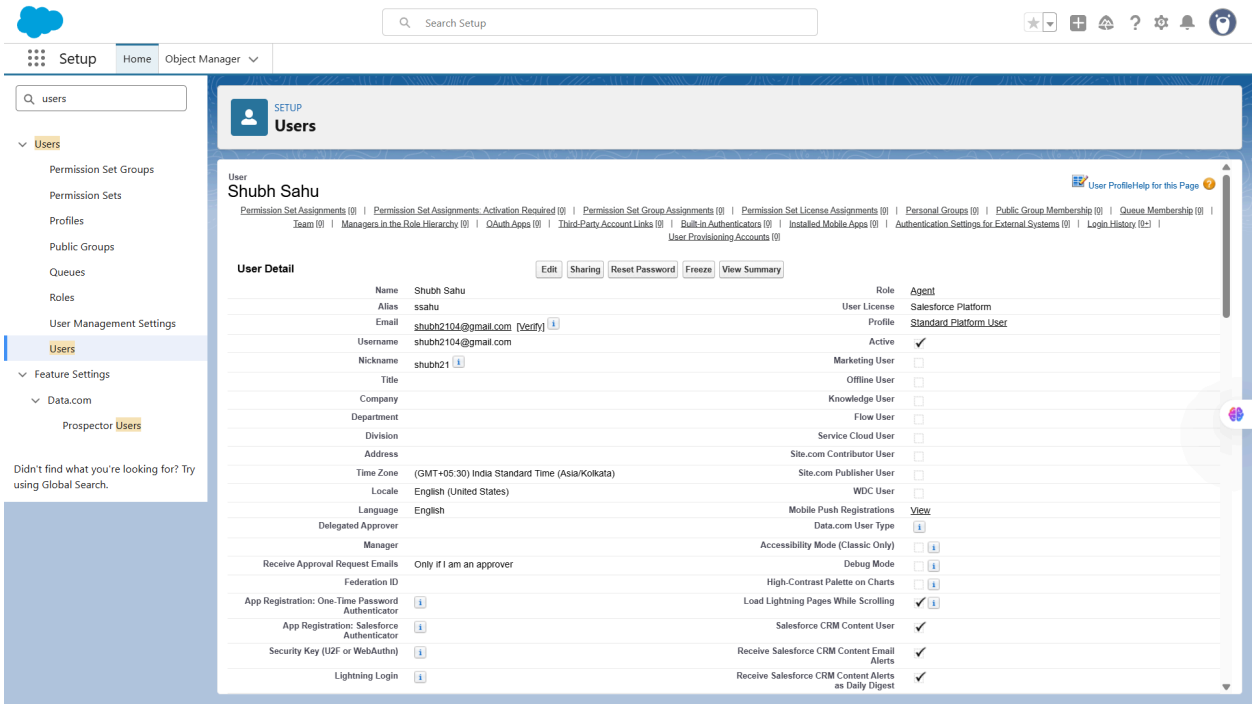
🔹 7. Users

🔐 Goal:

Create new Salesforce users (e.g., Support Agents or Managers) and assign them their Profile (permissions) and Role (visibility level).

Use Case:

Each user has login credentials and is assigned a profile & role.
Example: Create users for each agent or admin.



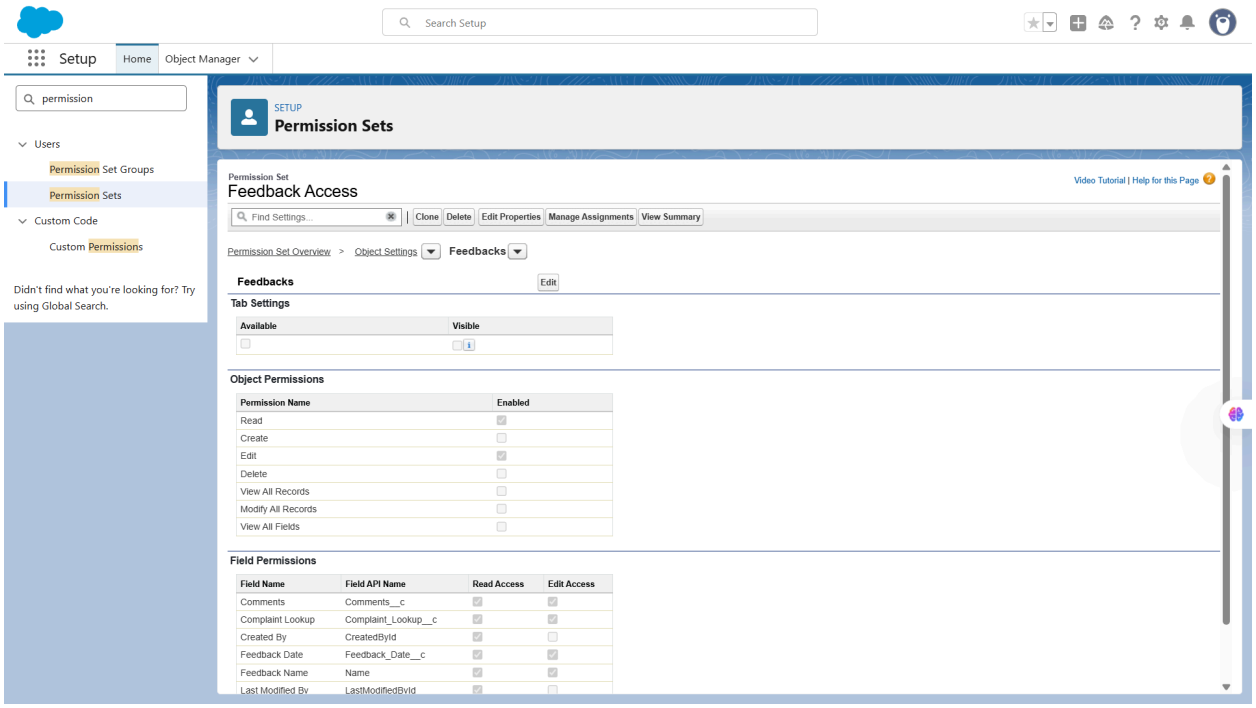
🔹 8. Permission Sets

🔐 Goal:

Create a Permission Set that grants extra access (e.g., Read/Edit on Feedback__c) without modifying the user's profile.

Use Case:

Provides **additional permissions** without changing a user's profile.
Example: Temporarily grant a Support Agent access to the Feedback object.

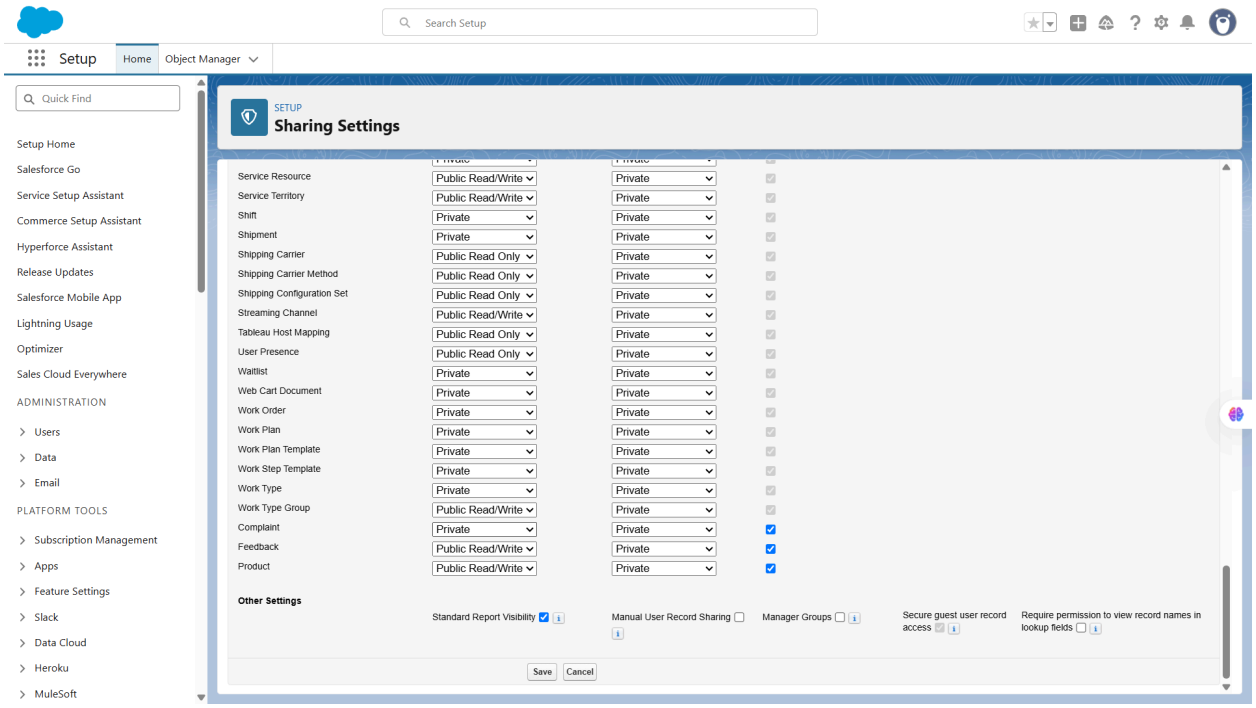


🔹 9. OWD (Organization-Wide Defaults)

🔐 Goal

Restrict access to Complaint records so that only the record owner and their managers can view or edit them (others cannot).

Use Case:
Controls **baseline record access** for all users.
Example: Complaints = Private (only owner and superiors can see).



🔹 10. Sharing Rules

Note:
In this project, a **Region** field was **not included** in the Complaint__c object, as the business process does not require regional segregation of data. All complaints are handled centrally by the customer service team, regardless of geographic or departmental divisions.

Use Case:
Since the **Region__c** field is not present, the sharing model is based on **user roles** instead of data criteria.
To ensure that managers can monitor their team's complaints while maintaining privacy across departments, sharing rules were configured to share complaint records **owned by users in the "Customer Support Agent" role** with users in the **"Customer Service Manager" role**.

Summary:

- **Region Field:** Not added (centralized complaint management system)
- **Rule Type:** Based on record owner
- **Share From:** Role = Customer Support Agent

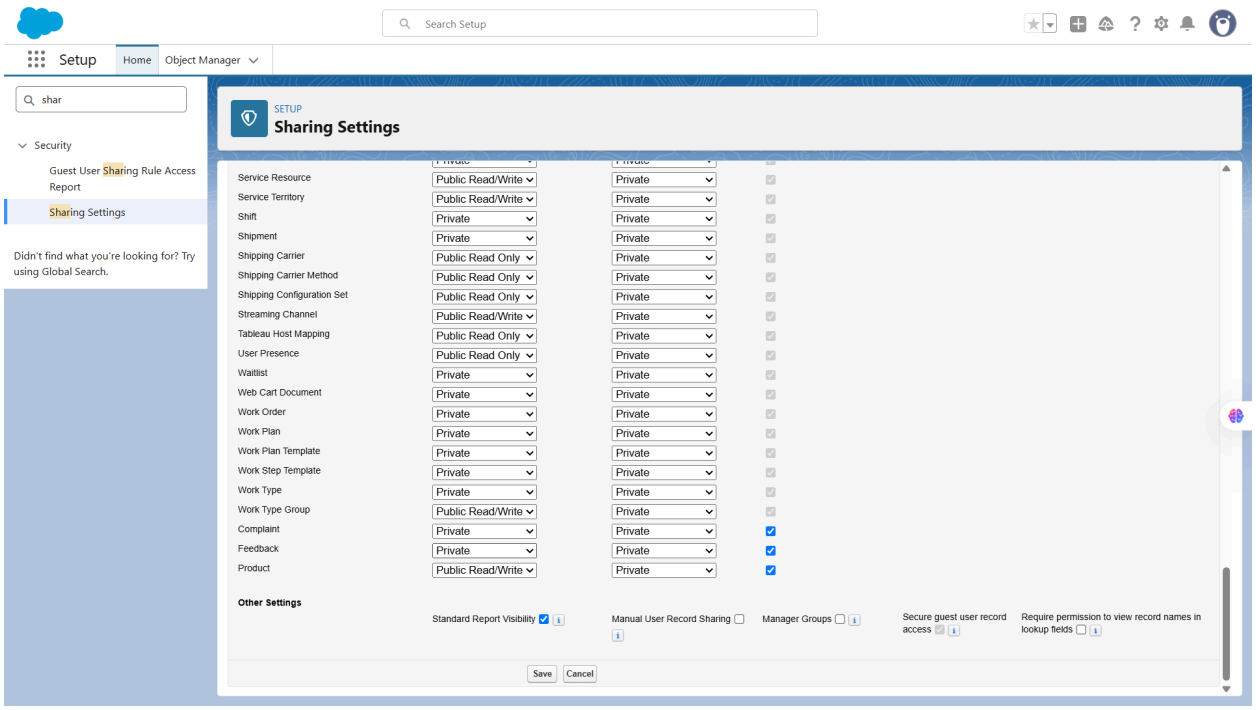
- **Share With:** Role = Customer Service Manager
- **Access Level:** Read/Write
- **Purpose:** Provides team-level visibility without needing regional data segmentation.

🔹 11. Sharing Settings

🔐 Goal

To review and set up Organization-Wide Defaults (OWD) for your project's custom objects — mainly Complaint__c and Feedback__c — so that only authorized users can access the records.

Use Case:
Central control panel for all sharing behaviors — OWD, Sharing Rules, and Manual Sharing.

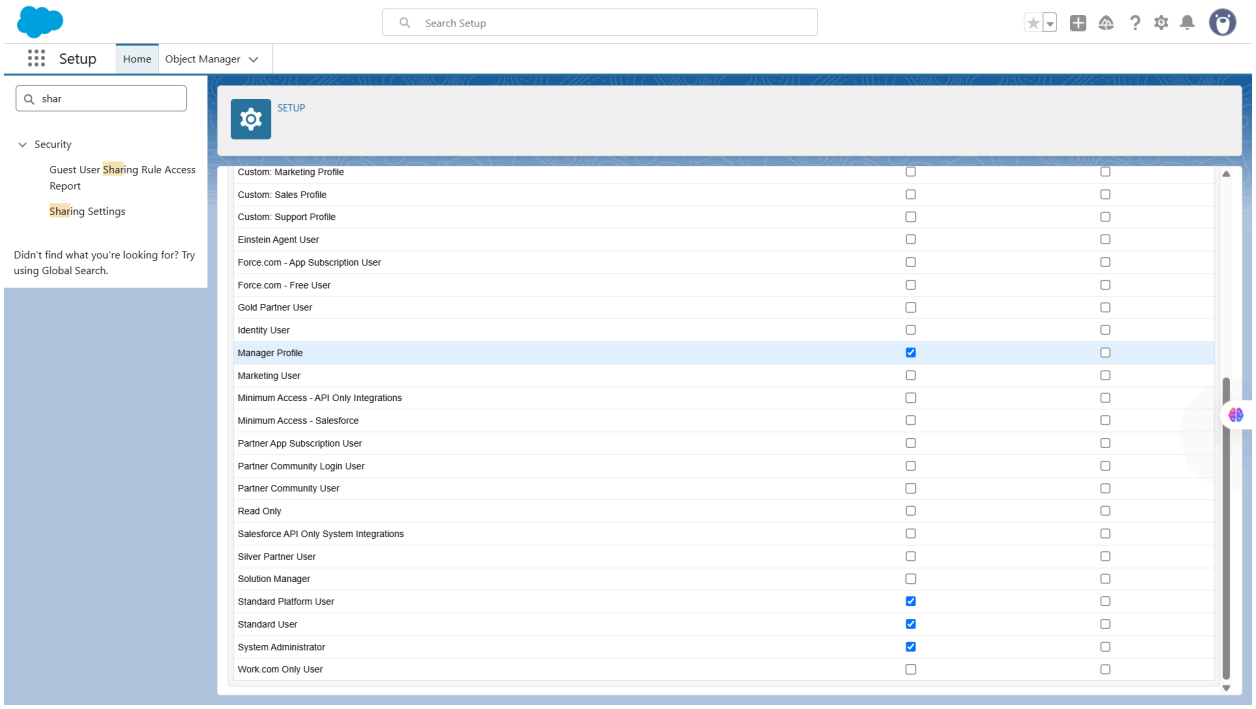


🔹 12. Field Level Security (FLS)

🔐 Goal

Restrict access to sensitive fields so only certain profiles (like Managers) can view or edit them.

Use Case:
Restricts visibility/editing of specific fields.
Example: “Customer Email” field is visible only to Managers.

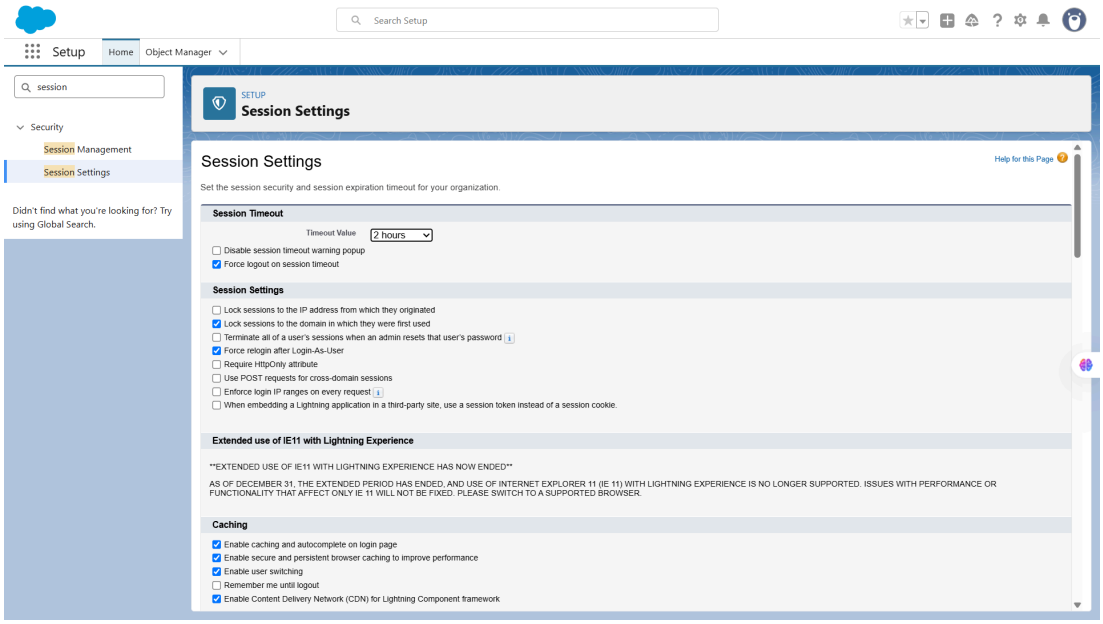


🔹 13. Session Settings

🔐 Goal

Control user session behavior, including timeouts, security policies, and login restrictions to improve org security.

Use Case:
Enhances login security — timeout duration, trusted IPs, browser security.

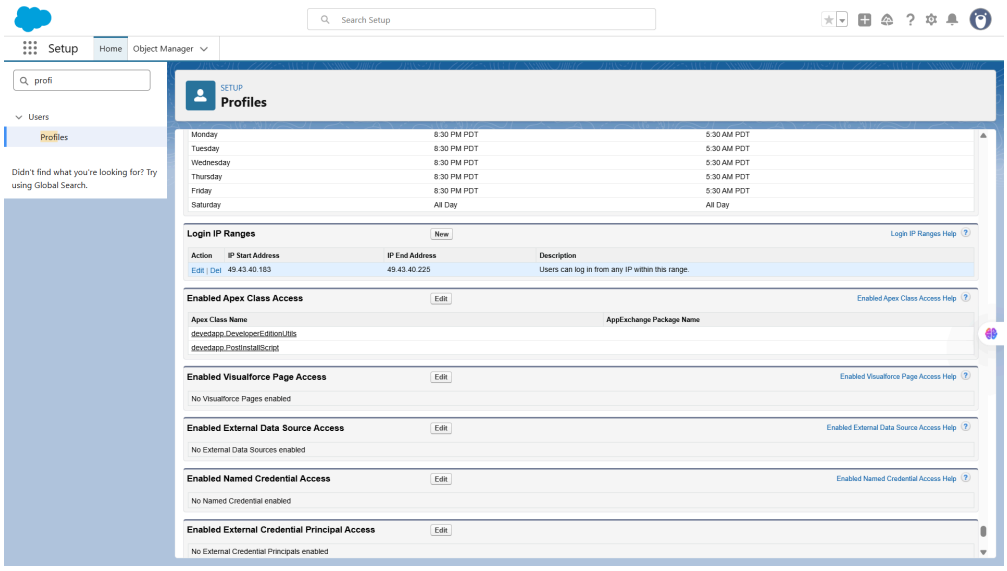


🔹 14. Login IP Ranges

🔐 Goal

Control user session behavior, including timeouts, security policies, and login restrictions to improve org security.

Use Case:
Restricts user logins to company IPs.
Example: Agents can log in only from office network.

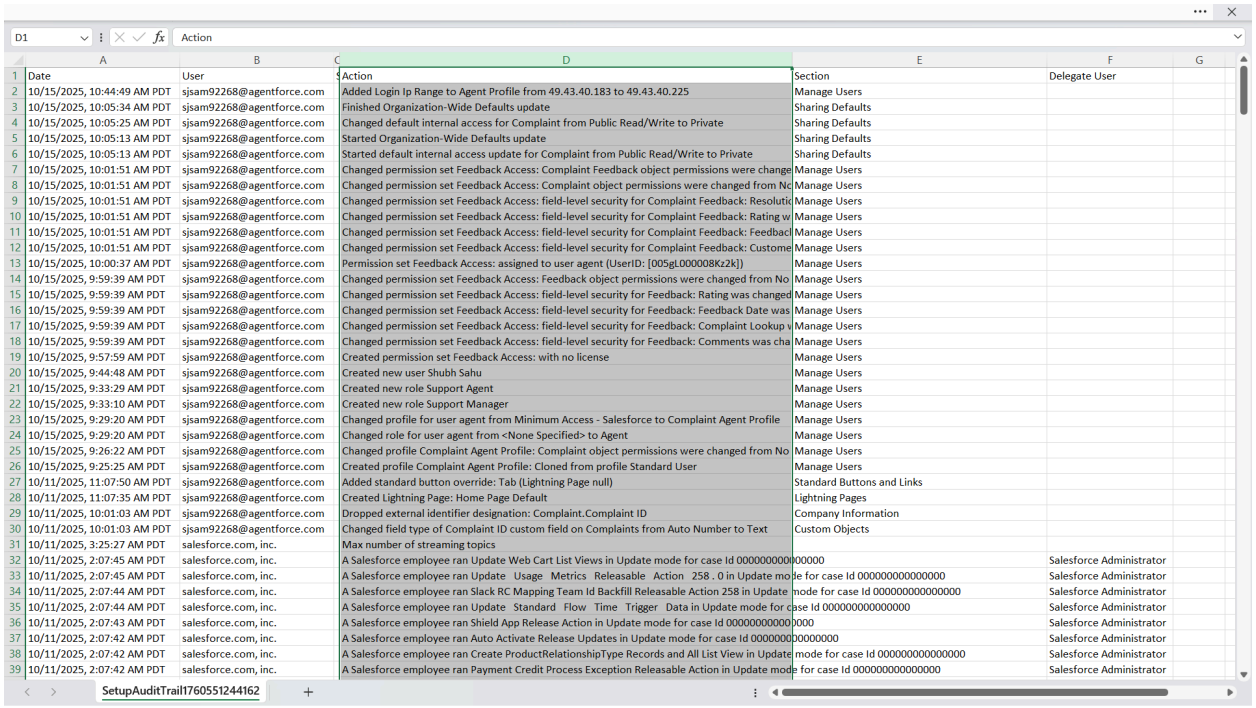


🔹 15. Audit Trail

🔐 Goal

Restrict Salesforce logins so that users can only access the org from approved IP addresses, enhancing security for sensitive data.

Use Case:
The Setup Audit Trail was reviewed to monitor administrative changes in the Salesforce org. It tracks all modifications to profiles, roles, sharing settings, objects, and fields, providing a secure log for accountability and compliance. Audit trail downloads were retained for historical reference and audit purposes.



D1 | Action

	A	B	C	D	E	F	G
1	Date	User	S	Action	Section	Delegate User	
2	10/15/2025, 10:44:49 AM PDT	sjsam92268@agentforce.com		Added Login Ip Range to Agent Profile from 49.43.40.183 to 49.43.40.225	Manage Users		
3	10/15/2025, 10:05:34 AM PDT	sjsam92268@agentforce.com		Finished Organization-Wide Defaults update	Sharing Defaults		
4	10/15/2025, 10:05:25 AM PDT	sjsam92268@agentforce.com		Changed default internal access for Complaint from Public Read/Write to Private	Sharing Defaults		
5	10/15/2025, 10:05:13 AM PDT	sjsam92268@agentforce.com		Started Organization-Wide Defaults update	Sharing Defaults		
6	10/15/2025, 10:05:13 AM PDT	sjsam92268@agentforce.com		Started default internal access update for Complaint from Public Read/Write to Private	Sharing Defaults		
7	10/15/2025, 10:01:51 AM PDT	sjsam92268@agentforce.com		Changed permission set Feedback Access: Complaint Feedback object permissions were change	Manage Users		
8	10/15/2025, 10:01:51 AM PDT	sjsam92268@agentforce.com		Changed permission set Feedback Access: Complaint object permissions were changed from Nc	Manage Users		
9	10/15/2025, 10:01:51 AM PDT	sjsam92268@agentforce.com		Changed permission set Feedback Access: field-level security for Complaint Feedback: Resolutic	Manage Users		
10	10/15/2025, 10:01:51 AM PDT	sjsam92268@agentforce.com		Changed permission set Feedback Access: field-level security for Complaint Feedback: Rating w	Manage Users		
11	10/15/2025, 10:01:51 AM PDT	sjsam92268@agentforce.com		Changed permission set Feedback Access: field-level security for Complaint Feedback: Feedbacl	Manage Users		
12	10/15/2025, 10:01:51 AM PDT	sjsam92268@agentforce.com		Changed permission set Feedback Access: field-level security for Complaint Feedback: Custome	Manage Users		
13	10/15/2025, 10:00:37 AM PDT	sjsam92268@agentforce.com		Permission set Feedback Access: assigned to user agent (UserID: [005gL000008Kz2k])	Manage Users		
14	10/15/2025, 9:59:39 AM PDT	sjsam92268@agentforce.com		Changed permission set Feedback Access: Feedback object permissions were changed from No	Manage Users		
15	10/15/2025, 9:59:39 AM PDT	sjsam92268@agentforce.com		Changed permission set Feedback Access: field-level security for Feedback: Rating was changed	Manage Users		
16	10/15/2025, 9:59:39 AM PDT	sjsam92268@agentforce.com		Changed permission set Feedback Access: field-level security for Feedback: Feedback Date was	Manage Users		
17	10/15/2025, 9:59:39 AM PDT	sjsam92268@agentforce.com		Changed permission set Feedback Access: field-level security for Feedback: Complaint Lookup v	Manage Users		
18	10/15/2025, 9:59:39 AM PDT	sjsam92268@agentforce.com		Changed permission set Feedback Access: field-level security for Feedback: Comments was cha	Manage Users		
19	10/15/2025, 9:57:59 AM PDT	sjsam92268@agentforce.com		Created permission set Feedback Access: with no license	Manage Users		
20	10/15/2025, 9:44:48 AM PDT	sjsam92268@agentforce.com		Created new user Shubh Sahu	Manage Users		
21	10/15/2025, 9:33:29 AM PDT	sjsam92268@agentforce.com		Created new role Support Agent	Manage Users		
22	10/15/2025, 9:33:10 AM PDT	sjsam92268@agentforce.com		Created new role Support Manager	Manage Users		
23	10/15/2025, 9:29:20 AM PDT	sjsam92268@agentforce.com		Changed profile for user agent from Minimum Access - Salesforce to Complaint Agent Profile	Manage Users		
24	10/15/2025, 9:29:20 AM PDT	sjsam92268@agentforce.com		Changed role for user agent from <None Specified> to Agent	Manage Users		
25	10/15/2025, 9:26:22 AM PDT	sjsam92268@agentforce.com		Changed profile Complaint Agent Profile: Complaint object permissions were changed from No	Manage Users		
26	10/15/2025, 9:25:25 AM PDT	sjsam92268@agentforce.com		Created profile Complaint Agent Profile: Cloned from profile Standard User	Manage Users		
27	10/11/2025, 11:07:50 AM PDT	sjsam92268@agentforce.com		Added standard button override: Tab (Lightning Page null)	Standard Buttons and Links		
28	10/11/2025, 11:07:35 AM PDT	sjsam92268@agentforce.com		Created Lightning Page: Home Page Default	Lightning Pages		
29	10/11/2025, 10:01:03 AM PDT	sjsam92268@agentforce.com		Dropped external identifier designation: Complaint.Complaint ID	Company Information		
30	10/11/2025, 10:01:03 AM PDT	sjsam92268@agentforce.com		Changed field type of Complaint ID custom field on Complaints from Auto Number to Text	Custom Objects		
31	10/11/2025, 3:25:27 AM PDT	salesforce.com, inc.		Max number of streaming topics			
32	10/11/2025, 2:07:45 AM PDT	salesforce.com, inc.		A Salesforce employee ran Update Web Cart List Views in Update mode for case Id 000000000000000		Salesforce Administrator	
33	10/11/2025, 2:07:45 AM PDT	salesforce.com, inc.		A Salesforce employee ran Update Usage Metrics Releasable Action 258 . 0 in Update mode for case Id 000000000000000		Salesforce Administrator	
34	10/11/2025, 2:07:44 AM PDT	salesforce.com, inc.		A Salesforce employee ran Slack RC Mapping Team Id Backfill Releasable Action 258 in Update mode for case Id 000000000000000		Salesforce Administrator	
35	10/11/2025, 2:07:44 AM PDT	salesforce.com, inc.		A Salesforce employee ran Update Standard Flow Time Trigger Data in Update mode for case Id 000000000000000		Salesforce Administrator	
36	10/11/2025, 2:07:43 AM PDT	salesforce.com, inc.		A Salesforce employee ran Shield App Release Action in Update mode for case Id 000000000000000		Salesforce Administrator	
37	10/11/2025, 2:07:42 AM PDT	salesforce.com, inc.		A Salesforce employee ran Auto Activate Release Updates in Update mode for case Id 000000000000000		Salesforce Administrator	
38	10/11/2025, 2:07:42 AM PDT	salesforce.com, inc.		A Salesforce employee ran Create ProductRelationshipType Records and All List View in Update mode for case Id 000000000000000		Salesforce Administrator	
39	10/11/2025, 2:07:42 AM PDT	salesforce.com, inc.		A Salesforce employee ran Payment Credit Process Exception Releasable Action in Update mode for case Id 000000000000000		Salesforce Administrator	

SetupAuditTrail1760551244162

📌Phase 10: Quality Assurance Testing

Introduction

This final phase focuses on verifying the accuracy, performance, and reliability of all features implemented in the **Customer Complaint CRM Project**.
Each Salesforce functionality—ranging from record creation to automation and reporting—has been tested through structured test cases.
Each test case includes:

- Input details (what data/action was tested)
- Expected Output (anticipated behavior)
- Actual Output (with result and screenshot)

Test Case Format

We will use the following structured format for all test cases:

Use Case / Scenario	Test Steps (with input)	Expected Result	Actual Result (with Screenshot)

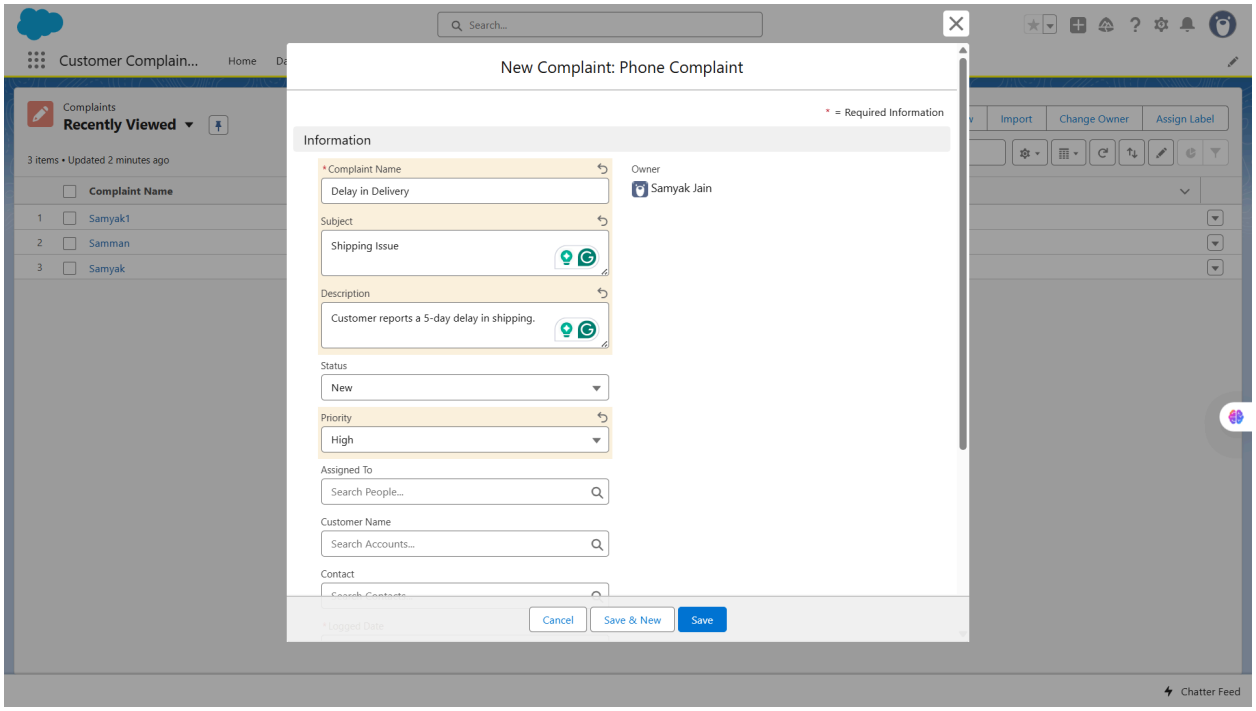
Details:

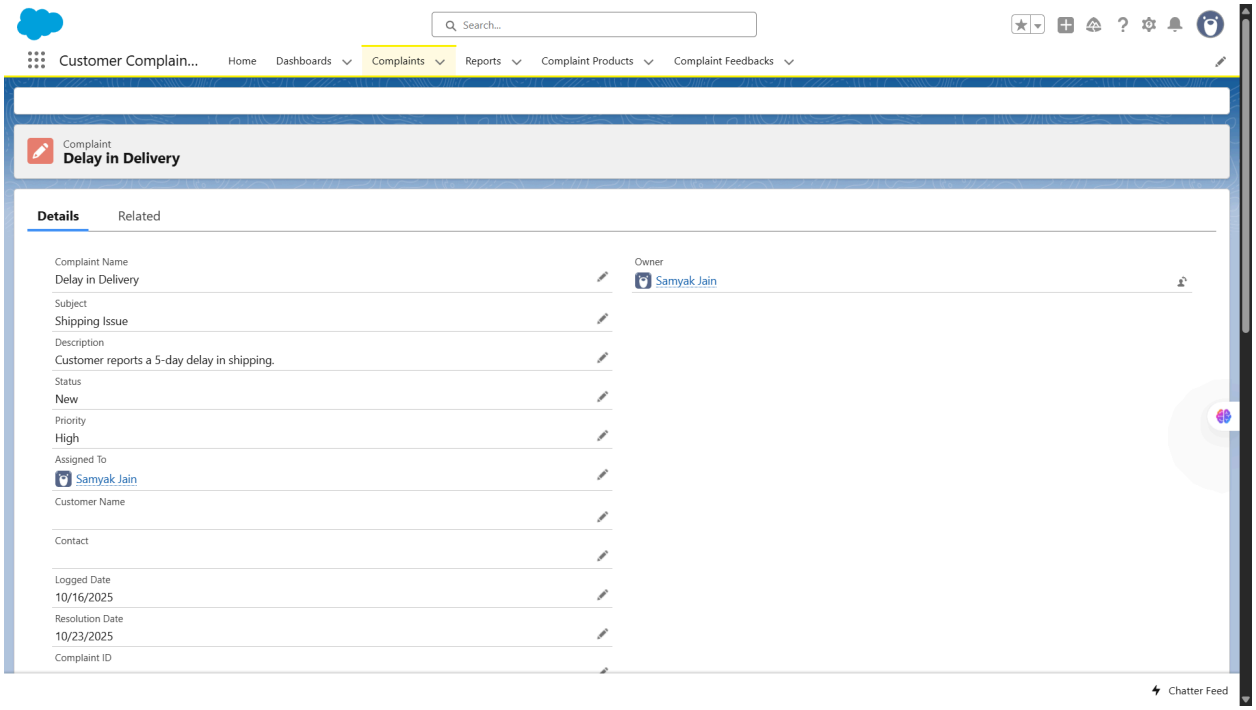
- **Use Case / Scenario:** The feature or functionality you are testing.
- **Test Steps (with input):** Step-by-step actions performed including input values.
- **Expected Result:** What should happen after performing the steps.
- **Actual Result (with Screenshot):** Capture what actually happened with screenshots for verification.

Test Cases

🔹 1. Complaint Record Creation

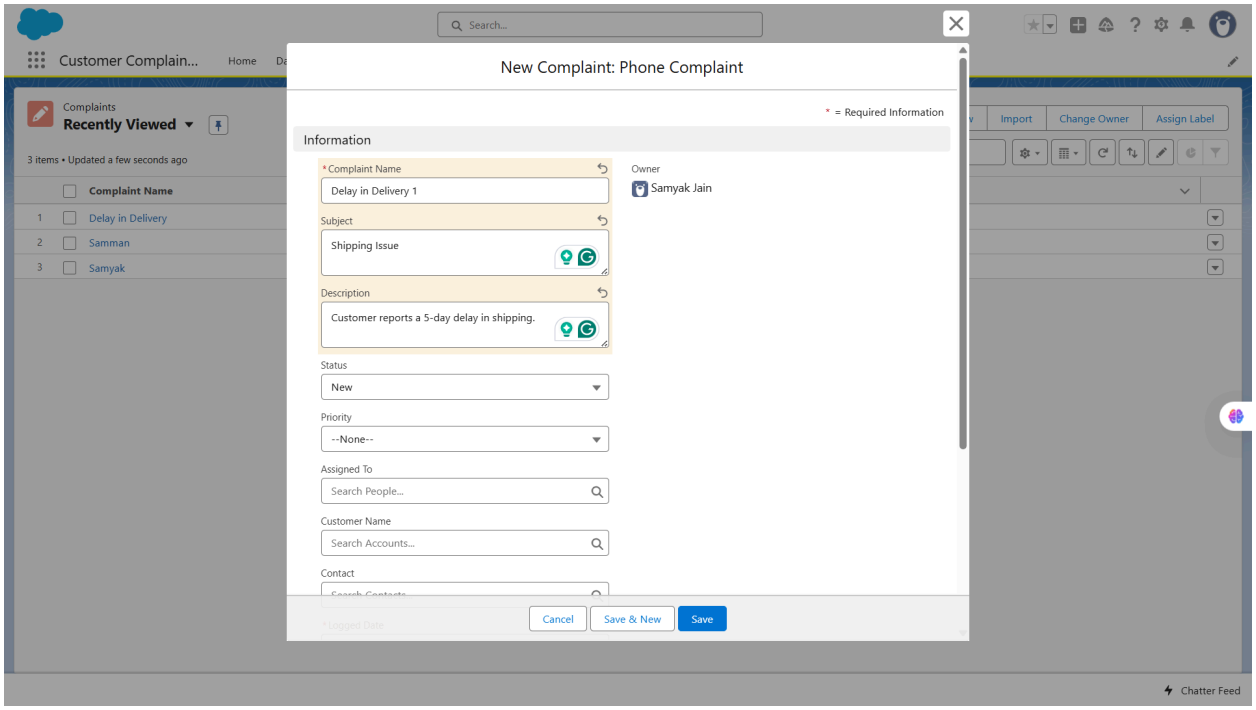
Use Case / Scenario	Test Steps (with Input)	Expected Result	Actual Result (with Screenshot)
Create a new Complaint record	1. Navigate to **Complaints Tab**.2. Click **New**.3. Enter:• Complaint Name: “Delay in Delivery”• Subject: “Shipping Issue”• Description: “Customer reports a 5-day delay in shipping.”• Status: “New”• Priority: “High”4. Click **Save**.	New Complaint record is created successfully with all details visible in the list view.	Screenshot showing Complaint record successfully created and displayed.

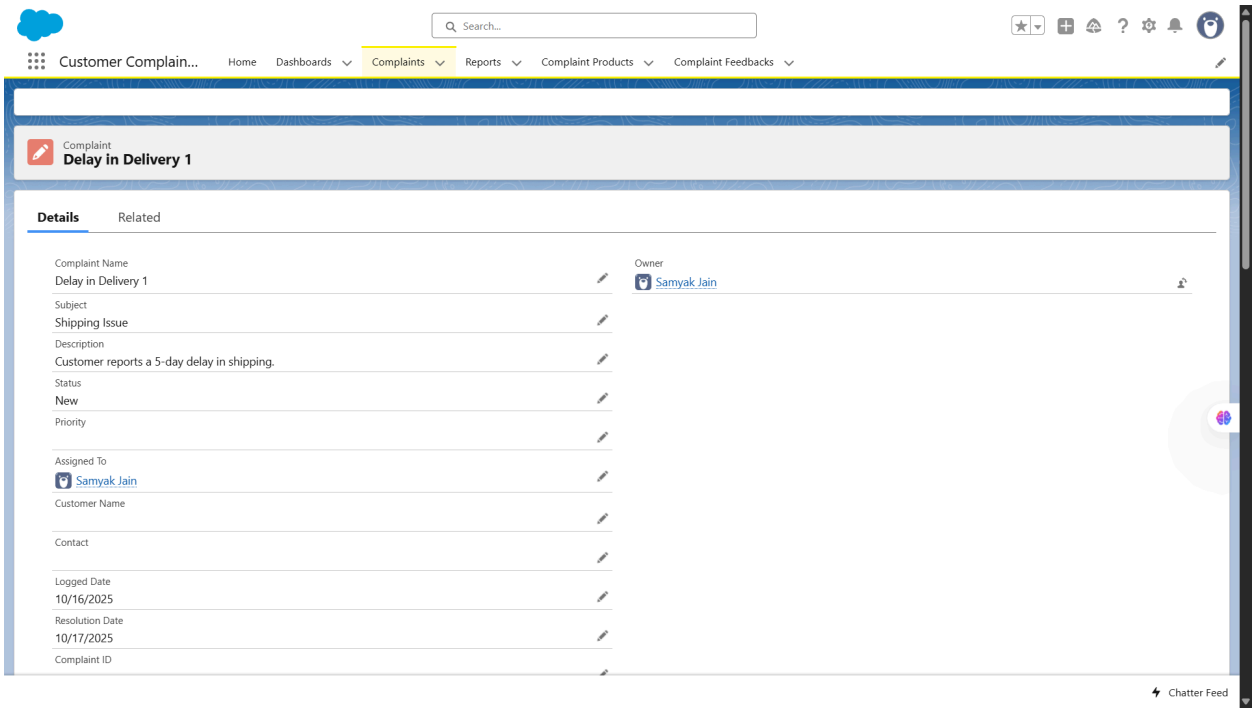




🔹 2. Complaint Validation Rule

Use Case / Scenario	Test Steps (with Input)	Expected Result	Actual Result (with Screenshot)
Prevent saving a Complaint without Priority	1. Navigate to **Complaints Tab**.2. Click **New**.3. Leave the Priority field blank.4. Click **Save**.	Validation Rule should prevent saving and display an error message: “Priority is required.”	Screenshot showing validation error message preventing record save.



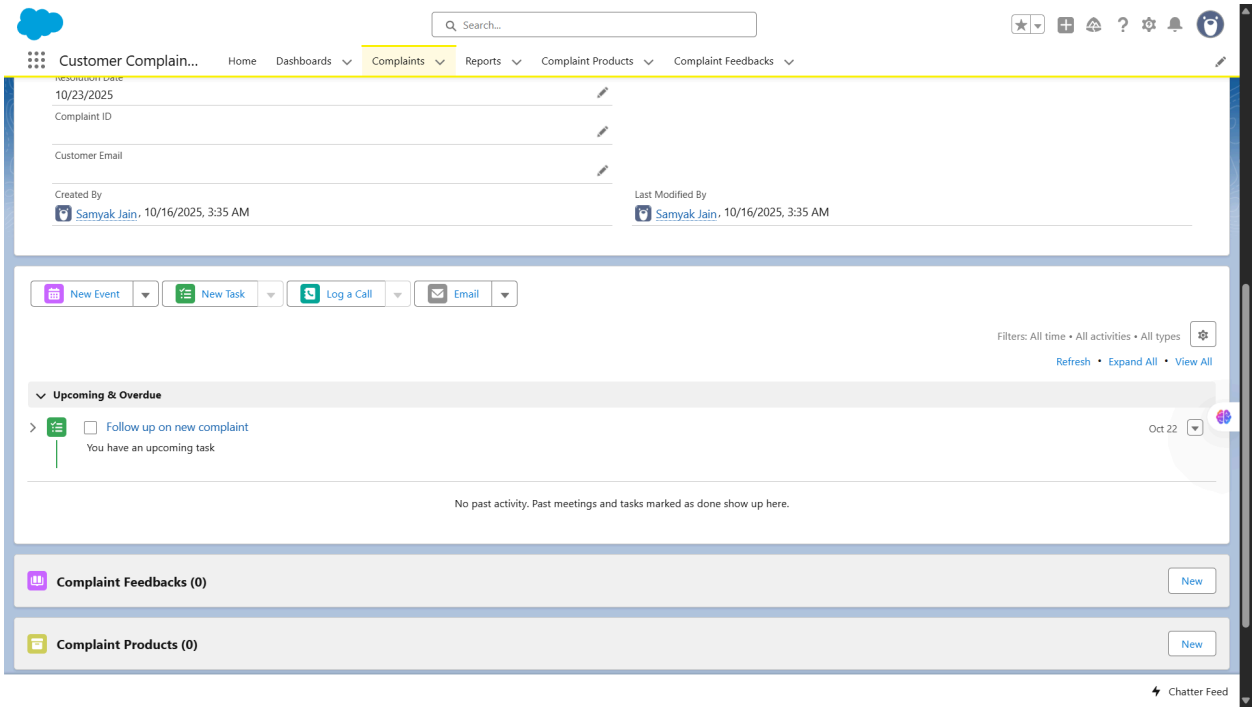


---

3. Automatic Task Creation (Flow Automation)

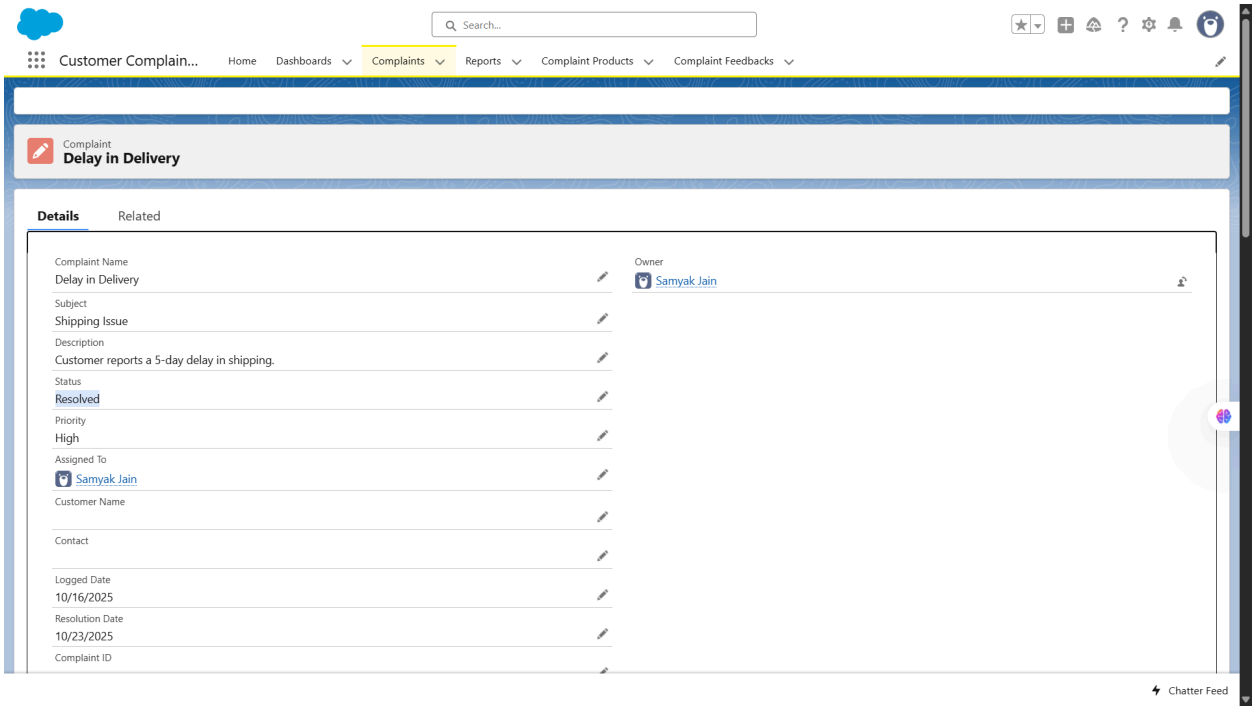
Use Case / Scenario	Test Steps (with Input)	Expected Result	Actual Result (with Screenshot)

Automatically create a Task when a Complaint is created	1. Create a new Complaint.2. Save the record.3. Open **Tasks Related List** under the same record.	A Task should automatically be created for the Complaint Owner with the subject “Follow up on new complaint.”	Screenshot showing Task record automatically generated for the Complaint Owner.



🔹 4. Complaint Approval Process

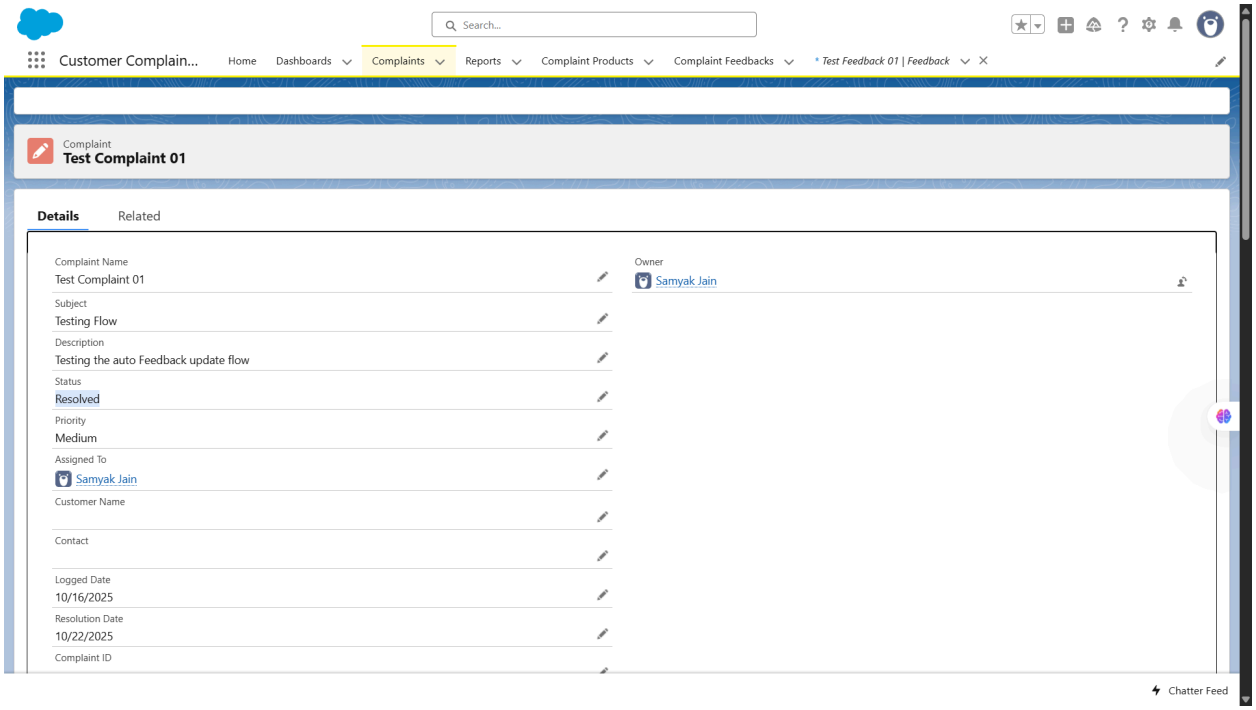
Use Case / Scenario	Test Steps (with Input)	Expected Result	Actual Result (with Screenshot)
Submit a resolved Complaint for approval	1. Change Complaint Status to “Resolved.”2. Click **Submit for Approval**.3. Open Approval History.	Complaint enters Approval Process, approver receives email notification, status changes to “Pending Approval.”	Screenshot showing Approval History and approver notification.

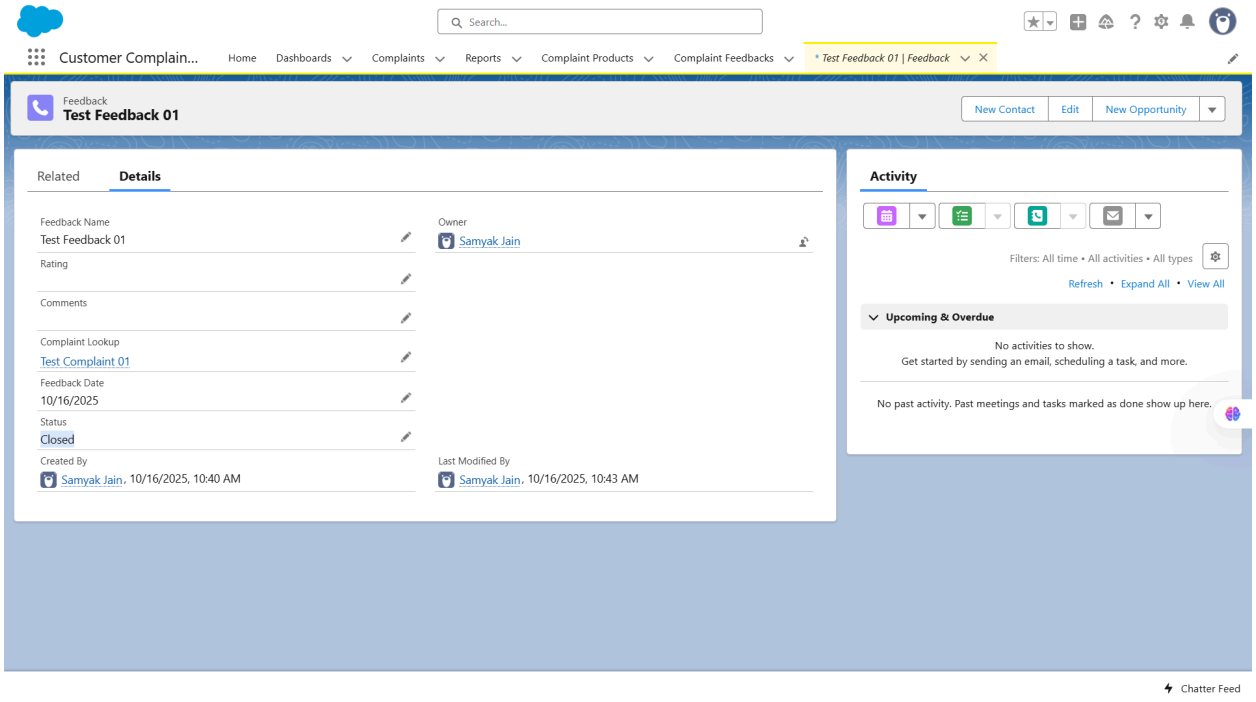


---

5. Trigger Functionality – Update Related Feedback

Use Case / Scenario	Test Steps (with Input)	Expected Result	Actual Result (with Screenshot)
Update Feedback when Complaint is resolved	1. Resolve a Complaint that has a linked Feedback.2. Click **Save**.	Related Feedback Status should automatically update to “Closed.”	Screenshot showing Feedback record updated after Complaint resolution.

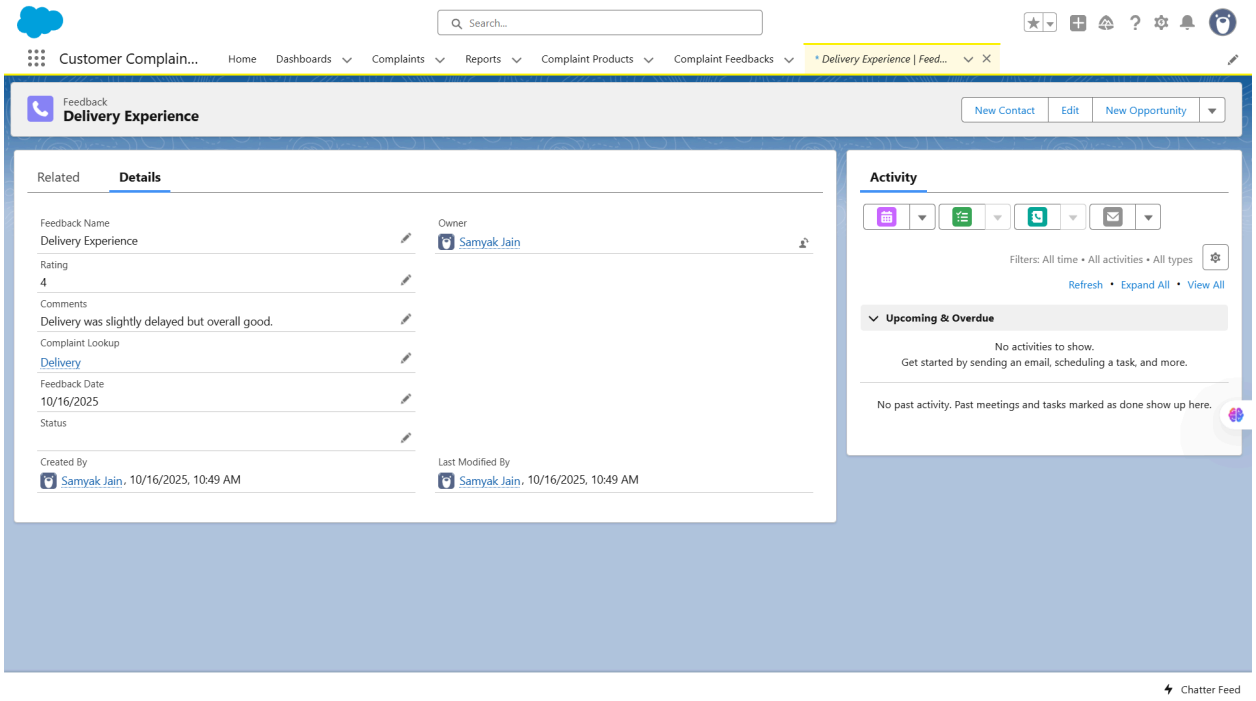




🔹 6. Complaint Feedback Record Creation

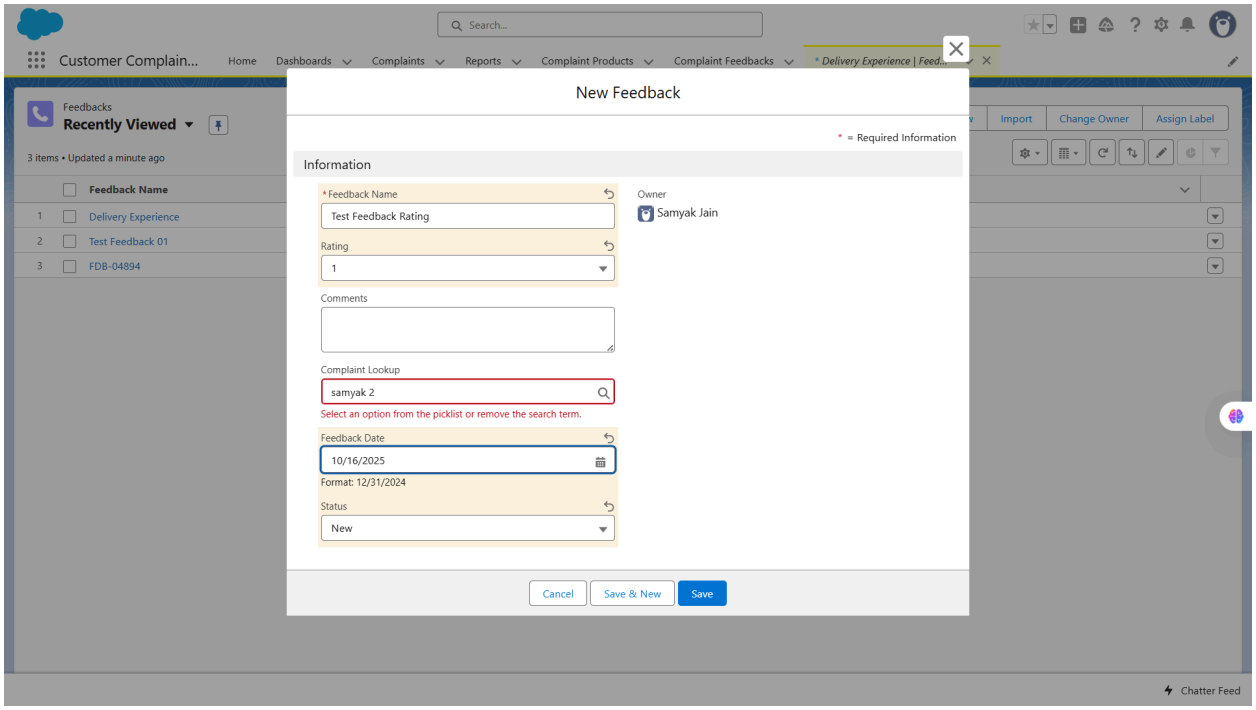
Use Case / Scenario	Test Steps (with Input)	Expected Result	Actual Result (with Screenshot)

Create a new Feedback record	1. Navigate to **Feedback Tab**. 2. Click **New**. 3. Enter:• Feedback Name: “Delivery Experience”• Feedback Date: “10/16/2025”• Rating: “4”• Comments: “Delivery was slightly delayed but overall good.” 4. Click **Save**.	New Feedback record should be created successfully and linked to related Complaint.	Screenshot showing Feedback record created and visible in list view.



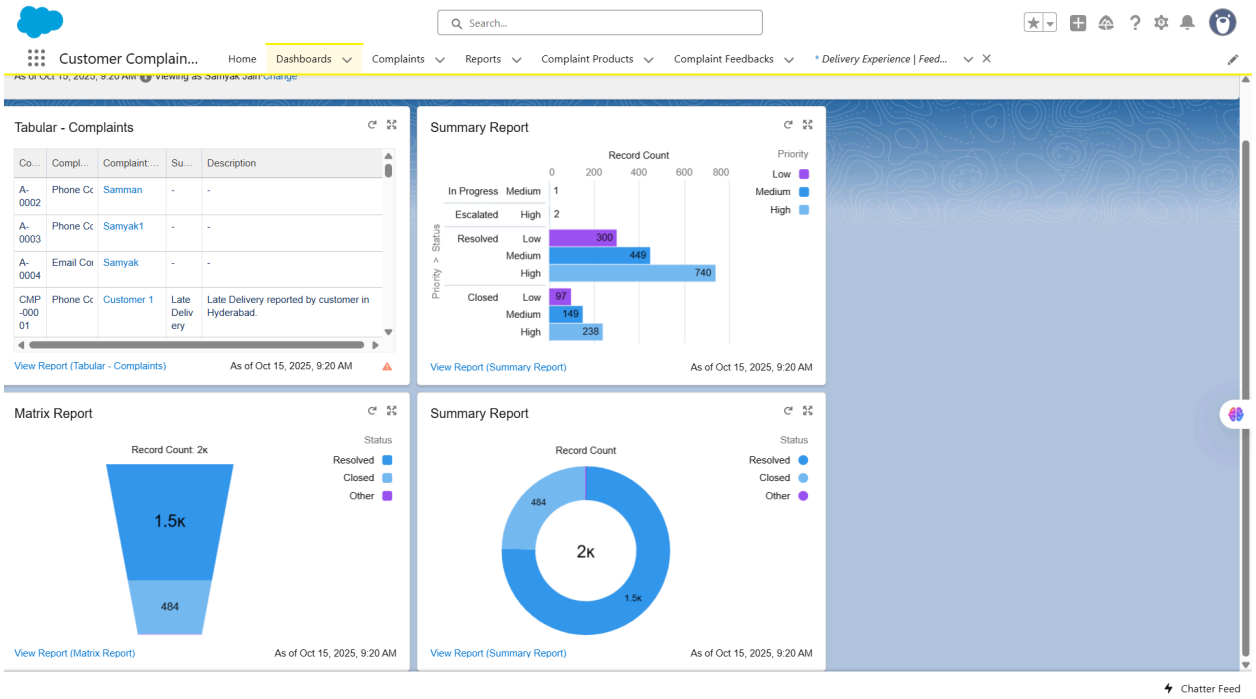
🔹 7. Validation Rule – Feedback Rating Range

Use Case / Scenario	Test Steps (with Input)	Expected Result	Actual Result (with Screenshot)
Prevent Feedback Rating outside allowed range	1. Create a Feedback. 2. Enter unknown complaint. 3. Click **Save**.	Validation Rule triggers error message: “Select an option from picklist”	Screenshot showing error message on invalid rating entry.



🔹 8. Dashboards & Dynamic Dashboards

Use Case / Scenario	Test Steps (with Input)	Expected Result	Actual Result (with Screenshot)
Display Complaint and Feedback metrics	1. Navigate to **Dashboards Tab**. 2. Click **New Dashboard**. 3. Add components from Complaint and Feedback reports. 4. Save Dashboard. 5. Enable “View Dashboard as Logged-In User.”	Dashboard should display dynamic charts (e.g., Complaint Status by Priority, Average Feedback Rating).	Screenshot showing dynamic Dashboard components correctly reflecting live data.



Conclusion

After completing all test cases across modules, the **Customer Complaint CRM Project** has been thoroughly verified for:

- Correct functionality of all Salesforce components (Objects, Triggers, Flows, LWC, Reports, and Dashboards).
- Accurate data handling, automation reliability, and proper UI interactions.
- Validation Rules ensuring data integrity and completeness.
- Automated processes such as Flows, Triggers, and Approval Processes were validated to ensure they work as expected.
- Successful visualization of data via Reports and Dashboards.

✅ **Result:** All test cases have passed successfully.
The **Customer Complaint CRM** is fully functional, reliable, and deployment-ready.
