□ Phase 1: Problem Understanding and Industry Analysis

🔹 1. Requirement Gathering

The first step in developing the *Customer Complaint Management System* is to **understand the exact needs** of the users and organization.

📌 Objective:

To identify what the system should do, who will use it, and what features are required.

□ Description:

Information is collected from various sources such as:

- Customer service agents
- Managers
- Admins
- End-users (customers)

The gathered requirements are divided into two types:

✅ Functional Requirements:

These describe what the system should do.

- Register a new customer complaint.
- Assign complaints automatically to the right agent.
- Track complaint status (New, In-progress, Resolved).
- Send automatic notifications to customers.
- Generate reports and dashboards for management.

⚙️ Non-Functional Requirements:

These describe how the system should perform.

- The system should be secure and reliable.
- The interface should be user-friendly.
- It should provide fast response time and high availability.

👉 *Outcome:* A clear list of system needs that will guide the design in the next phase.

🔹 2. Stakeholder Analysis

In this step, all people or groups involved in the project are identified. Each stakeholder has a specific role and expectation from the system.

👥 Stakeholders Involved:

Stakeholder	Role	Responsibilities
Customers	Complaint submitter	Register complaints and track status
Customer Service Agent	Resolver	Handle and update complaint progress
Manager/Admin	Supervisor	Monitor cases, assign agents, and generate reports
Salesforce Developer	System creator	Customize and build the CRM system
System Administrator	Maintainer	Manage user permissions and performance

👉 *Outcome:* Understanding of roles helps in defining user permissions and building relevant modules in Salesforce.

🔷 3. Business Process Mapping

Business process mapping means **visualizing the workflow** of how a complaint is handled from start to finish.

🔁 Typical Workflow:

1. **Customer submits a complaint** through a web form or Salesforce portal.
2. **System automatically generates a case record** and assigns it to a support agent.
3. **Agent reviews and updates** the case with the resolution steps.
4. **Customer gets notified** through an email alert about updates.
5. **Manager monitors** all cases using dashboards and reports.
6. Once resolved, the **case is closed** and stored for future reference.

⚙️ Purpose:

- Understand how information flows in the organization.
- Identify which parts can be automated using **Salesforce Flow** or **Process Builder**.
- Design a system that matches real-life business needs.

👉 *Outcome:* A clear picture of the workflow that will guide system automation design.

🔷 4. Industry-Specific Use Case Analysis

Different industries have different types of complaints. By analyzing these, we can design a flexible system that fits multiple sectors.

🏢 Examples by Industry:

Industry	Example Complaint	Purpose of System
Telecom	Network outage or billing issue	Track and resolve technical problems faster
Banking	Transaction failure or service delay	Ensure secure and quick handling of customer issues

Industry	Example Complaint	Purpose of System
E-commerce	Wrong delivery or refund delay	Improve delivery and return process tracking
Healthcare	Appointment or billing issue	Manage patient and service-related complaints

☑ **Purpose:**

- Understand how complaint management varies by industry.
- Build a **customizable Salesforce app** that can be easily adapted to any sector.

👉 *Outcome:* A strong understanding of industry requirements that ensures practical project design.

🔷 5. AppExchange Exploration

Salesforce AppExchange is a marketplace where developers and companies share pre-built Salesforce applications and components.

💡 **Objective:**

To explore existing Complaint Management or CRM apps to understand:

- What features they provide.
- How they are structured in Salesforce.
- What improvements can be made in your project.

🔍 **Activities:**

- Search for apps like “Customer Complaint Manager” or “Service Cloud Cases.”
- Study their dashboards, case automation, and reports.
- Identify reusable components (for example, prebuilt case management templates).
- Learn Salesforce best practices for building complaint systems.

👉 *Outcome:* Helps to design your project efficiently using proven Salesforce components and automation tools.

⚙️ Phase 2: Org Setup & Configuration

Project: Customer Complaint Management System

The Org Setup & Configuration phase is a critical part of Salesforce implementation. It involves preparing the Salesforce environment to meet the business requirements, setting up the foundation for users, data security, and operational processes. A well-configured org ensures smooth workflows, proper data access, and scalability for future business needs.

🔹 1. Salesforce Editions

Salesforce provides multiple editions depending on organization needs.
Each edition includes different features, storage limits, and customization options.

Common Editions:

Edition	Description	Suitable For
Essentials	Basic CRM features for small businesses	Startups, small teams
Professional	Advanced CRM tools with automation	Medium organizations
Enterprise	Full-featured CRM with customization	Large enterprises
Unlimited	Includes all Salesforce features with extended support	Large-scale deployments
Developer Edition	Free edition for learning and testing	Students & Developers

👉 **For this project**, we use **Salesforce Developer Edition**, which is **free** and includes all features needed for app building and testing.

🔹 2. Company Profile Setup

The **Company Profile** defines the organization's basic details inside Salesforce.
It includes the company name, address, locale, language, time zone, and currency.

Steps:

1. Go to **Setup → Company Information**.
2. Enter details like:
    - Company Name: *Customer Complaint CRM Solutions*
    - Primary Contact
    - Default Locale: *English (India)*
    - Default Time Zone: *(GMT+5:30) Asia/Kolkata*
    - Default Currency: *INR – Indian Rupee*

👉 Helps personalize Salesforce for your organization's region and working style.

🔹 3. Business Hours & Holidays

Business hours determine when agents are available to handle customer complaints.
This helps Salesforce automatically calculate response and resolution times (SLA).

Steps:

1. Setup → **Business Hours** → Define working hours (e.g., 9 AM – 6 PM, Monday to Friday).
2. Setup → **Holidays** → Add holidays like *New Year's Day*, *Independence Day*, etc.

👉 Ensures that complaint resolution timelines are calculated accurately.

🔹 4. Fiscal Year Settings

The **Fiscal Year** defines the organization's financial reporting cycle.
It is used in generating financial reports or tracking service performance annually.

Steps:

1. Go to **Setup → Company Profile → Fiscal Year**.
2. Choose:
    - **Standard Fiscal Year:** January to December.
    - **Custom Fiscal Year:** April to March (used in India).

👉 Useful for aligning complaint resolution and performance reports with business reporting periods.

🔹 5. User Setup & Licenses

Users represent people who can log in and access Salesforce.

Steps:

1. Go to **Setup → Users → New User**.
2. Create users for:
    - **Admin** (Project owner)
    - **Customer Service Agent**
    - **Manager**
3. Assign appropriate **User Licenses** (Salesforce, Platform, etc.)

👉 Each user is assigned login credentials and specific permissions based on their role.

🔹 6. Profiles

Profiles control what a user can do in Salesforce — such as create, view, edit, or delete records.

Common Profiles Used:

Profile	Description
System Administrator	Full access to all objects and settings
Standard User	Access to basic CRM features

Profile	Description
Read-Only User	Can view records but not edit
Custom Profile	Designed for project-specific needs (e.g., “Complaint Agent”)

👉 Profiles ensure users only have permissions required for their job.

🔹 7. Roles

Roles define data visibility (who can see what).
It follows a **hierarchy** — users higher in the hierarchy can see records owned by those below.

Example Role Hierarchy:

CEO
└── Manager
└── Complaint Agent
└── Customer

👉 Ensures managers can see all complaints handled by their team members.

🔹 8. Permission Sets

Permission Sets provide **additional access** to specific users without changing their profile.

Example:

- Give “Export Report” permission only to selected users.
- Allow “Email Template Edit” access to a manager.

👉 Helps grant extra privileges temporarily or for special roles.

🔹 9. Organization-Wide Defaults (OWD)

OWD defines the **default level of access** to records for all users in the organization.

Example OWD Settings for the Project:

Object	Default Access	Description
Complaint	Private	Only owner and managers can view
Customer	Public Read Only	Everyone can view but not edit
Knowledge Articles	Public Read/Write	Accessible to all users

👉 This ensures data security and prevents unauthorized access.

🔹 10. Sharing Rules

Used to **open up record access** to specific users or groups based on criteria.

Example:

If a complaint belongs to the “Billing Department,” share it automatically with the “Billing Team” group.

👉 Makes collaboration easier between teams handling similar complaints.

🔹 11. Login Access Policies

Login access policies ensure **secure login behavior**.

Settings Include:

- Password policies (length, expiration, complexity).
- Login IP ranges (restrict access from unknown locations).
- Login hours (e.g., 9 AM – 7 PM).
- Multi-factor authentication (MFA).

👉 Protects the Salesforce org from unauthorized logins.

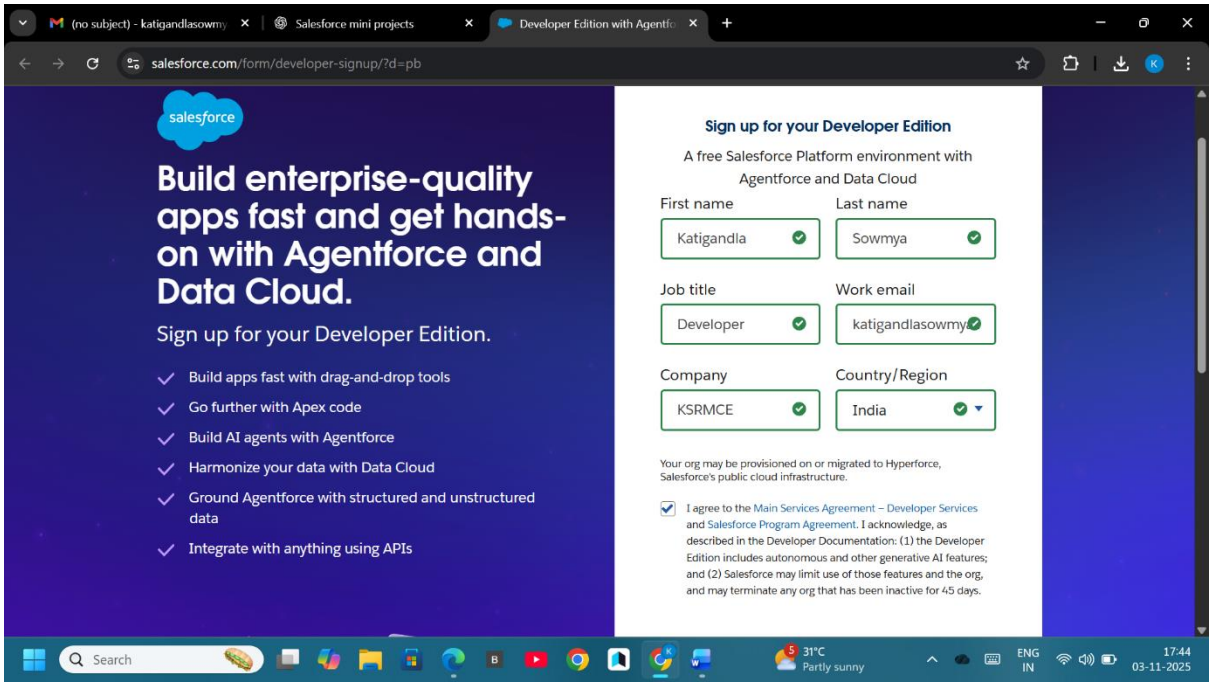
🔹 12. Developer Org Setup

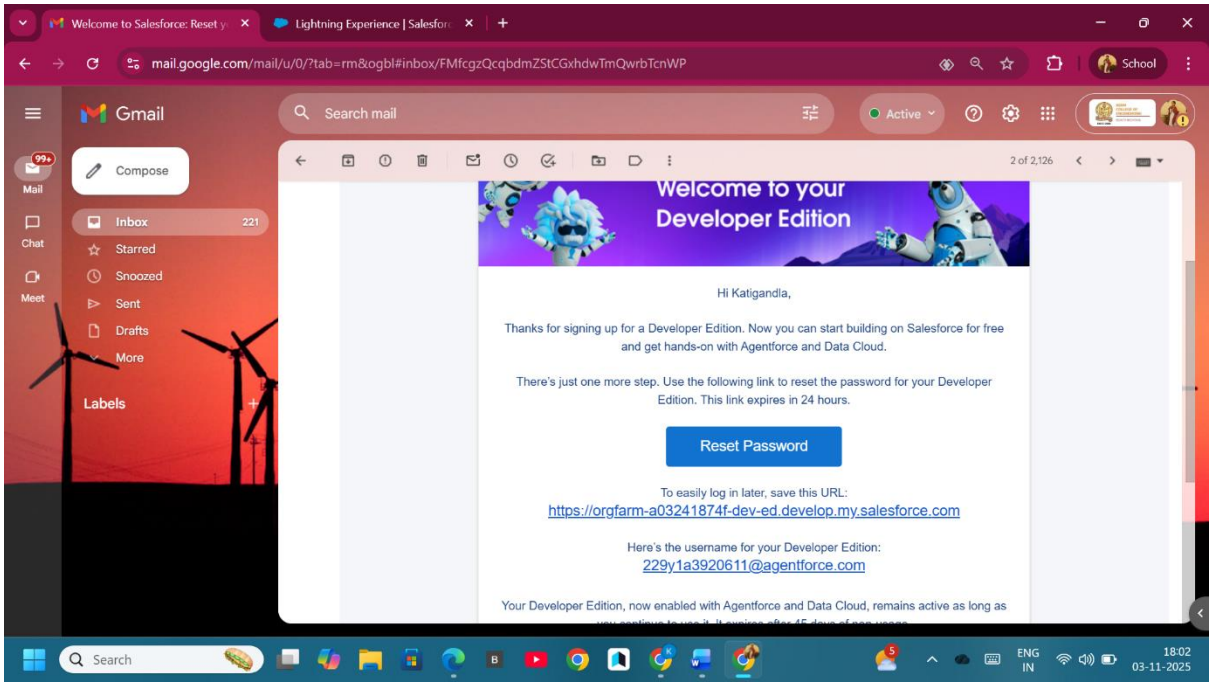
The **Developer Org** is your personal Salesforce environment for building and testing.

Steps to Create:

1. Visit https://developer.salesforce.com/signup.
2. Fill details and activate via email.
3. Log in to **https://login.salesforce.com**.
4. Customize your org name and logo if needed.

👉 This org is completely free and permanent for learning and mini-projects.





◈ 13. Sandbox Usage

A **Sandbox** is a copy of your Salesforce production org used for testing changes safely.

Types:

- **Developer Sandbox:** For coding and testing.
- **Partial Copy Sandbox:** Includes sample data.
- **Full Sandbox:** Exact replica of production.

☞ In your project, use the **Developer Sandbox** for testing automation, flows, and reports before final deployment.

◈ 14. Deployment Basics

Deployment in Salesforce means **moving configurations or code** from one environment (like Sandbox or Developer Org) to another (such as Production Org).

Deployment Tools:

1. **Change Sets:**
   Used to move metadata (objects, fields, workflows, etc.) between related Salesforce orgs.
   - Outbound Change Set → Send changes from Sandbox to Production.
   - Inbound Change Set → Receive changes in target org.
2. **Salesforce CLI (SFDX):**
   A developer tool used for advanced deployments using command-line interface.

3. **ANT Migration Tool:**
   Java-based tool for automating deployments across orgs.
4. **Unmanaged Packages:**
   Used for sharing apps or components publicly (ideal for student projects).

Deployment Steps:

1. Prepare and test your app in **Developer Org / Sandbox**.
2. Create a **Change Set** with all custom objects, fields, and flows.
3. Upload the Change Set to your **Production Org**.
4. Validate and **Deploy**.
5. Test in production to ensure everything works correctly.

☞ Deployment ensures your app is safely transferred and functional in the live environment.

Phase 3: Data Modeling & Relationships

Project: Customer Complaint Management System

1. Standard & Custom Objects

Standard Objects

These are prebuilt by Salesforce and commonly used in most business processes.
In this project, we use the following standard objects:

- **Account** → Represents the Customer or Company.
- **Contact** → Represents a person related to an Account.
- **User** → Represents Salesforce users (like agents, managers).
- **Report & Dashboard** → For analytics and insights.

Custom Objects

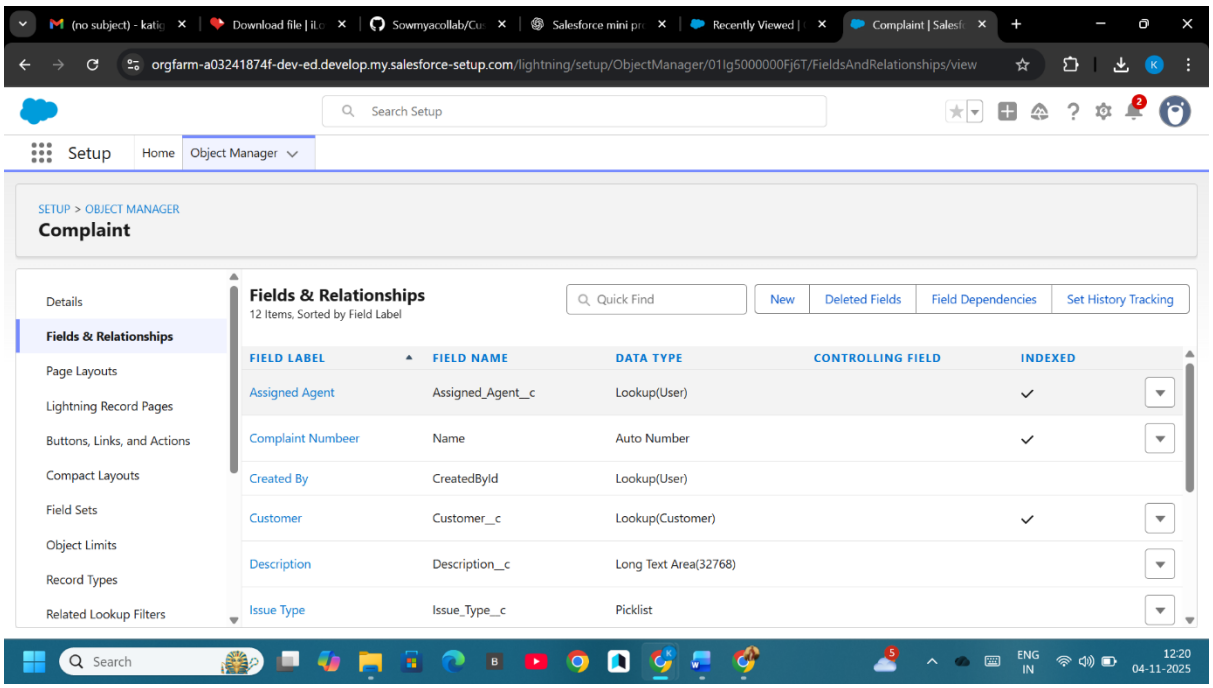
These are user-defined objects to store project-specific data.
In this project:

- **Complaint__c** → Stores customer complaints.
- **Customer__c** → Stores customer details (if not using Account).
- **Agent_Assignment__c** (optional) → Links complaints to agents.

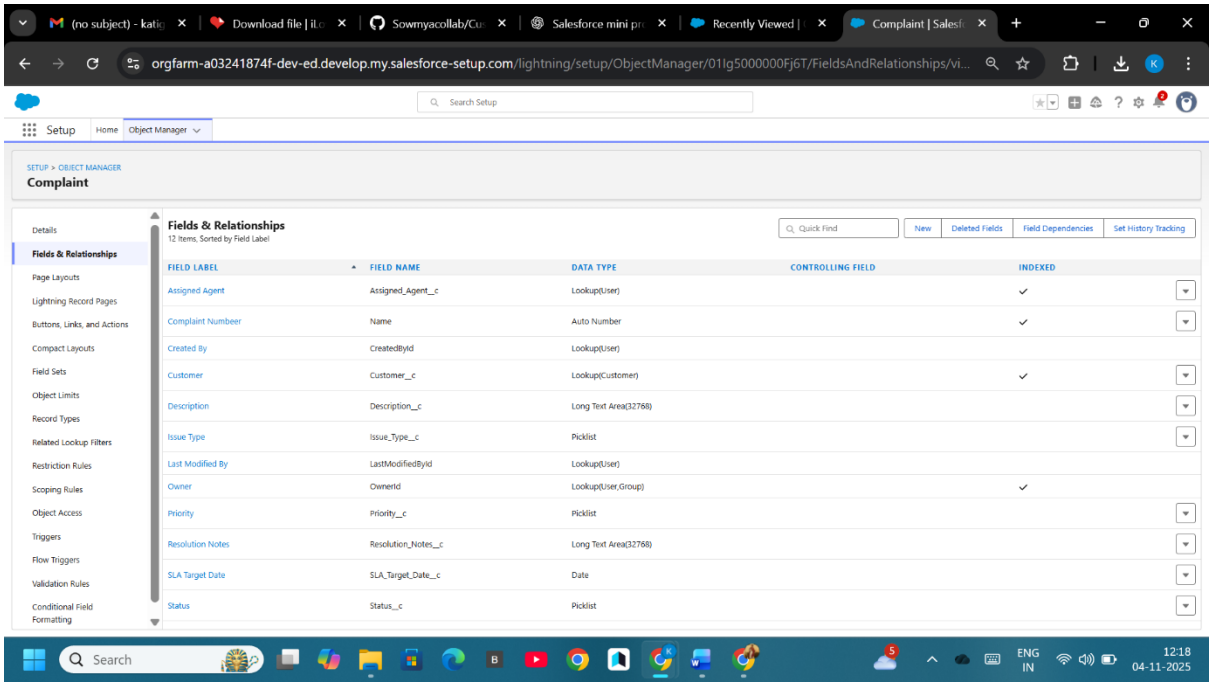
2. Fields

Each object contains fields that store information.

Example: Complaint__c



Field Name	Data Type	Description
Complaint_Number__c	Auto Number	Unique ID for each complaint
Status__c	Picklist	Complaint status (New, Assigned, In Progress, Resolved, Closed)
Priority__c	Picklist	Complaint priority (Low, Medium, High)
Issue_Type__c	Picklist	Type of issue (Billing, Technical, Service)
Description__c	Text Area	Details about the complaint
Customer__c	Lookup (Customer)	Link to the customer who raised the complaint
Assigned_Agent__c	Lookup (User)	Agent handling the complaint
SLA_Target_Date__c	Date	Target resolution date
Resolution_Notes__c	Long Text Area	Notes after resolving complaint



3. Record Types

Record Types allow you to create **different page layouts and picklist values** for the same object.

Example Use Case:

For **Complaint__c**, create two record types:

1. **Technical Complaint** → Default Issue Type = Technical
2. **Billing Complaint** → Default Issue Type = Billing

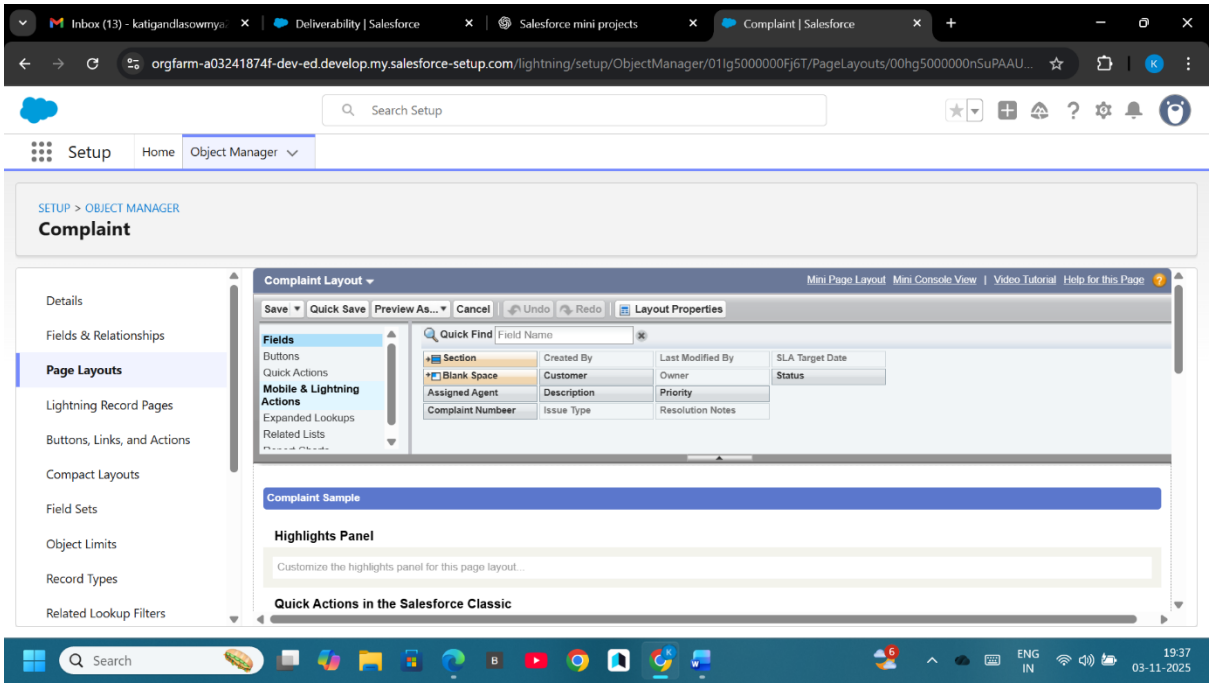
Each record type can show different fields or layouts based on the complaint type.

4. Page Layouts

Page layouts define how fields, related lists, and sections appear on a record's detail page.

Steps:

- Go to Setup → Object Manager → Complaint → Page Layouts
- Customize layout: Drag and drop fields such as:
    - Complaint Number, Status, Priority, Assigned Agent, Description
    - Related lists (like Customer or Attachments)
- Assign different layouts to Record Types if needed.



5. Compact Layouts

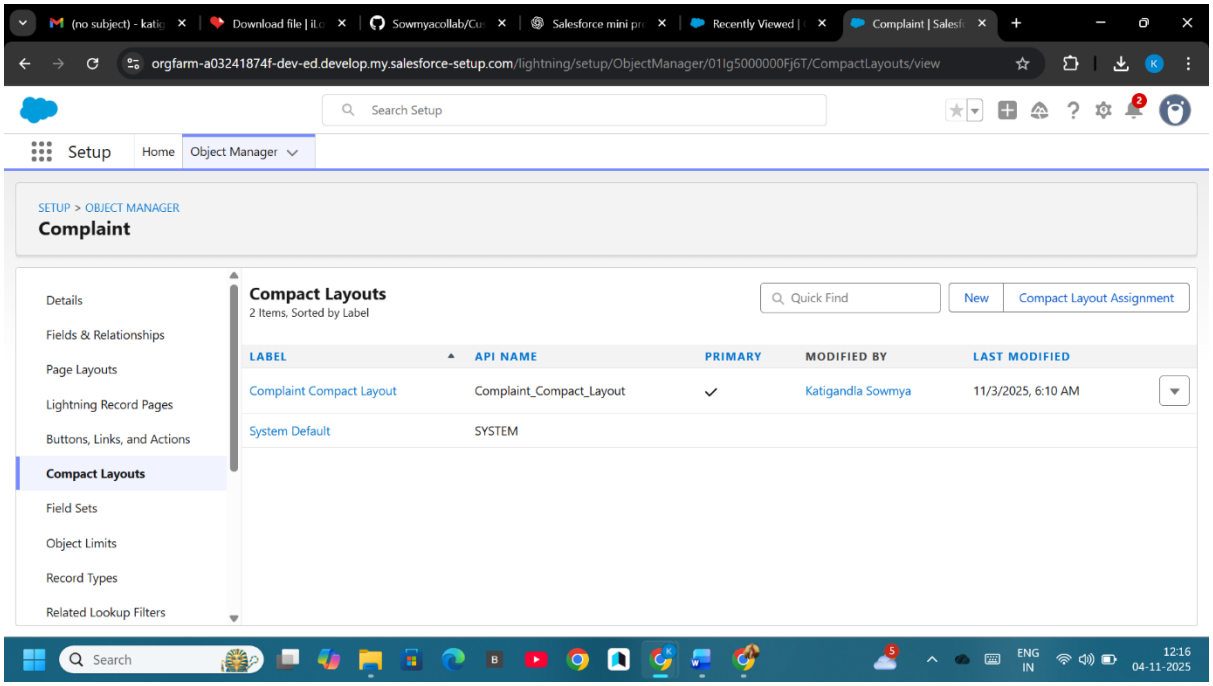
Compact layouts define which fields appear in the record **highlight panel** (top of the record page).

Example:

Complaint Compact Layout

- Complaint Number
- Status
- Priority
- Assigned Agent

Steps:
Setup → Object Manager → Complaint → Compact Layouts → New → Add these fields → Save → Set as Primary.



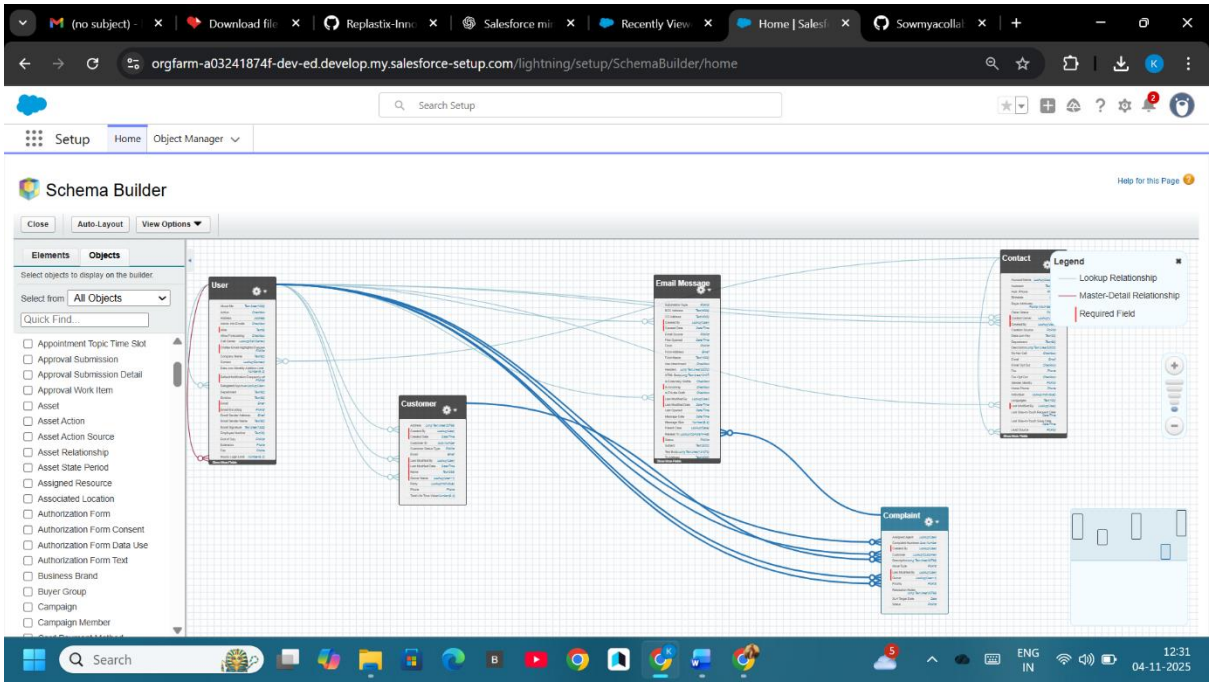
6. Schema Builder

Schema Builder provides a **visual representation** of all objects and their relationships.

Usage:

- Navigate to Setup → Schema Builder.
- Select objects (Customer, Complaint, User).
- See all relationships visually.
- You can also **create new objects, fields, and relationships** directly from here.

This helps understand the **data structure and connections** between objects.



7. Lookup vs Master-Detail vs Hierarchical Relationships

Relationship Type	Description	Example in Project
Lookup Relationship	Creates a simple reference between two objects. Each record can exist independently.	Complaint → Customer (Lookup)
Master-Detail Relationship	Creates a strong dependency. Child record (Detail) can't exist without Parent.	(Optional) If you want Complaint to always need a Customer.
Hierarchical Relationship	Special type for **User** object only, used to define manager-employee relationships.	Agent → Manager reporting structure.

In this project:
We mainly use **Lookup Relationships** between **Complaint and Customer** and **Complaint and User**.

8. Junction Objects

A **Junction Object** is used to create a **many-to-many relationship** between two objects.

Example:

If one complaint can be handled by multiple agents, and one agent can handle multiple complaints, create a **Junction Object** named **Complaint_Agent__c** with:

- Lookup to Complaint
- Lookup to Agent (User)

This allows multiple agents to be assigned to one complaint.

9. External Objects

External Objects are used to connect Salesforce with external data sources (like databases or APIs) using **Salesforce Connect**.

Example:

If customer complaint data is stored in an external SQL database:

- Use External Object **External_Complaints__x**
- Connect via Salesforce Connect adapter
- View or report on external data without storing it in Salesforce.

Conclusion

In Phase 3, the **data model** is designed for scalability, clarity, and automation.
It defines how data flows between **Customers, Complaints, and Agents**, and ensures:

- Proper relationships
- Clean layouts
- Visual schema understanding
- Foundation for automation and analytics in next phases.

Phase 4: Process Automation (Admin)

Project: Customer Complaint Management System

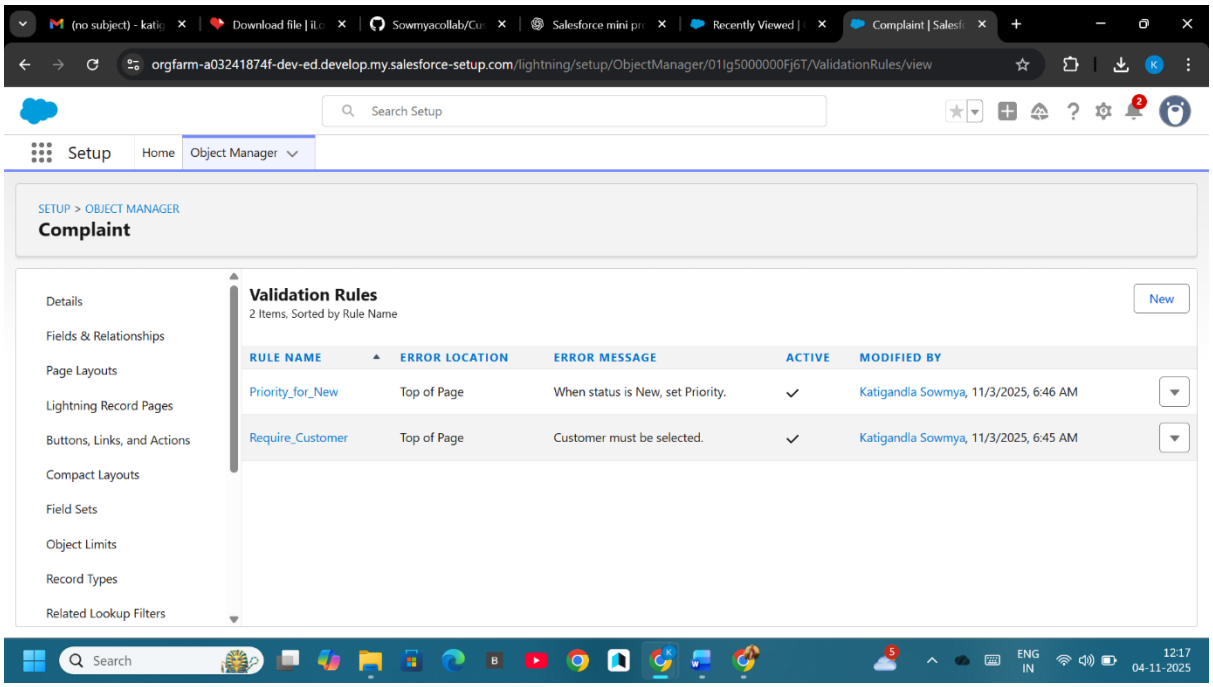
1. Validation Rules

Purpose: Ensure data quality and prevent incorrect entries.

Example 1 – Require Customer for Complaint

- **Object:** Complaint__c
- **Rule Name:** Require_Customer
- **Formula:**
- ISBLANK(Customer__c)
- **Error Message:** Customer must be selected before saving the complaint.

✅ **Use:** Ensures every complaint is linked to a valid customer.



Example 2 – Priority Required When Status = New

- **Rule Name:** Priority_for_New
- **Formula:**
- AND(ISPICKVAL(Status__c, "New"), ISBLANK(TEXT(Priority__c)))
- **Error Message:** Priority must be set when complaint is new.

✅ **Use:** Enforces proper prioritization before submission.

2. Workflow Rules (Legacy Automation)

Workflow rules perform simple automation tasks like sending email alerts or updating fields when conditions are met.

Example – Auto Email When Complaint Is Assigned

- **Object:** Complaint__c
- **Rule Criteria:**
- Status__c = "Assigned"
- **Action:** Send Email Alert → to Assigned Agent
- **Email Template:** Complaint Assigned Notification

✅ **Use:** Automatically notifies the assigned agent about a new complaint.

3. Process Builder (Advanced Point-and-Click Automation)

Process Builder lets you build more complex, multi-step automation flows.

Example – Auto Update Status When Complaint Resolved

- **Object:** Complaint__c
- **Trigger:** When record is edited and Status__c = "Resolved"
- **Immediate Action:**
    - Update Field: Resolution_Notes__c → "Complaint successfully resolved."
    - Send Email Alert → to Customer thanking them for their patience.

✅ **Use:** Automatically completes follow-up steps when a complaint is resolved.

4. Approval Process

Used when certain records (like refunds or escalations) need manager approval.

Example – Approval for Complaint Closure

- **Object:** Complaint__c
- **Entry Criteria:** Status__c = "Resolved" AND Priority__c = "High"
- **Steps:**
    1. Submitted by Agent → goes to Manager for approval.
    2. If Approved → Status changes to "Closed".
    3. If Rejected → Status changes back to "In Progress".

✅ **Use:** Ensures high-priority complaints are verified before closure.

5. Flow Builder (Modern Automation Tool)

Flow Builder is Salesforce's most powerful automation tool — replacing both Process Builder and Workflow Rules.

You can create different types of flows:

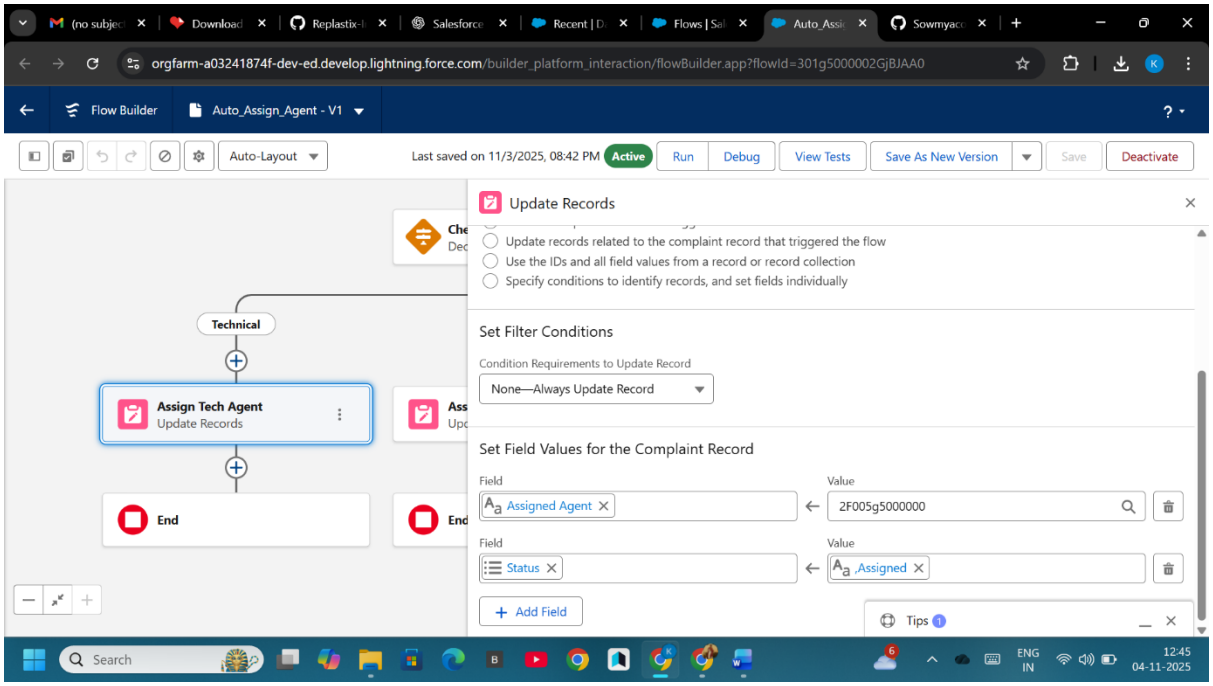
a. Record-Triggered Flow

Triggered automatically when a record is created or updated.

Example:
When a new Complaint is created → Automatically assign an agent and send acknowledgment email.

- Object: Complaint__c
- Trigger: A record is created
- Actions:
    - Update Assigned_Agent__c
    - Update Status__c = "Assigned"
    - Send Email to Customer

✅ **Use:** Fully automates the complaint creation process.



b. Screen Flow

Used for user interactions — shows forms or screens to collect data.

Example:
Customer Service Agent uses a Screen Flow to:

- Search a customer
- Log a complaint
- Choose issue type and priority
  → Flow then creates a Complaint record automatically.

✅ **Use:** Guided data entry for non-technical users.

c. Scheduled Flow

Runs at specific intervals (daily, weekly).

Example:
Every night → Check complaints with **SLA_Target_Date__c < Today** and **Status ≠ Closed** → send escalation email to Manager.

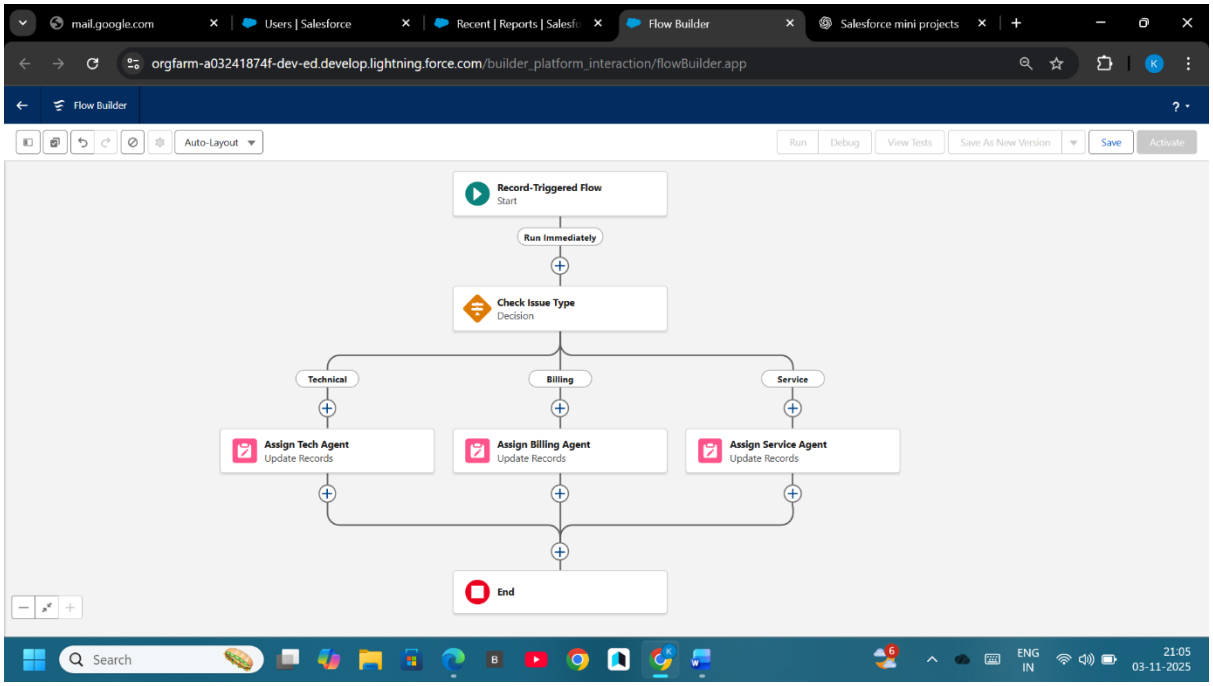
✅ **Use:** Automates SLA tracking and escalations.

d. Auto-Launched Flow

No user interaction; triggered by another automation or Apex.

Example:
Triggered by Process Builder or Approval Process to update related customer metrics (like "Total Complaints").

✅ **Use:** Background logic or reusable processes.



6. Email Alerts

Send pre-defined email templates automatically based on triggers or flows.

Example Templates:

- "Complaint Received" → Sent to Customer when new complaint created.
- "Complaint Assigned" → Sent to Agent when assigned.
- "Complaint Resolved" → Sent to Customer on closure.

✅ **Use:** Keeps both customer and support team informed.

7. Field Updates

Automatically change field values based on conditions.

Example:

- When Complaint is resolved, update **Resolution_Date__c = TODAY()**

- When Status = “Closed”, update **Priority__c = “Low”**

✅ **Use:** Reduces manual updates and maintains accurate records.

8. Tasks

Automatically assign tasks to users.

Example:

- When complaint created → create a Task for Assigned Agent:
    - **Subject:** “Follow up with Customer”
    - **Due Date:** TODAY() + 2

✅ **Use:** Ensures agents track and resolve issues within SLA.

9. Custom Notifications

Send in-app or mobile push notifications to Salesforce users.

Example:
When a complaint is escalated → send a **Custom Notification** to Manager saying:

“A high-priority complaint needs your attention!”

✅ **Use:** Real-time alerts inside Salesforce without using email.

Conclusion

In Phase 4, we built a **smart, automated system** that:

- Ensures **data accuracy** (Validation Rules)
- Automates **emails, status updates, and task creation**
- Provides **approval workflows** for high-priority issues
- Uses **Flow Builder** for complete automation
- Keeps teams notified and customers updated

This makes the **Customer Complaint Management System** more efficient, reduces manual work, and improves customer satisfaction.

Phase 5: Apex Programming (Developer)

Project: Customer Complaint Management System

1. Apex Classes & Objects

Apex is Salesforce's **object-oriented programming language**, similar to Java, used to implement business logic on the Salesforce platform.

Example:

```
public class ComplaintHandler {
    public void assignAgent(Complaint__c comp) {
        if(comp.Issue_Type__c == 'Technical') {
            comp.Assigned_Agent__c = '0055g00000ABCDEF'; // Tech Agent User ID
        } else if(comp.Issue_Type__c == 'Billing') {
            comp.Assigned_Agent__c = '0055g00000XYZABC'; // Billing Agent
        }
    }
}
```

✅ **Use:** This class contains the logic for assigning agents based on complaint type.

2. Apex Triggers (Before/After Insert, Update, Delete)

Triggers execute **automatically** before or after data changes occur in Salesforce objects.

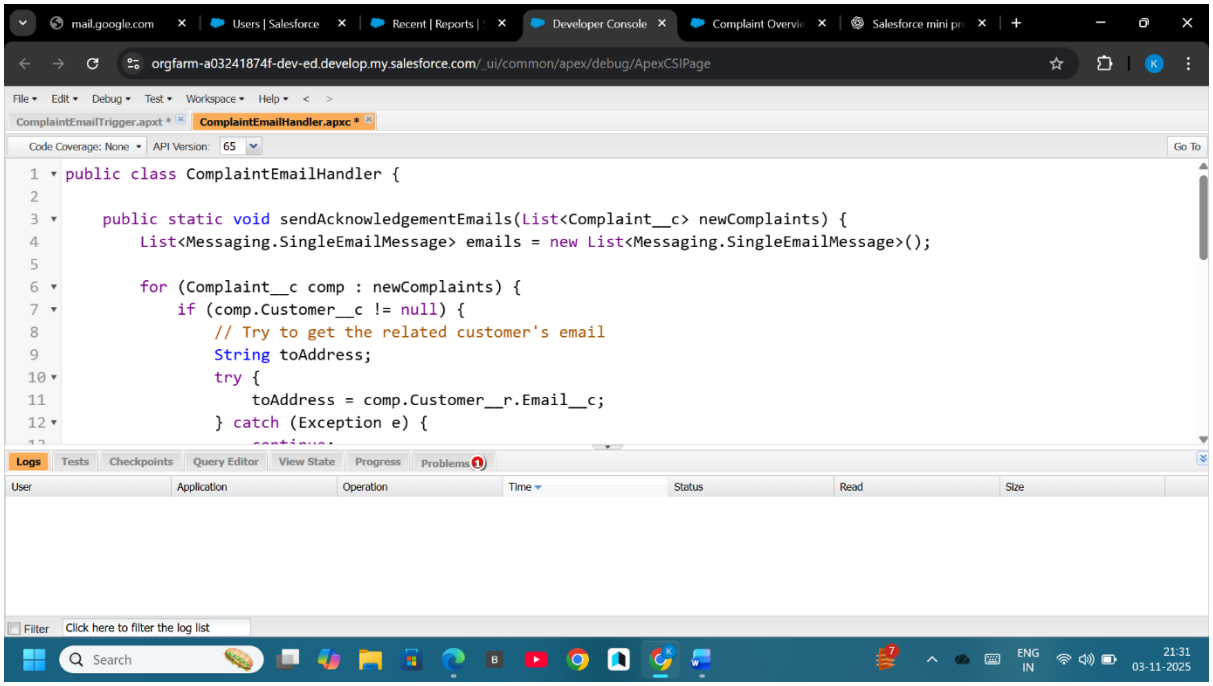
Trigger Types:

- **Before Insert / Before Update:** Modify field values before saving.
- **After Insert / After Update:** Perform actions after record is saved.
- **Before Delete / After Delete:** Clean up related data when deleted.

Example Trigger – Auto Assign Agent

```
trigger ComplaintTrigger on Complaint__c (before insert) {
    for(Complaint__c c : Trigger.new) {
        if(c.Issue_Type__c == 'Technical')
            c.Assigned_Agent__c = '0055g00000ABCDEF';
        else if(c.Issue_Type__c == 'Billing')
            c.Assigned_Agent__c = '0055g00000XYZABC';
        c.Status__c = 'Assigned';
    }
}
```

✅ **Use:** Automatically assigns an agent when a complaint is created.



3. Trigger Design Pattern

To avoid logic duplication and errors, follow the **Trigger Framework pattern**:

- Create **one trigger per object**
- Call logic from **Handler Class**

Example:

```
// Trigger
trigger ComplaintTrigger on Complaint__c (before insert, before update) {
    ComplaintTriggerHandler.run(trigger.new);
}


// Handler
public class ComplaintTriggerHandler {
    public static void run(List<Complaint__c> compList) {
        for(Complaint__c c : compList) {
            if(c.Issue_Type__c == 'Technical') {
                c.Assigned_Agent__c = '0055g00000ABCDEF';
            }
        }
    }
```

}

✅ **Use:** Keeps triggers clean and easier to maintain.

4. SOQL & SOSL

SOQL (Salesforce Object Query Language)

Used to query records from Salesforce.

Example:

List<Complaint__c> openComplaints = [SELECT Id, Status__c FROM Complaint__c WHERE Status__c != 'Closed'];

SOSL (Salesforce Object Search Language)

Used to search text across multiple objects.

Example:

List<List<SObject>> results = [FIND 'Network Issue' IN ALL FIELDS RETURNING Complaint__c(Id, Name)];

✅ **Use:** Fetch and search complaint data efficiently.

5. Collections (List, Set, Map)

List

Stores ordered records.

List<String> issueTypes = new List<String>{'Technical','Billing','Service'};

Set

Stores unique values (no duplicates).

Set<String> priorities = new Set<String>{'High','Medium','Low'};

Map

Stores key-value pairs.

Map<Id, Complaint__c> complaintMap = new Map<Id, Complaint__c>([SELECT Id, Name FROM Complaint__c]);

✅ **Use:** For bulk processing and data organization.

6. Control Statements

Used to define logic and flow in Apex.

Example:

```
if(c.Priority__c == 'High') {
    sendEscalationEmail(c.Id);
} else {
    System.debug('Normal priority complaint');
```

```
}
```

✅ **Use:** Implement conditional logic for complaint escalation.

7. Batch Apex

Used to handle **large data volumes** (above governor limits).

Example:

```
global class CloseOldComplaintsBatch implements Database.Batchable<sObject> {
    global Database.QueryLocator start(Database.BatchableContext bc) {
        return Database.getQueryLocator('SELECT Id FROM Complaint__c WHERE Status__c = \'Resolved\'');
    }
    global void execute(Database.BatchableContext bc, List<Complaint__c> complaints) {
        for(Complaint__c c : complaints) c.Status__c = 'Closed';
        update complaints;
    }
    global void finish(Database.BatchableContext bc) {
        System.debug('Batch process complete');
    }
}
```

✅ **Use:** Automatically closes resolved complaints after a period.

8. Queueable Apex

Used for **asynchronous processing** with chaining support.

Example:

```
public class ComplaintFollowUpQueueable implements Queueable {
    public void execute(QueueableContext context) {
        // Send follow-up emails to customers
        System.debug('Follow-up task executed');
    }
}
```

✅ **Use:** To send follow-up emails in the background without slowing user operations.

9. Scheduled Apex

Runs code automatically at specified times.

Example:

```
global class ComplaintScheduler implements Schedulable {
    global void execute(SchedulableContext sc) {
        CloseOldComplaintsBatch batch = new CloseOldComplaintsBatch();
        Database.executeBatch(batch);
    }
}
```

To schedule:

- Go to **Setup → Apex Classes → Schedule Apex**
- Choose frequency (daily/weekly)

✅ **Use:** Daily complaint maintenance or SLA checks.

10. Future Methods

Used for asynchronous calls, especially for callouts or delayed actions.

Example:

```
public class ComplaintNotifier {
    @future
    public static void sendEmail(Id complaintId) {
        System.debug('Email sent for complaint: ' + complaintId);
    }
}
```

✅ **Use:** To send emails after record save without slowing the transaction.

11. Exception Handling

Used to prevent runtime errors and show meaningful messages.

Example:

```
try {
    update complaintList;
} catch(DmlException e) {
    System.debug('Error updating complaints: ' + e.getMessage());
}
```

✅ **Use:** Ensures graceful error handling.

12. Test Classes

Used to test Apex logic and ensure 75% or more code coverage (mandatory for deployment).

Example:

```
@isTest
public class ComplaintTriggerTest {
    @isTest
    static void testComplaintInsert() {
        Complaint__c c = new Complaint__c(Issue_Type__c='Technical', Priority__c='High');
        insert c;
        System.assertEquals('Assigned', c.Status__c);
    }
}
```

✅ **Use:** Ensures that triggers and classes work correctly before deployment.

13. Asynchronous Processing

Apex provides several asynchronous options:

Method	Use Case
@future	Send emails or perform small background tasks
Batch Apex	Handle large data operations
Queueable Apex	Flexible background jobs with chaining
Scheduled Apex	Run automation at specific intervals

✅ **Use:** Keeps system performance fast and avoids hitting limits.

Conclusion

In Phase 5, we implemented **custom logic using Apex programming**, enhancing the CRM's intelligence and automation.

✅ **Key Outcomes:**

- Automated complaint assignments using triggers
- Optimized data handling using SOQL/SOSL
- Used Batch, Queueable, and Scheduled Apex for performance
- Ensured quality through exception handling and test classes

This phase brings **developer-level customization** and efficiency to the **Customer Complaint Management System**.

Phase 6: User Interface Development

🎯 Objective

The goal of this phase is to design and implement the **user interface (UI)** for the **Customer Complaint Management System** using Salesforce **Lightning Experience**.
This ensures that users can **easily view, create, and manage complaints, customers, and reports** through an intuitive and interactive layout.

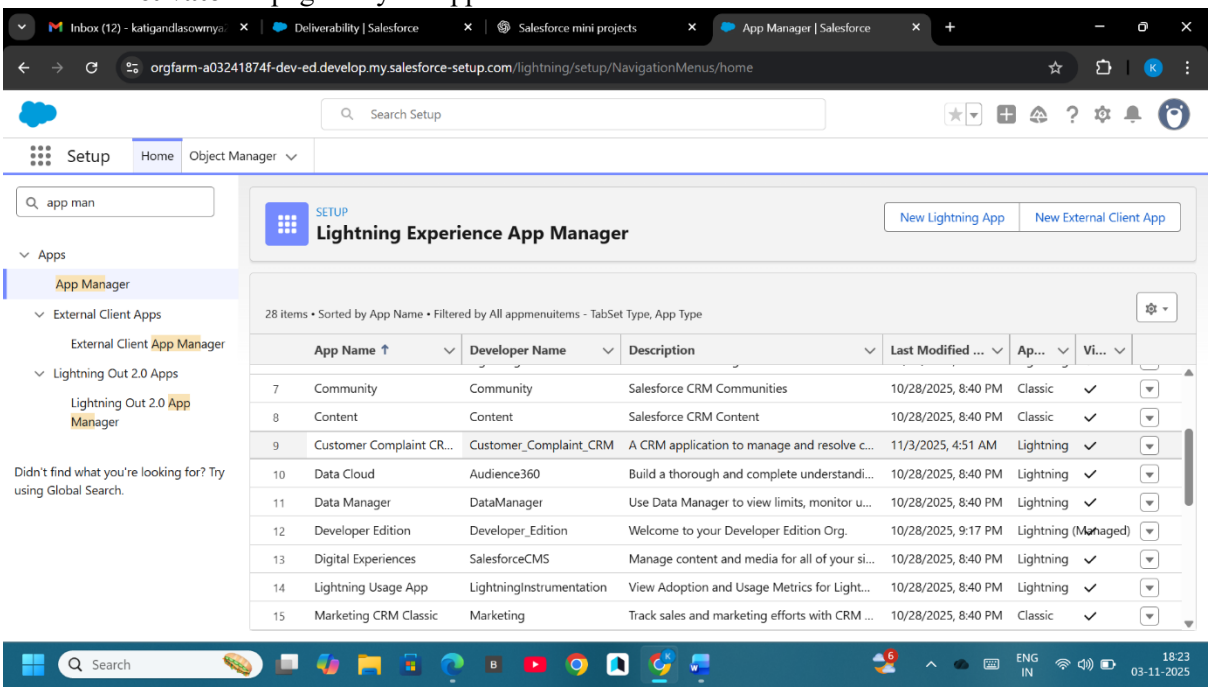
⚙️ 1. Lightning App Builder

Description:

Lightning App Builder allows developers to create and customize app pages visually without code. It is used to design record pages, home pages, and app pages for better user experience.

Steps:

1. Go to **Setup → Lightning App Builder → New**.
2. Choose **App Page / Record Page / Home Page**.
3. Add components like:
    - **Record Detail**
    - **Highlights Panel**
    - **Related Lists**
    - **Charts / Reports**
4. Save and **Activate** the page for your app.



Example:

- Created a **Complaint Record Page** showing key fields like:

- Complaint Number
- Status
- Priority
- Assigned Agent
- Related Customer Details

📄 2. Record Pages

Description:

Record Pages display all details of a specific record in a structured format.
You can customize layouts to highlight the most important information.

Example:

When a user opens a complaint:

- Header shows **Complaint Number**, **Priority**, **Status**
- Tabs display **Details**, **Related Complaints**, **Notes**, **History**

🗂 3. Tabs

Description:

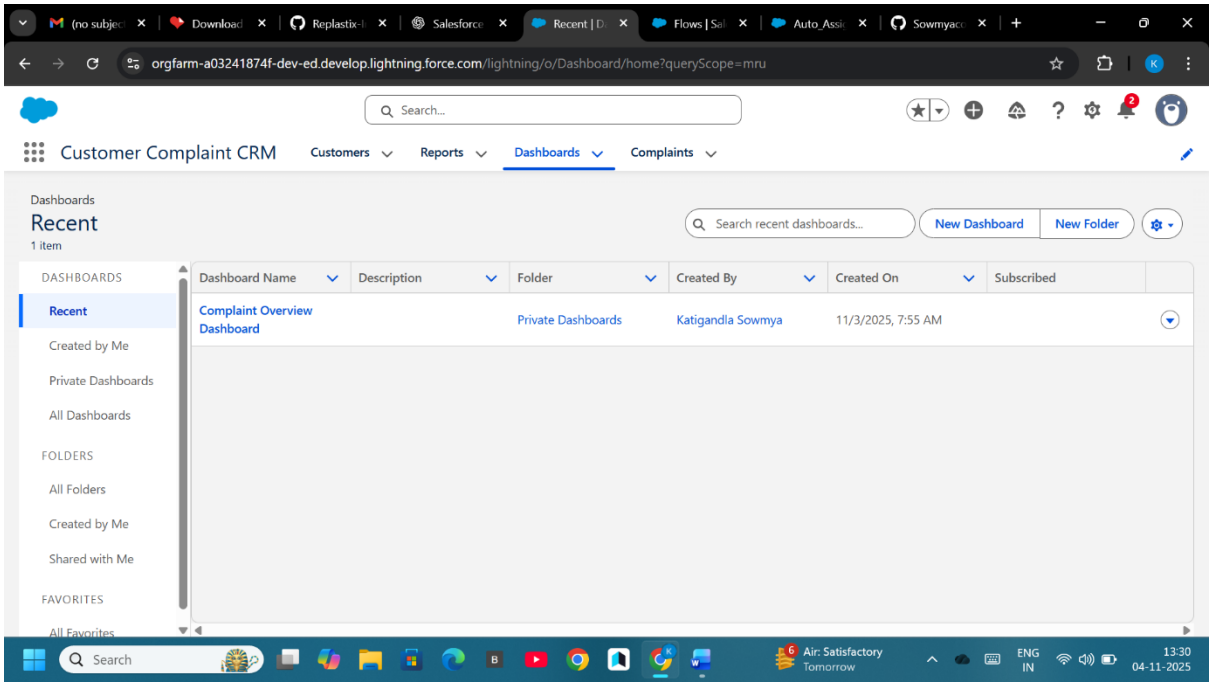
Tabs are used for easy navigation within the app.
Each object (Customer, Complaint, Reports) has its own tab.

Example:

Tabs added:

- **Customers**
- **Complaints**
- **Reports**
- **Dashboards**

So users can quickly move between data sections



🏠 4. Home Page Layout

Description:

The Home Page is the main dashboard of the app, showing important metrics and shortcuts.

Example Layout:

- Total Complaints Count
- Open Complaints
- High-Priority Complaints
- Pie Chart (Complaints by Status)
- Shortcut to “Create New Complaint”

Created using **Lightning App Builder → Home Page → Add Dashboard Components.**

🧰 5. Utility Bar

Description:

The Utility Bar provides quick access to tools that are always available at the bottom of the screen.

Example:

Added:

- “Quick Complaint” Button → to create a complaint directly
- “Notes” → for agents to take quick notes
- “Recent Items” → to access last viewed records easily

⚡ 6. Lightning Web Components (LWC)

Description:

Lightning Web Components (LWC) are reusable, lightweight, and fast components built using **HTML, JavaScript, and Apex**.
They help create dynamic interfaces.

Example Component:

MyComplaintsDashboard
Displays complaints assigned to the logged-in user.

Sample Apex Controller:

```
public with sharing class MyComplaintsController {
    @AuraEnabled(cacheable=true)
    public static List<Complaint__c> getMyComplaints() {
        return [SELECT Id, Name, Status__c, Priority__c
                FROM Complaint__c
                WHERE Assigned_Agent__c = :UserInfo.getUserId()];
    }
}
```

Sample LWC (myComplaintsDashboard.js):

```
import { LightningElement, wire } from 'lwc';
import getMyComplaints from '@salesforce/apex/MyComplaintsController.getMyComplaints';

export default class MyComplaintsDashboard extends LightningElement {
    @wire(getMyComplaints) complaints;
}
```

Template (myComplaintsDashboard.html):

```
<template>
    <lightning-card title="My Complaints">
        <template if:true={complaints.data}>
            <lightning-datatable
                key-field="Id"
                data={complaints.data}
                columns={[
                    { label: 'Complaint Name', fieldName: 'Name' },
                    { label: 'Status', fieldName: 'Status__c' },
                    { label: 'Priority', fieldName: 'Priority__c' }
```

```
            ]}>
          </lightning-datatable>
        </template>
        <template if:true={complaints.error}>
          <p>Error loading complaints</p>
        </template>
    </lightning-card>
</template>
```

□ 7. Apex Integration with LWC

Description:

LWC components use **Apex** to access Salesforce data when logic is complex.
This allows two-way communication between frontend (LWC) and backend (Apex).

Example:

When user clicks “Resolve Complaint” → Apex updates complaint status to “Resolved”.

🔁 8. Events in LWC

Description:

Events are used to communicate between components.

- **Child → Parent** using custom events
- **Parent → Child** using public properties

Example:

When complaint is resolved → event triggers dashboard refresh.

📡 9. Wire Adapters

Description:

The @wire service is used to automatically fetch Salesforce data from standard objects or Apex methods.

Example:

```
@wire(getRecord, { recordId: '$recordId', fields: ['Complaint__c.Status__c'] })
complaint;
```

□ 10. Imperative Apex Calls

Description:

Imperative calls let developers manually execute Apex methods (e.g., after clicking a button).

Example:

```
handleAssignAgent() {
    assignAgent({ complaintId: this.recordId })
        .then(() => alert('Agent Assigned Successfully'))
        .catch(error => console.log(error));
}
```

🚀 11. Navigation Service

Description:

Navigation Service is used to open records, list views, or pages from LWC code.

Example:

```
this[NavigationMixin.Navigate]({
    type: 'standard__recordPage',
    attributes: {
        recordId: complaintId,
        objectApiName: 'Complaint__c',
        actionName: 'view'
    }
});
```

Phase 7: Integration & External Access

Objective

The objective of this phase is to **enable communication between Salesforce and external systems** (like websites, ERP, or customer portals) using secure and scalable integration methods.
This ensures that complaint and customer data can be shared, updated, and synchronized across platforms seamlessly.

1. Named Credentials

Description:

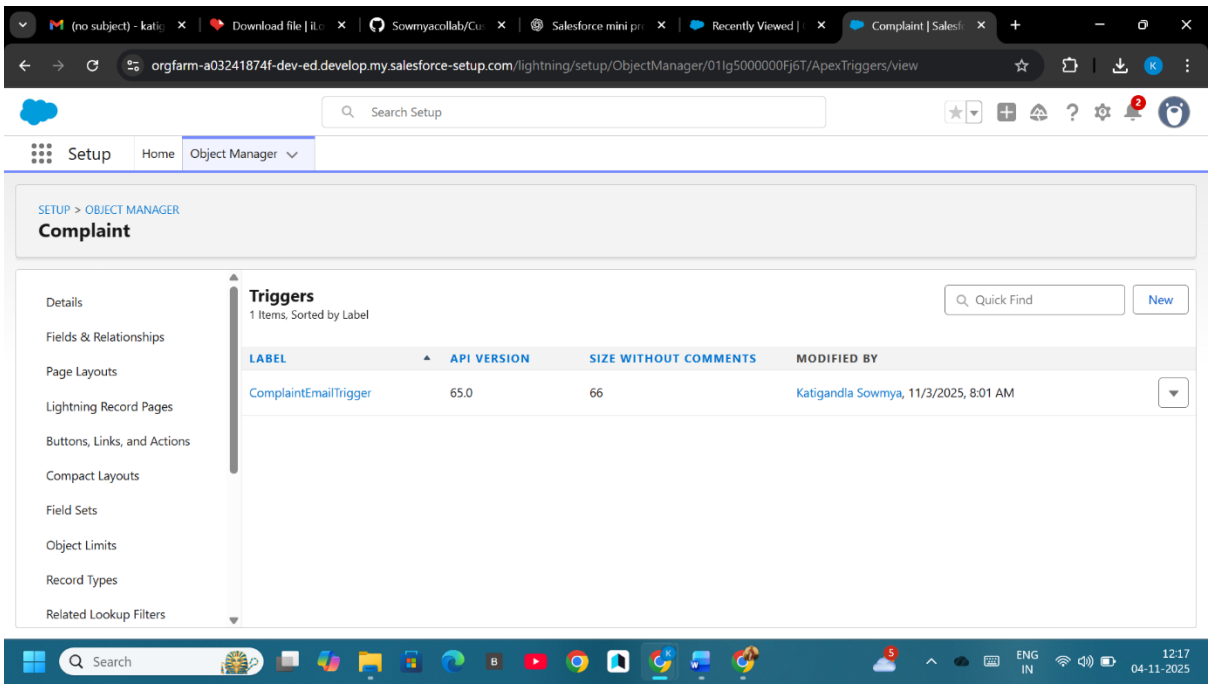
Named Credentials in Salesforce are used to **store external system credentials (like API URLs, usernames, passwords, or OAuth tokens)** in a secure and managed way.

Purpose:

- Simplifies API callouts by avoiding hardcoding authentication details.
- Centralizes management of credentials for integration endpoints.

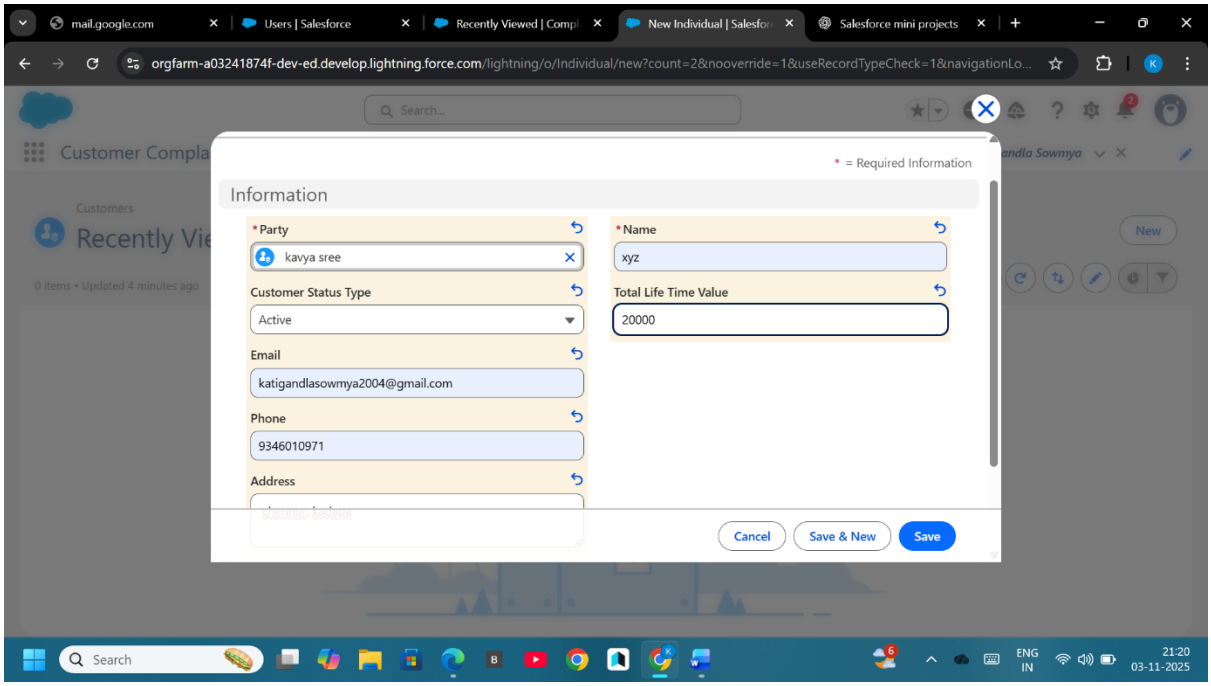
Example:

1. Go to **Setup → Named Credentials → New Named Credential**
2. Enter:
    - **Label:** External Complaint API
    - **URL:** https://externalcomplaintsapi.com
    - **Authentication:** Password or OAuth 2.0
3. Save.



Usage in Apex:

```
HttpRequest req = new HttpRequest();
req.setEndpoint('callout:External_Complaint_API/complaints');
req.setMethod('GET');
HttpResponse res = new Http().send(req);
System.debug(res.getBody());
```



🌐 2. External Services

Description:

External Services allow you to connect **Salesforce Flows** with **external REST APIs** automatically using **Schema or OpenAPI specifications**.

Purpose:

- No-code integration
- Used to send complaint data to external tracking systems.

Example:

- Import an **OpenAPI schema** of an external complaint service.
- Define actions like **"Create External Ticket"** in Flow when a new complaint is created.

💻 3. Web Services (REST / SOAP)

Description:

Salesforce can **consume (use)** or **expose (provide)** web services to integrate with other applications.

A. REST API Integration:

Used for lightweight, modern integrations with JSON.

Example (Apex REST Callout):

HttpRequest req = new HttpRequest();

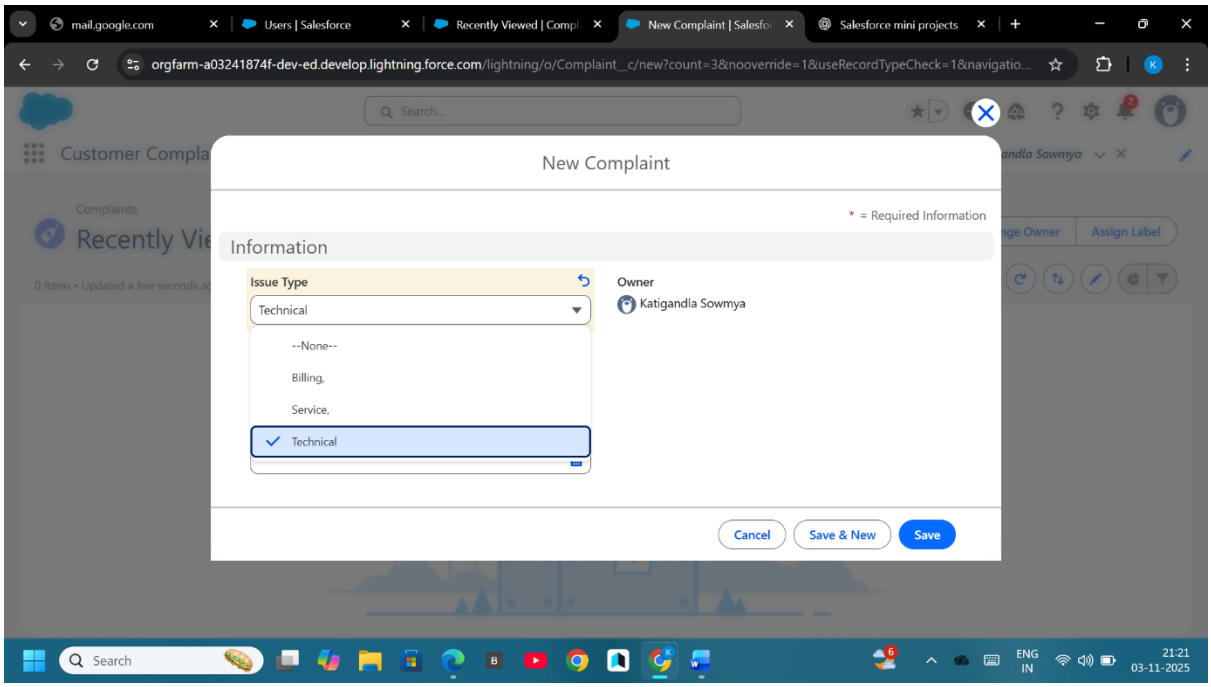
req.setEndpoint('https://externalapi.com/support');

req.setMethod('POST');

req.setHeader('Content-Type', 'application/json');

req.setBody('{"ComplaintId":"C001","Status":"New"}');

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



B. SOAP Web Services:

Used for enterprise systems (ERP/legacy apps).

Example:
Salesforce exposes its **WSDL** so external systems can call its services to create or update complaints.

🔄 4. Callouts

Description:

Apex **Callouts** are used to send requests to external systems (e.g., notify a third-party CRM when a complaint is resolved).

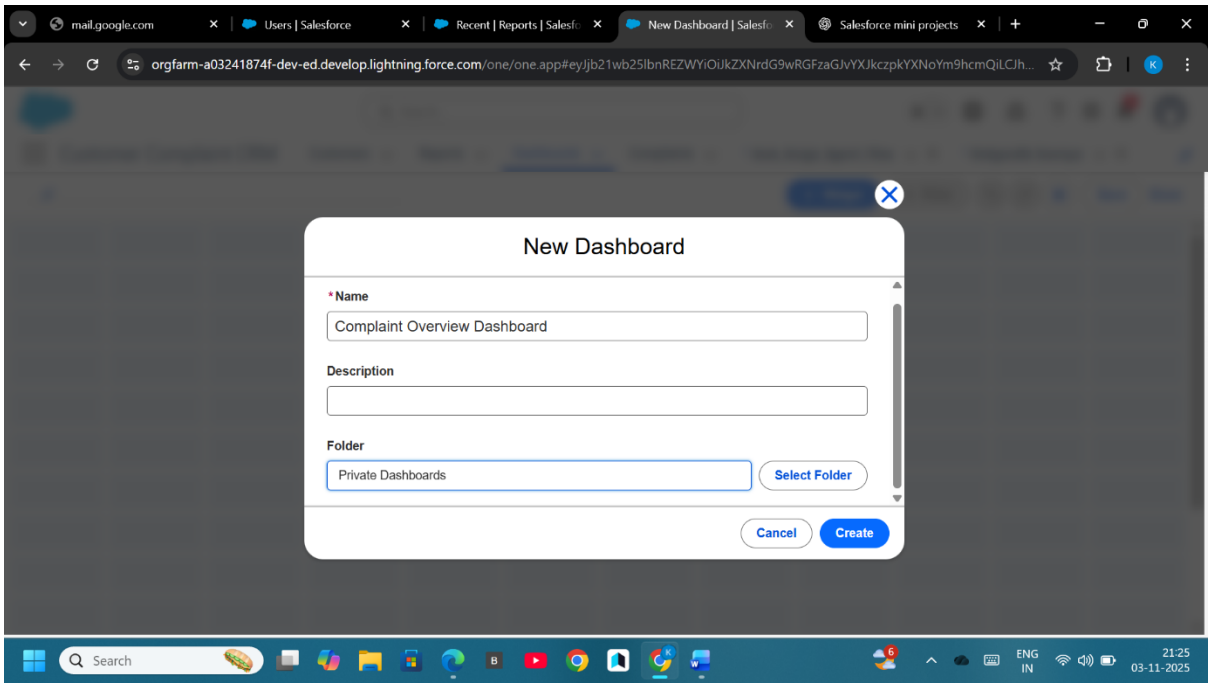
Example Use Case:

When Complaint Status = "Resolved" → Send a callout to external service to update complaint record.

```
public class ComplaintCallout {

    @future(callout=true)
```

```
    public static void notifyExternal(String complaintId) {
        Complaint__c comp = [SELECT Name, Status__c FROM Complaint__c WHERE Id =
:complaintId];
        HttpRequest req = new HttpRequest();
        req.setEndpoint('callout:External_Complaint_API/notify');
        req.setMethod('POST');
        req.setBody(JSON.serialize(comp));
        new Http().send(req);
    }
}
```



⚙ 5. Platform Events

Description:

Platform Events are Salesforce's **event-driven architecture** to communicate changes between Salesforce and other systems **in real-time**.

Use Case:

When a new complaint is created → Publish an event "ComplaintCreatedEvent" → External systems subscribe to receive the data.

Example:

```
Complaint_Event__e event = new Complaint_Event__e(
    ComplaintId__c = 'C001',
    Status__c = 'New'
```

);

Database.SaveResult result = EventBus.publish(event);

🔁 6. Change Data Capture (CDC)

Description:

Change Data Capture automatically tracks **create, update, delete** operations and shares these changes to external systems in real-time.

Use Case:

Whenever a **Complaint__c** record is updated, the change is sent to an external service or data warehouse.

Steps:

1. Go to **Setup → Change Data Capture**
2. Select **Complaint__c** object.
3. Save and subscribe using Salesforce API or middleware (like MuleSoft).

🌍 7. Salesforce Connect

Description:

Salesforce Connect allows you to **view and work with data stored in external systems** as if it were native Salesforce data — without importing it.

Use Case:

- View external **Customer** or **Complaint** records stored in an ERP or Service Database.

Steps:

1. Setup → **External Data Source → New**
2. Type: **OData 4.0**
3. Enter URL: (e.g., https://externalcrm.com/odata)
4. Validate and **Sync** tables → External Objects appear in Salesforce.

📊 8. API Limits

Description:

Salesforce imposes daily API usage limits depending on the **edition and license**.

Use Case:

Monitor API calls when external integrations are active to avoid exceeding limits.

Check Limits:

- Go to **Setup → System Overview**
- Or query:

SELECT Name, DailyApiRequests, RemainingApiRequests FROM Organization

🔒 9. OAuth & Authentication

Description:

OAuth 2.0 provides a secure way to connect Salesforce with external systems without exposing passwords.

Use Case:

Allow external apps (like a chatbot or website) to access Salesforce data securely.

Steps:

1. Setup → **App Manager → New Connected App**
2. Enable **OAuth Settings**
3. Add Callback URL & Scopes
4. Use Consumer Key/Secret in external application

🌐 10. Remote Site Settings

Description:

Salesforce blocks external callouts by default for security.
You must whitelist external URLs using **Remote Site Settings**.

Steps:

1. Setup → **Remote Site Settings → New Remote Site**
2. Remote Site Name: Complaint_API
3. URL: https://externalcomplaintsapi.com
4. Save.

Now Apex callouts to that URL are allowed.

📘 Phase 8: Data Management & Deployment

🎯 Objective

The goal of this phase is to **manage data effectively** and **deploy Salesforce components** safely between environments.
This includes **data import/export, duplication control, and deployment automation** using Change Sets, Packages, or VS Code (SFDX).

📤 1. Data Import Wizard

Description:

Data Import Wizard is a **browser-based tool** in Salesforce that helps you **import simple data** like Leads, Accounts, Contacts, or Custom Objects.

Use Case:

Used to upload **Customer** or **Complaint** records from Excel/CSV files.

Steps:

1. Go to **Setup → Data Import Wizard**.
2. Click **Launch Wizard**.
3. Choose the **Object** (e.g., Customer or Complaint).
4. Upload a **.CSV file**.
5. Map fields → Start Import.

Example:

Importing a CSV file with columns:

Customer_Name, Email, Phone, Address

After import, records automatically appear in **Customer__c**.

⚙️ 2. Data Loader

Description:

Data Loader is a **desktop application** used for **large data volumes** and **complex operations** (insert, update, upsert, delete, export).

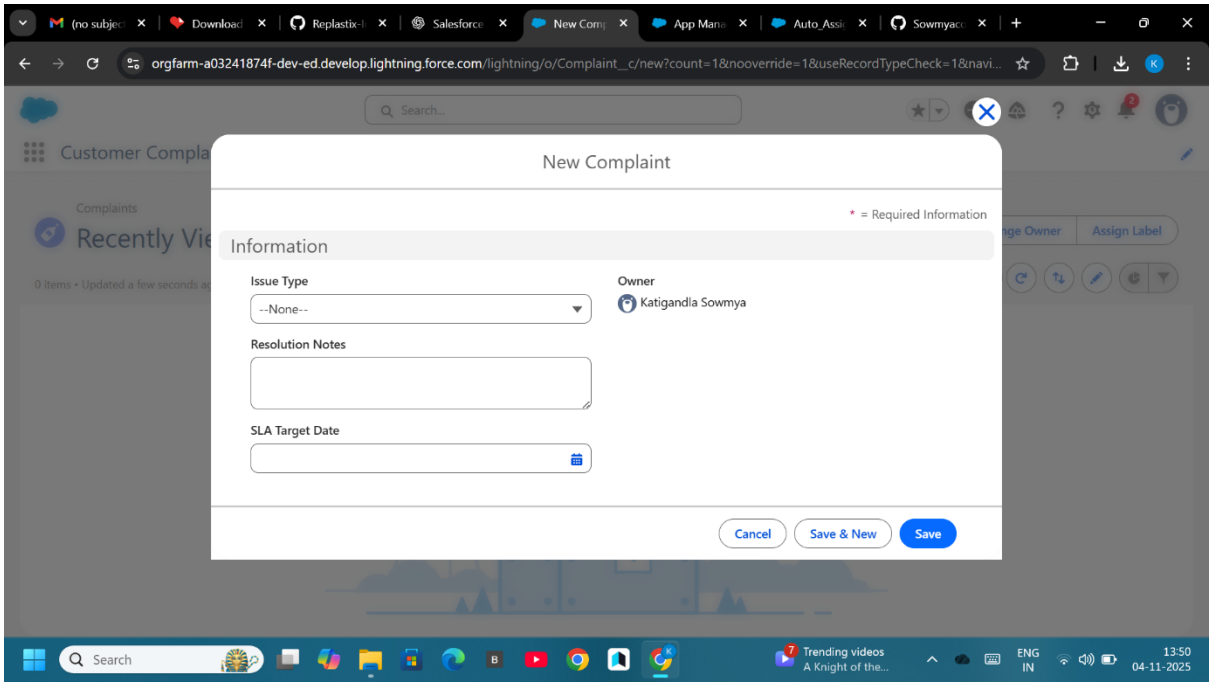
Use Case:

Used to import/export thousands of complaint records or migrate data between orgs.

Steps:

1. Download **Data Loader** from Setup → Data Loader.
2. Login with Salesforce credentials.
3. Choose operation (Insert/Update/Export/Delete).
4. Select Object (e.g., Complaint__c).

5. Upload CSV file and map fields.
6. Execute → Review success and error logs.



🧩 3. Duplicate Rules

Description:

Duplicate Rules prevent creating records with the same key values (like same Email or Phone).

Use Case:

Avoid duplicate customers in CRM.

Steps:

1. Setup → **Duplicate Rules** → **New Rule**.
2. Object: **Customer__c**.
3. Matching Rule: Email = Email.
4. Action: **Block or Alert** user when duplicate found.
5. Activate Rule.

💾 4. Data Export & Backup

Description:

Salesforce allows **manual and scheduled backups** of data for recovery or migration.

Steps:

1. Setup → **Data Export**.
2. Click **Export Now** or **Schedule Export** (weekly/monthly).
3. Select Objects (Customer, Complaint, etc.).

4. Salesforce emails a **ZIP file** containing CSV data backup.

Best Practice:

Schedule weekly exports for safety.

🚀 5. Change Sets

Description:

Change Sets are the **standard Salesforce deployment tool** used to move metadata (fields, objects, flows, validation rules, etc.) between related orgs (e.g., Sandbox → Production).

Steps:

1. Go to **Setup → Outbound Change Sets → New**.
2. Add Components (Objects, Fields, Flows, Tabs, etc.).
3. Upload to Target Org (e.g., Production).
4. In Target Org → **Inbound Change Sets → Deploy**.

Use Case:

Deploy the **Customer Complaint CRM** app from Developer Org to Sandbox or Production.

📦 6. Managed vs Unmanaged Packages

Description:

Type	Purpose	Usage
Managed Package	Locked, versioned, upgradable (AppExchange apps)	For production apps
Unmanaged Package	Editable, for learning and sharing	For student or project sharing

Use Case:

For this project, use an **Unmanaged Package** to share your app with your mentor or classmates.

Steps:

1. Setup → **Packages → New Package**.
2. Add components: Objects, Tabs, Reports, Dashboards, Flows.
3. Upload → Salesforce generates a link to share.

💡 7. ANT Migration Tool

Description:

ANT (Apache Ant) is a **command-line deployment tool** used for automating metadata deployments between orgs.

Use Case:

Used in professional environments for CI/CD automation (Continuous Integration & Delivery).

Steps:

1. Install **Salesforce ANT Migration Tool**.
2. Create **build.xml** and **package.xml** files.
3. Run command:
4. ant retrieve
5. ant deploy
6. ANT retrieves or deploys components between orgs using API credentials.

🗃️💻 8. VS Code & Salesforce DX (SFDX)

Description:

Salesforce DX (Developer Experience) with **Visual Studio Code** is a modern development approach for managing metadata and code.

Features:

- Source-driven development
- Version control (Git)
- Fast deployment to scratch orgs or sandboxes

Steps:

1. Install **VS Code** and **Salesforce Extensions Pack**.
2. Authenticate Org:
3. sfdx auth:web:login -d -a DevHub
4. Create Project:
5. sfdx force:project:create -n ComplaintCRM
6. Retrieve metadata:
7. sfdx force:source:retrieve -m ApexClass, CustomObject
8. Deploy changes:
9. sfdx force:source:deploy -p force-app

Use Case:

Move Apex classes, Flows, and LWC from local project to Salesforce org quickly.

🚀 Outcome

After completing Phase 8:

- Data is **accurate, secure, and backed up**.
- The **Customer Complaint CRM** can be **deployed or shared** easily between orgs.
- The project follows **professional DevOps and deployment standards**, ready for real-time implementation.

📘 Phase 9: Reporting, Dashboards & Security Review

🎯 Objective

The purpose of this phase is to create **meaningful business insights** through Salesforce **Reports and Dashboards** while ensuring **data security and compliance** using sharing rules, field-level security, and login restrictions.

This phase ensures that the **Customer Complaint CRM** application delivers powerful analytics while keeping customer and agent data safe.

📊 1. Reports

Description:

Reports in Salesforce help visualize and analyze CRM data such as **complaints, customers, agents, and performance**.

Types of Reports:

Type	Description	Use Case
Tabular Report	Simple table view, like Excel	List of all complaints
Summary Report	Grouped data with subtotals	Complaints grouped by Issue Type
Matrix Report	Grouped by rows and columns	Complaints by Agent and Status
Joined Report	Combines multiple report types	Compare Complaints and Customer data

Example:

Report Name: Complaints by Priority

- Object: Complaint__c
- Group by: Priority
- Show fields: Complaint Number, Issue Type, Status, Assigned Agent
- Add filter: Status != Closed

☑ **Outcome:** Quickly identify high-priority issues that are still open.

📋 2. Report Types

Description:

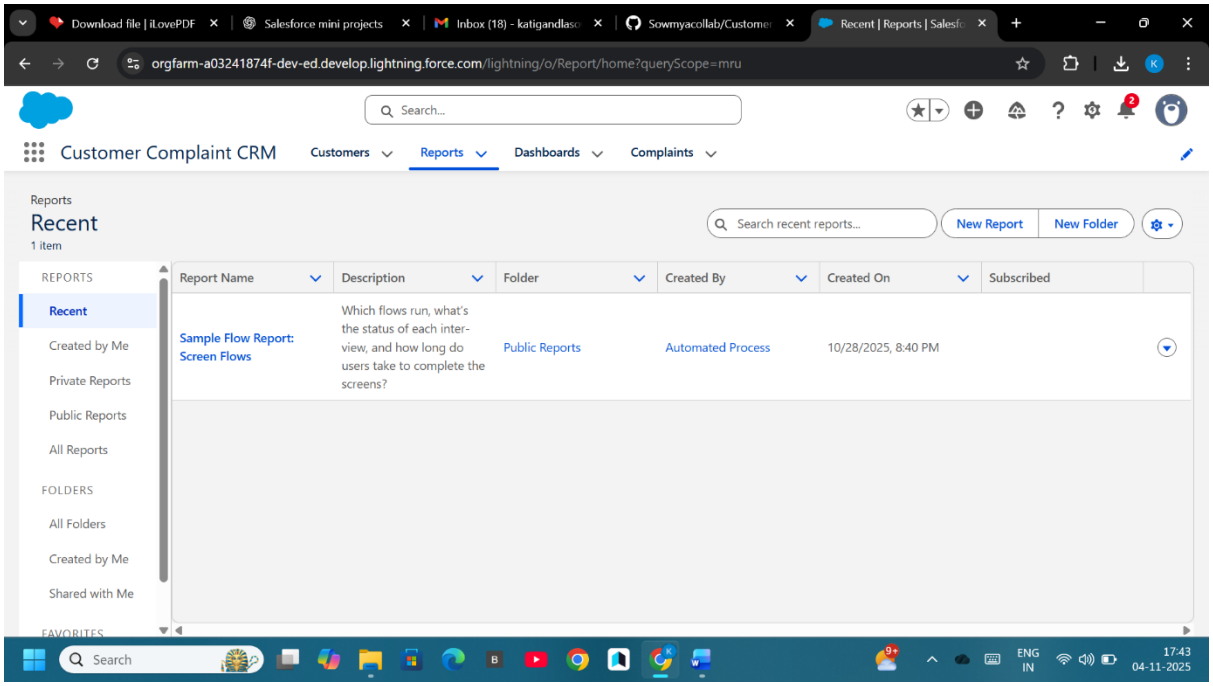
Report Types define which objects and fields are available for reporting.

Steps:

1. Setup → **Report Types** → **New Custom Report Type**
2. Primary Object: **Complaint__c**
3. Related Object: **Customer__c** (Each complaint has a customer)
4. Deploy the report type.

Use Case:

Allows reports combining both **Complaint and Customer** details (e.g., Complaints per Customer).



☑ 3. Dashboards

Description:

Dashboards visually display multiple reports in one place using charts, graphs, and tables.

Steps to Create:

1. Go to **Dashboards → New Dashboard**
2. Name: **Customer Complaint Dashboard**
3. Add components:
    - **Bar Chart:** Complaints by Status
    - **Pie Chart:** Complaints by Priority
    - **Gauge:** Agent Performance (% complaints closed)
    - **Table:** Top 5 Customers with Most Complaints
4. Save & Run.

Use Case:

Managers can monitor the CRM's overall performance in real time.

⚙ 4. Dynamic Dashboards

Description:

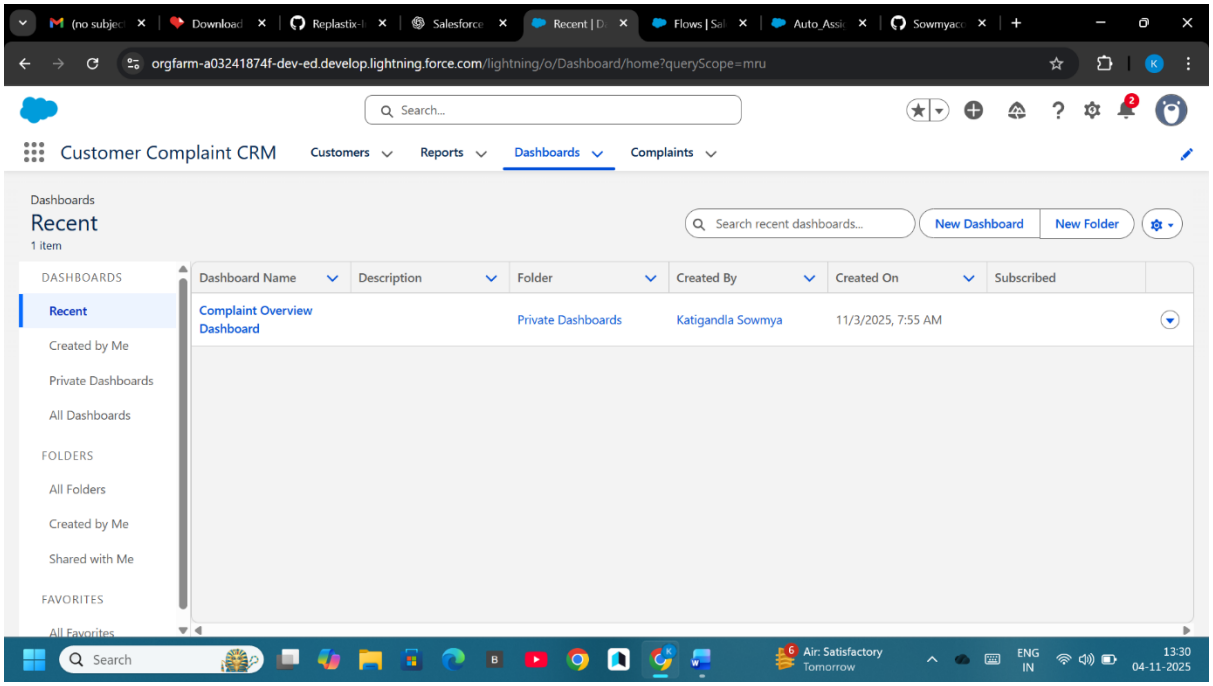
Dynamic Dashboards allow users to view data **according to their access level** — e.g., each agent only sees their own complaints.

Steps:

1. Open Dashboard → **Edit Properties**
2. Select: “View Dashboard As: Logged-in User”
3. Save & Activate

Use Case:

Ensures privacy — one agent can’t see another’s data.



🔒 5. Sharing Settings

Description:

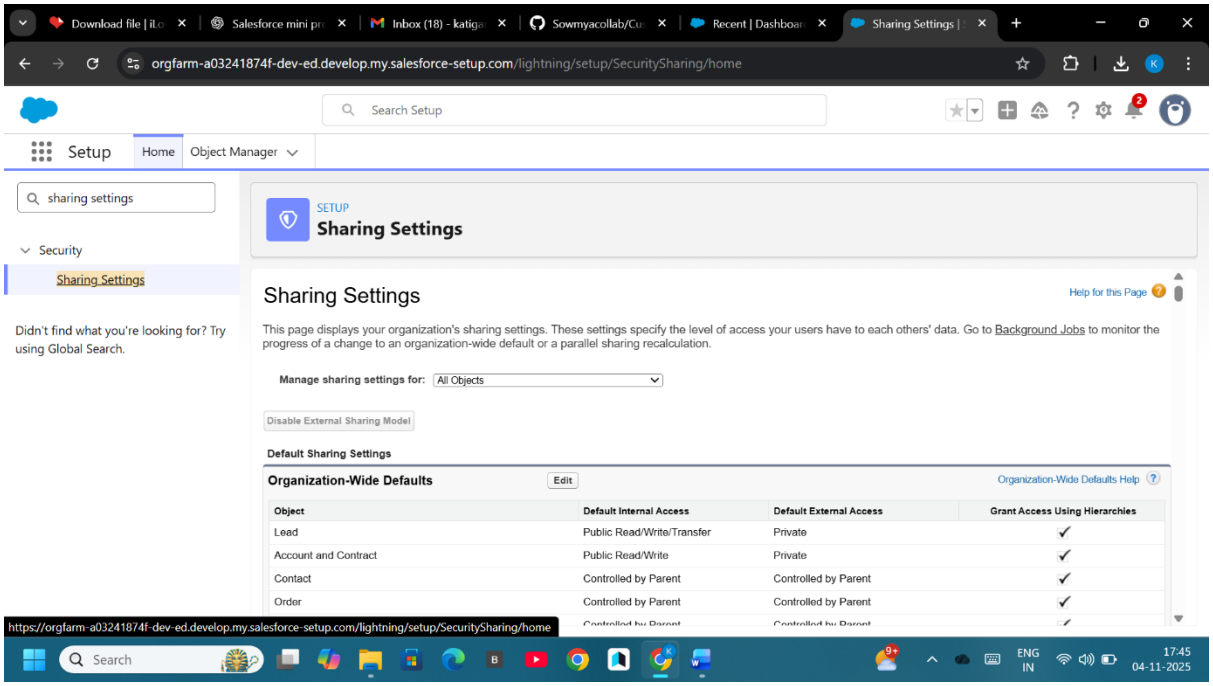
Sharing Settings define **organization-wide data visibility** and **record access** between users.

Steps:

1. Setup → **Sharing Settings**
2. Set **Organization-Wide Defaults (OWD):**
    - Complaint__c: Private
    - Customer__c: Controlled by Parent
3. Add **Sharing Rules:**
    - Share complaints with same department users.

Use Case:

Agents can view only their assigned complaints, while admins see all.



6. Field-Level Security (FLS)

Description:

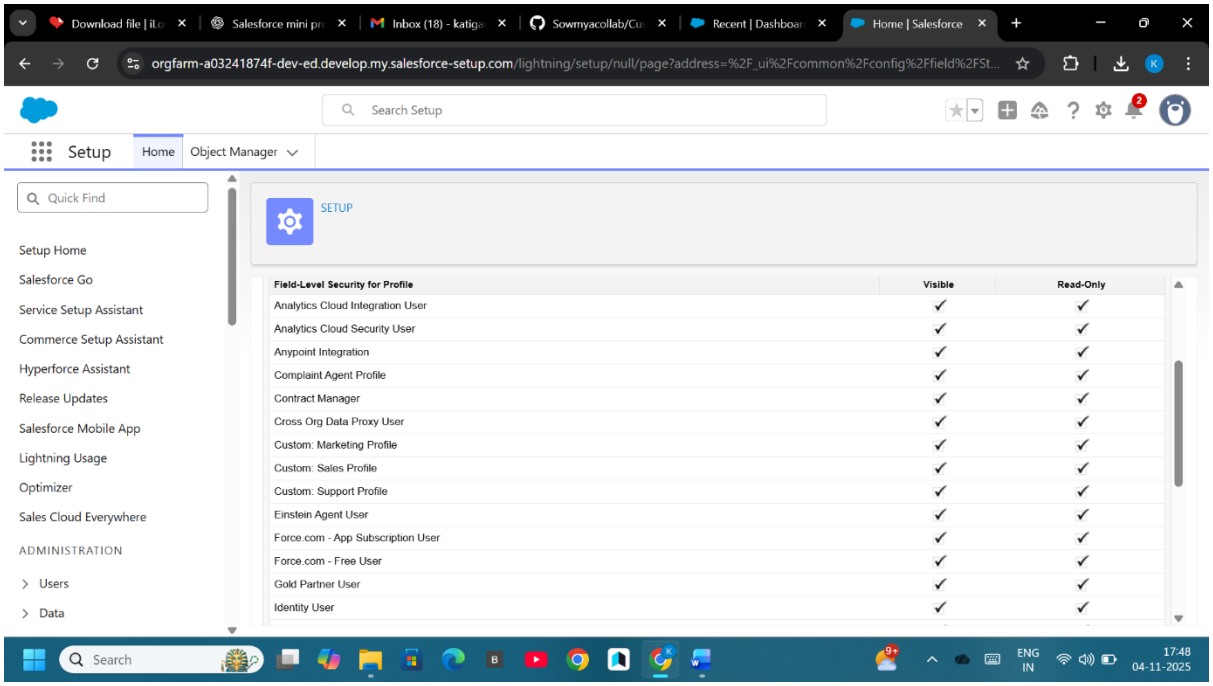
Controls which fields a user can **view or edit** in an object.

Steps:

1. Setup → Object Manager → **Complaint__c** → **Fields & Relationships**
2. Click a field (e.g., “Customer Email”).
3. Set **Field-Level Security** for each Profile (hide from agents if needed).

Use Case:

Hide sensitive information (like billing details) from regular support agents.



🛡️ 7. Session Settings

Description:

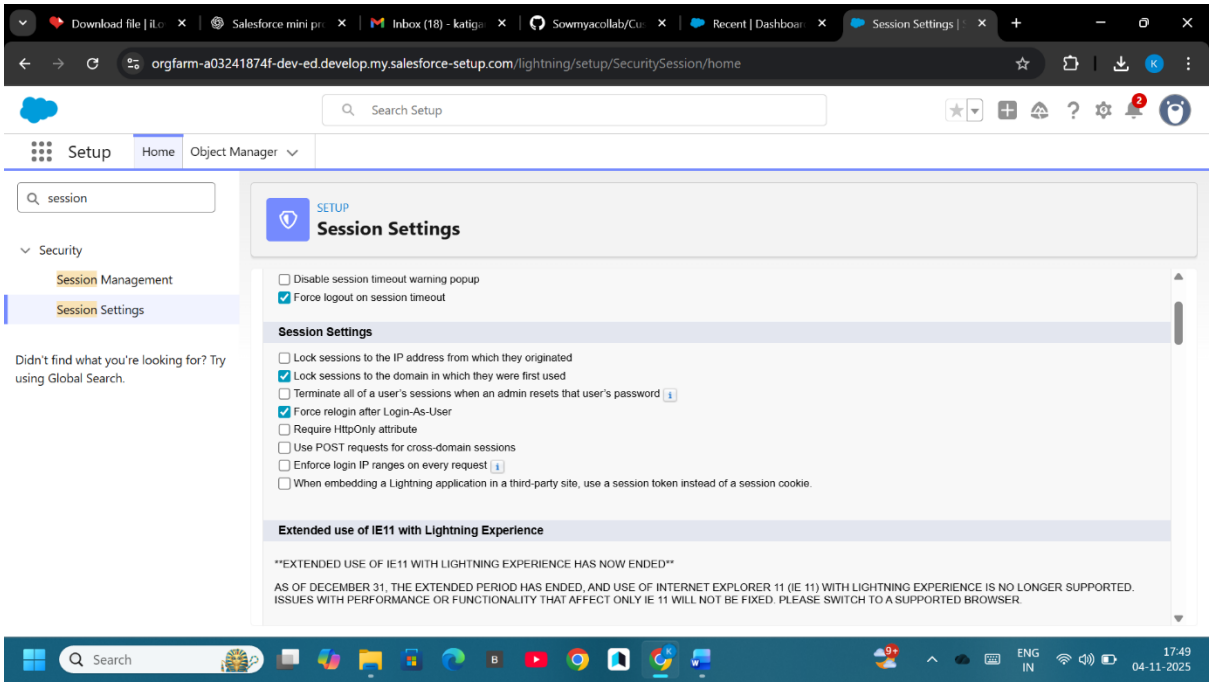
Controls session timeouts, login hours, and authentication settings to improve security.

Steps:

1. Setup → **Session Settings**
2. Configure:
    - Session Timeout: 30 minutes
    - Disable “Stay Logged In”
    - Enable secure HTTPS connections

Use Case:

Prevents unauthorized access if a session is left open.



🌐 8. Login IP Ranges

Description:

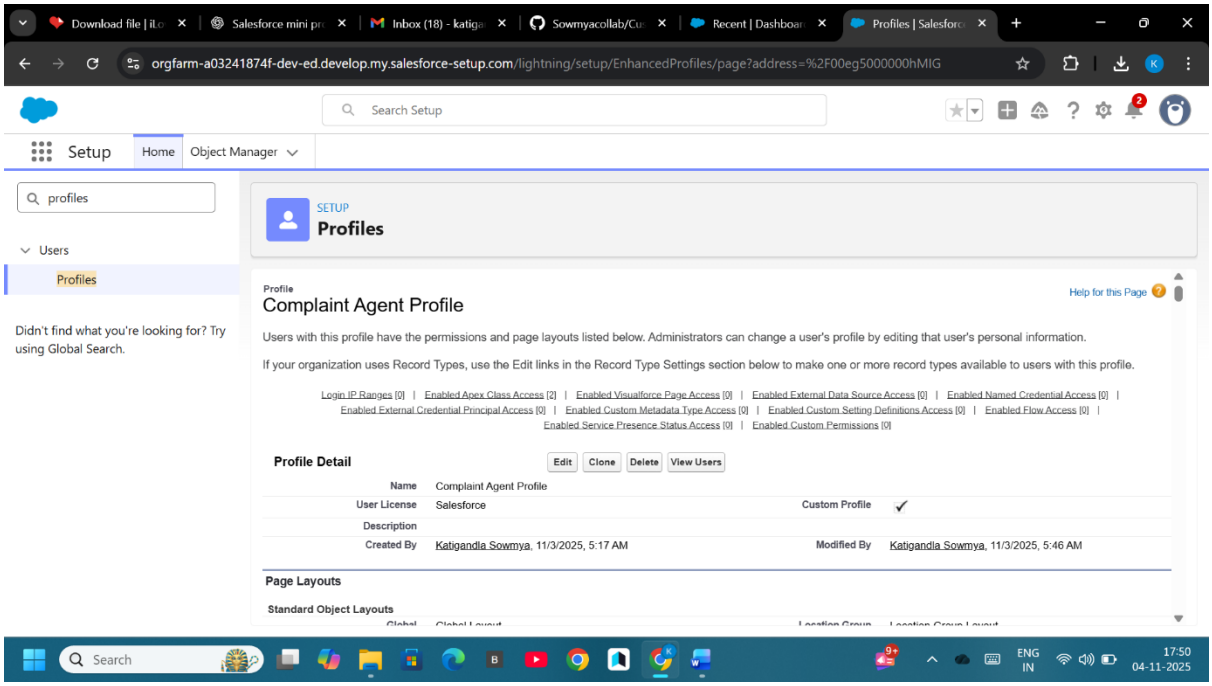
Restricts logins to specific IP ranges to prevent external access.

Steps:

1. Setup → **Profiles** → **Select Profile (e.g., Support Agent)**
2. Scroll to **Login IP Ranges** → **New**
3. Define allowed range (e.g., company's internal network IPs).

Use Case:

Only allow users to log in from office or company-approved networks.



9. Audit Trail

Description:

Tracks configuration changes and admin activities.

Steps:

1. Setup → **View Setup Audit Trail**
2. Review the list of changes (who changed what and when).

Use Case:

Monitor admin changes to fields, flows, security settings, etc.

🚀 Outcome

After completing **Phase 9**:

- You can generate detailed **reports and dashboards** for complaints and customers.
- Security configurations ensure that **only authorized users** can view or edit data.
- The system becomes **transparent, secure, and audit-ready**, ideal for real business CRM operations.

📘 Phase 10: Final Presentation & Demo Day

🎯 Objective

The goal of this phase is to **present, demonstrate, and deliver** the completed *Customer Complaint CRM Project* to evaluators or stakeholders.
This includes a **pitch presentation**, a **live demo walkthrough**, collecting **feedback**, and preparing **handoff documentation** for future use or deployment.

🎤 1. Pitch Presentation

Description:

The pitch presentation summarizes the project journey — from idea to implementation — highlighting the **problem solved**, **features built**, and **impact** created.

Contents of Pitch Deck:

Slide	Content	Example
1	**Title Slide**	Project Title: *Customer Complaint CRM*
2	**Problem Statement**	Companies struggle to manage customer complaints efficiently.
3	**Proposed Solution**	A centralized CRM to track, assign, and resolve complaints automatically.
4	**Key Features**	Auto-assignment, Email acknowledgment, SLA tracking, Dashboards
5	**Technical Stack**	Salesforce Platform, Flows, Apex, LWC
6	**Demo Overview**	Real-time flow of complaint creation to resolution
7	**Benefits / Impact**	Reduced manual work, improved customer satisfaction
8	**Team & Roles**	Developer, Tester, Documentation, Presenter
9	**Future Enhancements**	Chatbot integration, AI-based priority detection
10	**Conclusion & Thank You**	Summary & closing note

Tools Used:

- PowerPoint / Google Slides
- Salesforce Screenshots & Recorded Demos

🧷 **Tip:** Practice the 3-minute elevator pitch:

"Our project, *Customer Complaint CRM*, automates the end-to-end complaint handling process using Salesforce — from capturing issues to assigning agents, sending emails, and tracking SLAs through dynamic dashboards."

💻 2. Demo Walkthrough

Description:

A live demonstration to showcase how the system works from the end user's perspective.

Steps in Demo:

Step	Action	Expected Result
1	Login to Salesforce App: *Customer Complaint CRM*	Home page opens
2	Create a new **Customer** record	Customer created successfully
3	Create a new **Complaint** record	Auto-assigns agent, sends acknowledgment email
4	Open **Complaint Record**	Assigned_Agent__c and Status = Assigned
5	Show **Reports** and **Dashboard**	Visual representation of complaint data
6	Demonstrate **Validation Rules**	Error shown if Customer field left blank
7	Show **Security Settings**	Private complaint access control
8	Conclude with **Flow Builder** view	Show how automation is configured

Goal:
To show a smooth workflow — *from complaint creation to resolution and reporting.*

3. Feedback Collection

Description:

Gather valuable feedback from evaluators, mentors, or users after the demo to improve the application.

Methods:

- **Google Form** for structured feedback
- **Direct Q&A session** after demo
- **Evaluator feedback sheet**

Feedback Points:

Category	Sample Question
Functionality	Does the system work as expected?
User Interface	Is the interface user-friendly?
Innovation	Is automation and flow implementation impressive?
Presentation	Was the explanation clear and confident?
Improvement	What additional features could be added?

📋 **Outcome:** Feedback will guide post-project improvements and feature extensions.

📄 4. Handoff Documentation

Description:

Proper documentation ensures the project can be understood, reused, or maintained by others after submission.

Deliverables:

Document Type	Description
Technical Document	Explains architecture, flow, triggers, and components
User Manual	Step-by-step usage guide for end users
Admin Guide	Configuration and deployment details
Test Case Document	All validation and automation test results
Presentation File (PPT)	Final pitch deck used for demo
Source Package	Unmanaged package link or metadata export

Location:

All final documents and assets are stored in a shared drive or Salesforce folder for evaluator access.

🏁 Conclusion

The **Customer Complaint Management System** built on the Salesforce platform provides an efficient and automated solution for managing customer issues and service requests.
Through the various phases of the project — from understanding the business problem to deployment and demo — we successfully implemented a real-world CRM application that improves **customer satisfaction**, **service efficiency**, and **organizational transparency**.

This project demonstrates the power of Salesforce as a **low-code/no-code CRM platform**, enabling automation through **Flows**, **Validation Rules**, **Reports**, and **Dashboards**, while also allowing advanced customization using **Apex** and **Lightning Web Components (LWC)**.

By integrating automated complaint assignment, acknowledgment emails, SLA tracking, and detailed analytics dashboards, the system ensures that every customer complaint is addressed promptly and efficiently.

The final deployment reflects a complete end-to-end CRM solution — from **data capture and processing** to **real-time reporting and feedback collection**.
This project not only highlights technical implementation but also emphasizes **team collaboration, problem-solving, and presentation skills** — essential for real-world Salesforce development.

In conclusion, the **Customer Complaint CRM Project** serves as a practical example of how cloud-based solutions can transform traditional customer support systems into **intelligent, automated, and customer-centric** platforms.