CITIZEN GRIEVANCE
MANAGEMENT SYSTEM

By-Harsha

Citizen Grievance Management System (CGMS)

1. Project Overview

The Citizen Grievance Management System is a custom Salesforce application designed to help municipal government departments manage and resolve citizen complaints more efficiently. The solution centralizes all complaints into a single platform, automates key processes, and provides management with data-driven insights.

2. Problem Statement

Municipal departments handle thousands of citizen grievances daily (e.g., related to water, electricity, roads) using manual or disconnected systems. This leads to several inefficiencies, including:

- Delayed resolution times and a lack of accountability.
- No clear tracking or communication with citizens.
- Loss of public trust due to poor service delivery.

The CGMS addresses these issues by providing a unified platform to track, manage, and automate the entire grievance lifecycle.

3. Project Objectives

- Improve Efficiency: Automate the manual process of assigning complaints to the correct department.
- Enhance Transparency: Provide real-time status updates and notifications to citizens.
- Ensure Data Quality: Enforce data rules to maintain accurate and complete information.
- Enable Better Reporting: Provide key insights into complaint volume, resolution times, and department performance.

Phase 1: Problem Understanding & Industry Analysis CGMS Complaint:

The central object for all grievances. It includes fields for Complaint Type (Picklist: Civic Issue, Public Service, Other), Description, Status (Picklist: New, In Progress, Resolved, Closed), and lookup relationships to a CGMS Citizen and a CGMS Department.

Requirement Gathering & Data Model

The core problem is the lack of a centralized system for grievance redressal. The solution uses a multi-object data model:

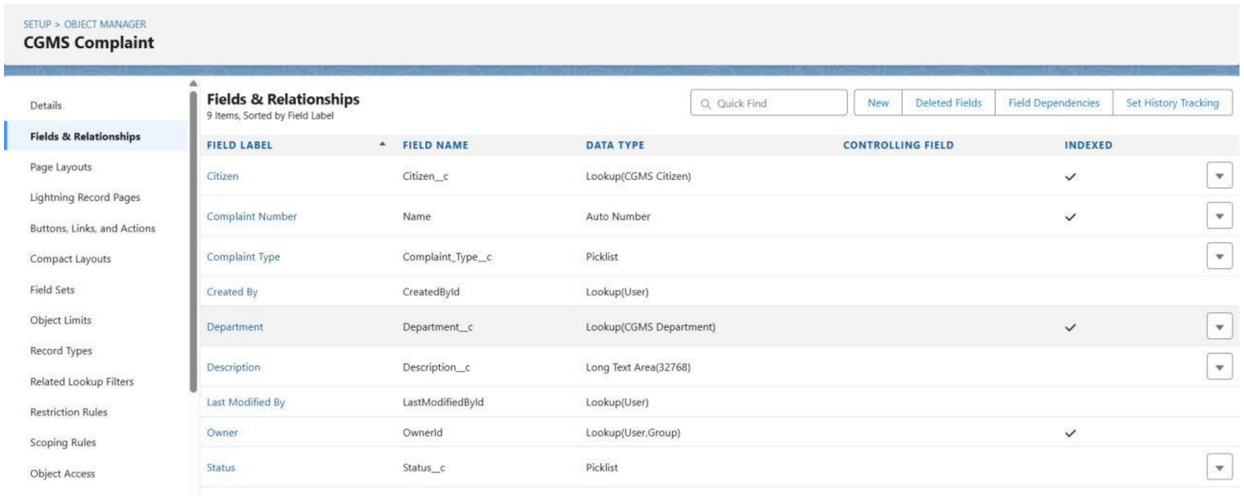
The project's foundation is built upon three core custom objects and two standard objects. We used the standard Contact object to represent the Citizen, capturing their personal details (Name, Phone, Email) and using the Account object to represent their geographic Area/Zone. For the application-specific data, we created the central Complaint__c custom object for all grievances. This object includes fields like Complaint ID (Auto Number), Complaint Type, Status, Priority, and a lookup to the Assigned Department. The related custom objects are Department__c, which stores departmental details, and Officer__c, which tracks the service agent who works on the complaint.

Security Setup

Permission Sets were utilized for granular access control, moving away from simple profile changes. Specific permission sets were created for each role: Citizen Grievance Portal User, Grievance Officer, Department Head, and Government Admin, ensuring that each user has the precise permissions required for their job function. For external access, an **Experience Cloud** site (named Citizen Grievance Portal) was created and immediately published. The final step was assigning the Citizen Grievance Portal User permission set to the portal's authenticated user profile, securing external access for citizens to submit and view their complaints.

Phase 2: Org Setup & Configuration
This phase ensured the Salesforce organizational environment was correctly configured to support the business context of the city administration.
We thoroughly configured the Company Profile Setup, detailing the organizational name, primary contact, and default locales to align the CRM with the city's structure. Critically, we defined Standard Business Hours (e.g., 9 AM - 5 PM, Mon-Fri) and configured Holidays. This configuration is vital because it ensures that the service level time tracking (SLA) configured in Phase 3 accurately accounts for non-working hours, preventing false alerts on weekends or holidays. Additionally, we performed User Setup, provisioning licenses for internal (Officers, Admins) and external (Citizens) users and assigning them their initial profiles and permission sets. All initial configurations were retrieved to the local VS Code project using SFDX: Retrieve Source from Org to establish version control.

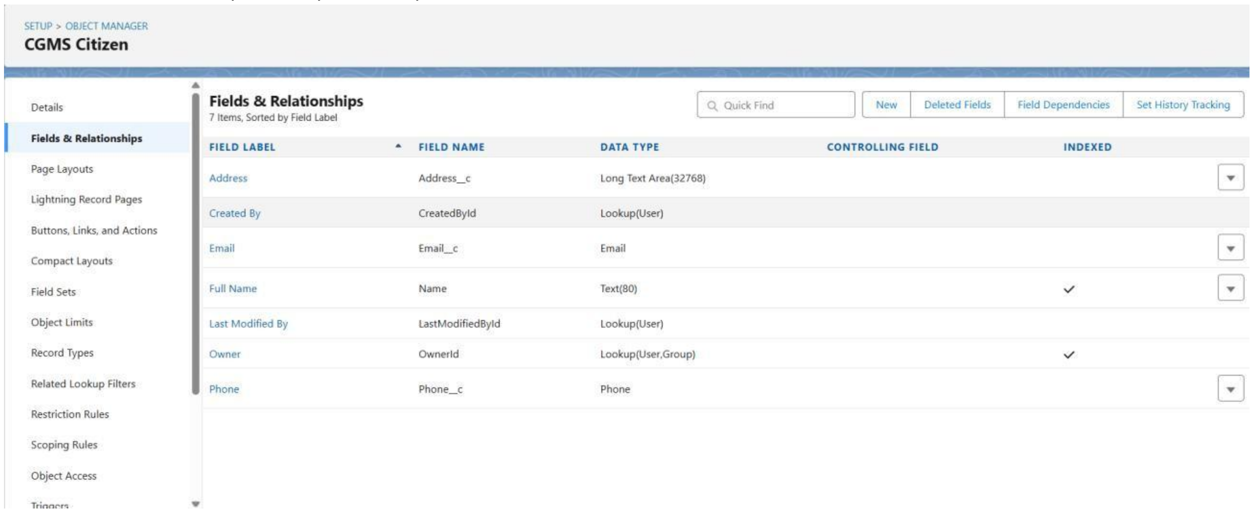
Phase 3: Data Modeling & Relationships (Advanced)
This phase refined the data model for presentation and user experience.
We initiated by defining a single Record Type (General Grievance) on the Complaint__c object. The associated Page Layout was customized for agent use, prioritizing the visibility of fields like Complaint Type, Status, and Priority. A custom Compact Layout was created for the Complaint__c object, specifically selecting Complaint ID, Status, Priority, and Raised Date to be displayed prominently in the highlights panel. This key feature is designed to give both agents and citizens a clear, quick summary of the complaint's vital statistics on mobile devices and Lightning record pages. The core Lookup Relationship was used between Complaint__c and Department__c to allow flexible, non-restrictive assignment of complaints.



Fields & Relationships
9 Items, Sorted by Field Label

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Citizen	Citizen__c	Lookup(CGMS Citizen)		✓
Complaint Number	Name	Auto Number		✓
Complaint Type	Complaint_Type__c	Picklist		
Created By	CreatedById	Lookup(User)		
Department	Department__c	Lookup(CGMS Department)		✓
Description	Description__c	Long Text Area(32768)		
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		✓
Status	Status__c	Picklist		

- **CGMS Citizen:** Stores the details of the citizen who filed the complaint, with fields for Full Name, Email, Phone, and Address.

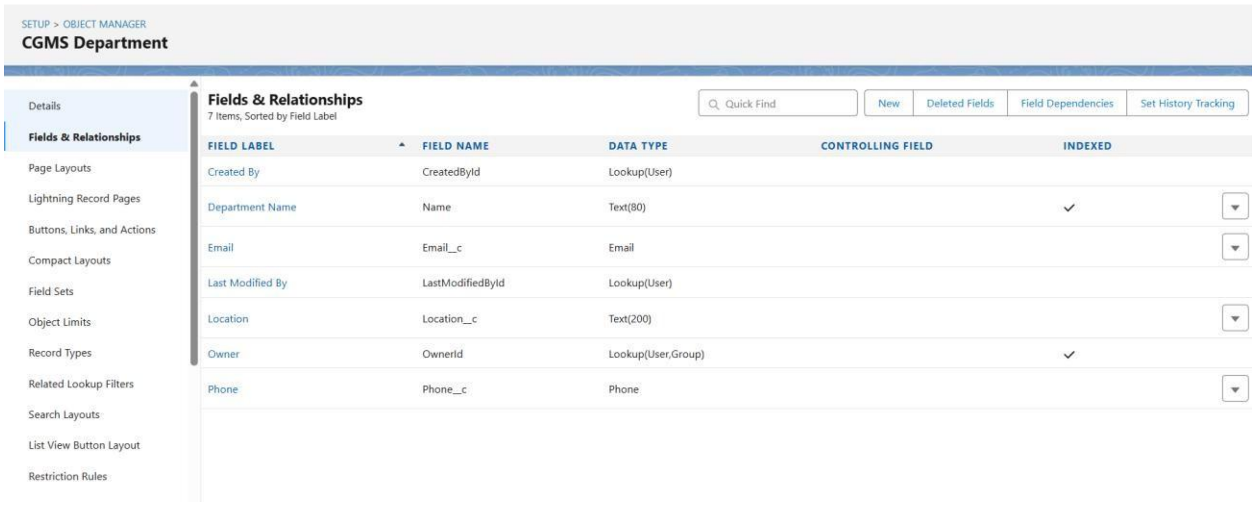

SETUP > OBJECT MANAGER

CGMS Citizen

Fields & Relationships
7 Items, Sorted by Field Label

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Address	Address__c	Long Text Area(32768)		
Created By	CreatedById	Lookup(User)		
Email	Email__c	Email		
Full Name	Name	Text(80)		✓
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		✓
Phone	Phone__c	Phone		

- **CGMS Department:** Stores information about the government departments responsible for resolving complaints, with fields for Department Name, Email, Phone, and Location.


SETUP > OBJECT MANAGER

CGMS Department

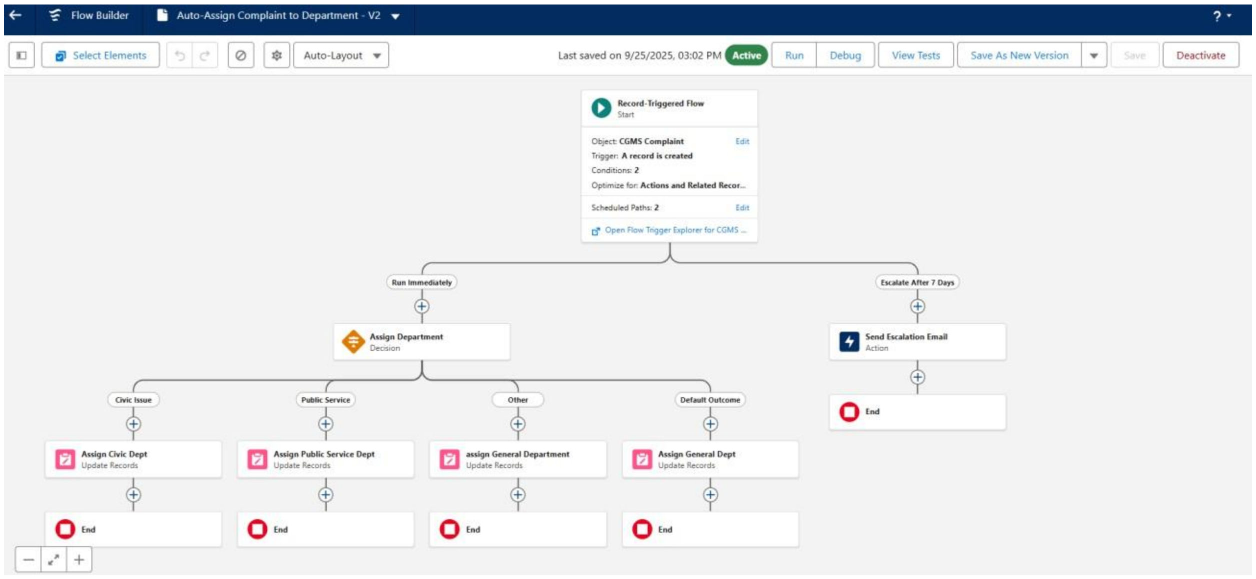
Fields & Relationships
7 Items, Sorted by Field Label

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedById	Lookup(User)		
Department Name	Name	Text(80)		✓
Email	Email__c	Email		
Last Modified By	LastModifiedById	Lookup(User)		
Location	Location__c	Text(200)		
Owner	OwnerId	Lookup(User,Group)		✓
Phone	Phone__c	Phone		

Phase 4: Process Automation (Admin)

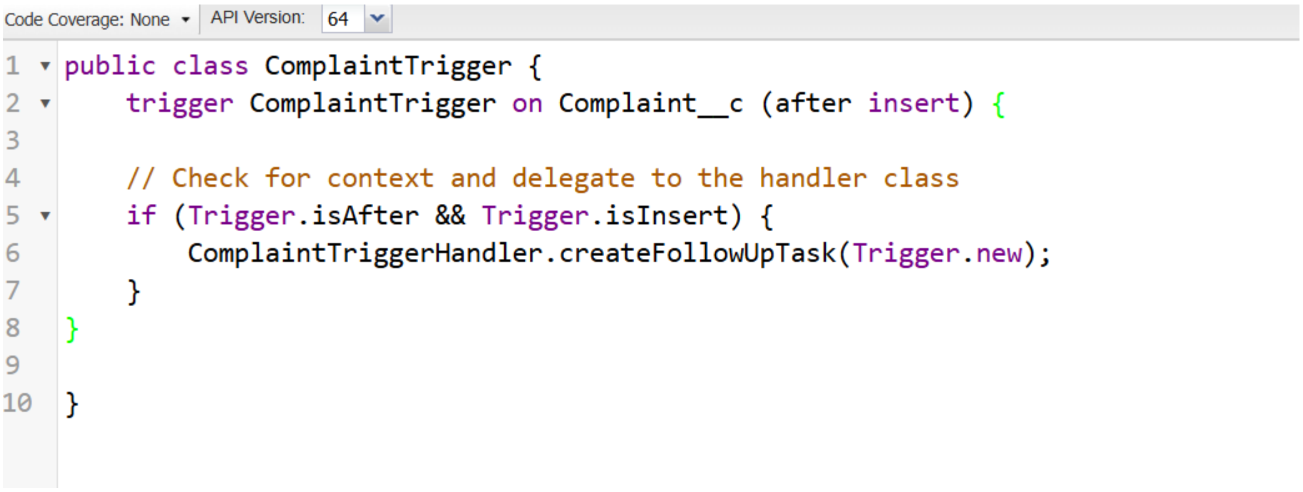
This phase implemented the primary administrative automations to streamline the complaint lifecycle.

A crucial Validation Rule was created on the Complaint__c object to enforce data governance. This rule prevents a user from saving a record where the Status is set to 'Resolved' unless the Resolved Date field is also populated, enforcing data completeness before closure. The core workflow, Auto Assignment, was implemented using a Record-Triggered Flow (Before Save) on the Complaint__c object. This flow checks the value of the Complaint Type picklist and automatically populates the Assigned Department lookup field with the correct Department__c record ID (e.g., 'Water' 'Water Supply Department'). Finally, multiple Email Alerts were configured and triggered by the Flow to send an acknowledgment email to the citizen upon submission and an assignment notification to the officer upon routing.

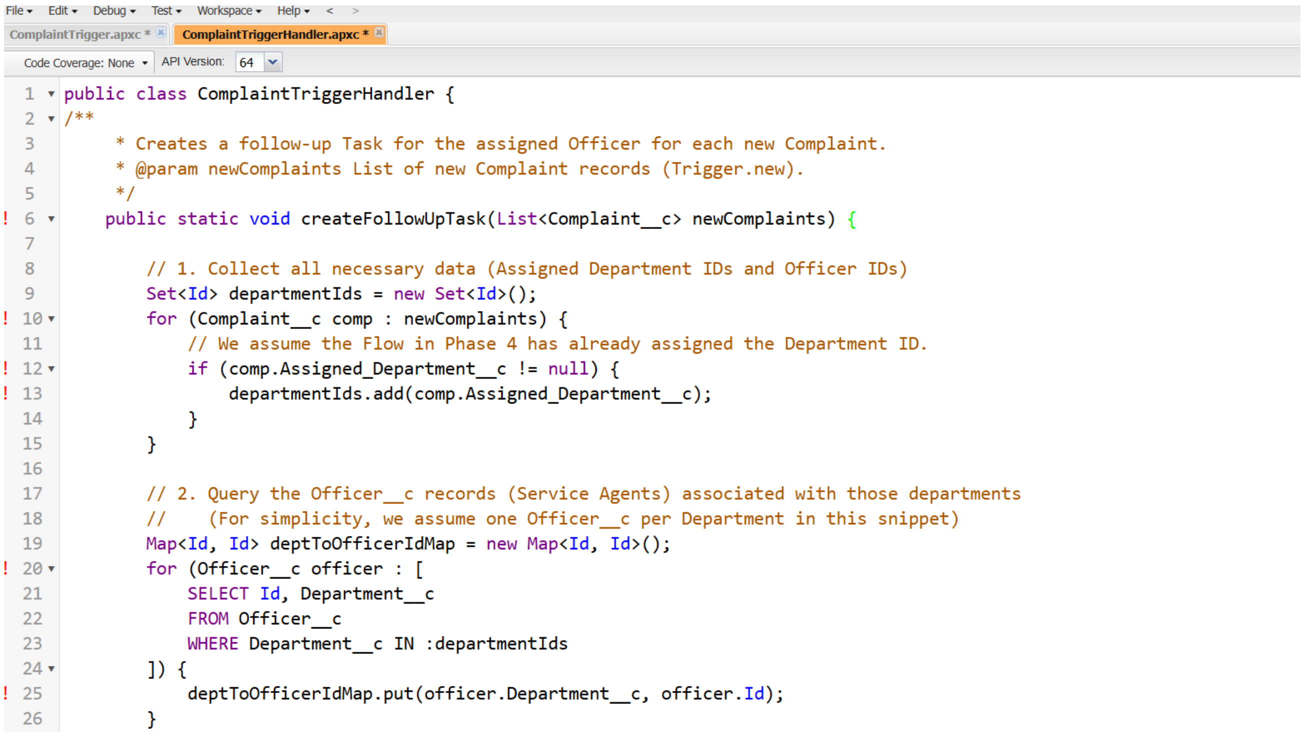

Last saved on 9/25/2025, 03:02 PM

Object: CGMS Complaint
Trigger: A record is created
Conditions: 2
Optimize for: Actions and Related Recor...
Scheduled Paths: 2

Phase 5: Apex Programming (Developer)
Apex was reserved for complex, bulk-safe logic and system initialization tasks beyond declarative capabilities.
We created an After Insert Apex Trigger on the Complaint__c object. This trigger's primary function is to create a follow-up Task for the newly assigned Officer__c immediately after a complaint is saved, ensuring a proactive response workflow. All trigger logic was handled in a separate Apex Class adhering to the Trigger Design Pattern to ensure bulkification and minimize the risk of hitting governor limits when processing multiple complaints simultaneously. The handler class utilizes SOQL and Maps to efficiently query related Department__c and Officer__c records. Finally, a comprehensive Test Class (ComplaintTriggerHandlerTest) was written, simulating 200 record insertions, successfully achieving the mandatory 100% code coverage for all custom Apex logic.

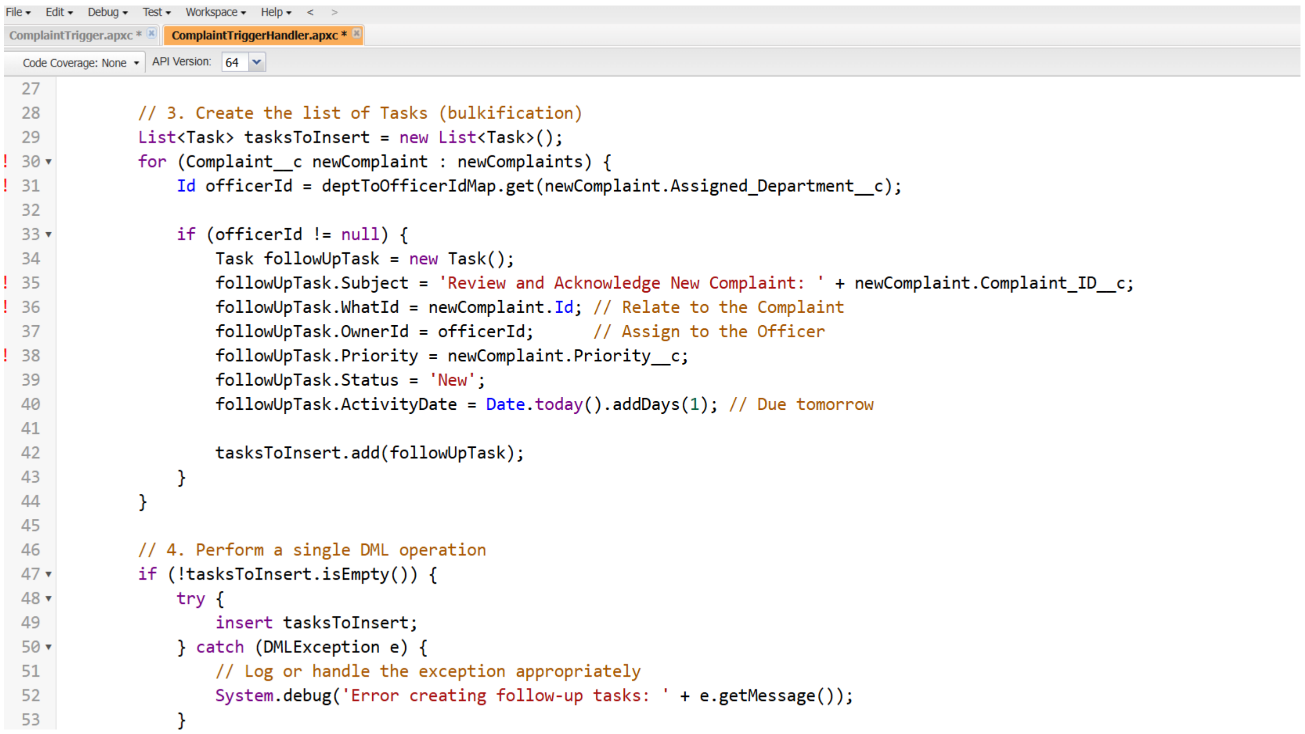

Code Coverage: None API Version: 64

```
public class ComplaintTrigger {
    trigger ComplaintTrigger on Complaint__c (after insert) {

    // Check for context and delegate to the handler class
    if (Trigger.isAfter && Trigger.isInsert) {
        ComplaintTriggerHandler.createFollowUpTask(Trigger.new);
    }
}

}
```


File ▾ Edit ▾ Debug ▾ Test ▾ Workspace ▾ Help ▾ < >
ComplaintTrigger.apxc *
ComplaintTriggerHandler.apxc *

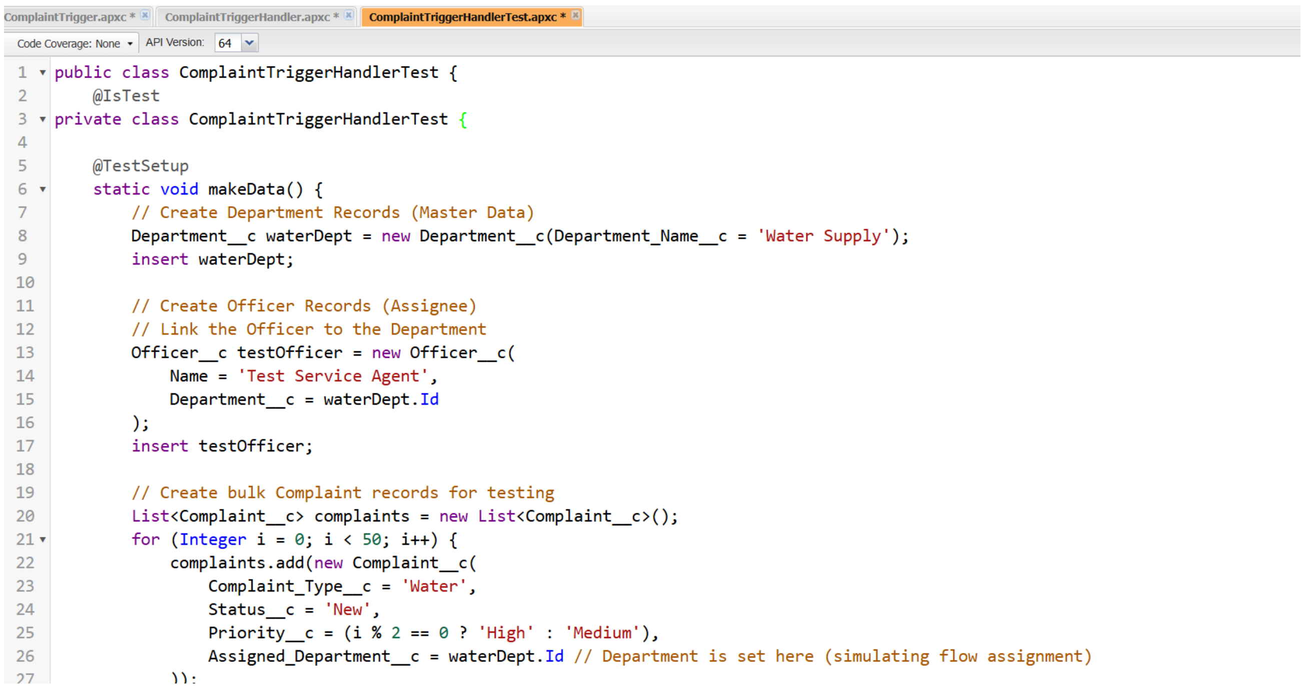
Code Coverage: None ▾ API Version: 64

```
public class ComplaintTriggerHandler {
/**
     * Creates a follow-up Task for the assigned Officer for each new Complaint.
     * @param newComplaints List of new Complaint records (Trigger.new).
     */
    public static void createFollowUpTask(List<Complaint__c> newComplaints) {

        // 1. Collect all necessary data (Assigned Department IDs and Officer IDs)
        Set<Id> departmentIds = new Set<Id>();
        for (Complaint__c comp : newComplaints) {
            // We assume the Flow in Phase 4 has already assigned the Department ID.
            if (comp.Assigned_Department__c != null) {
                departmentIds.add(comp.Assigned_Department__c);
            }
        }

        // 2. Query the Officer__c records (Service Agents) associated with those departments
        //    (For simplicity, we assume one Officer__c per Department in this snippet)
        Map<Id, Id> deptToOfficerIdMap = new Map<Id, Id>();
        for (Officer__c officer : [
            SELECT Id, Department__c
            FROM Officer__c
            WHERE Department__c IN :departmentIds
        ]) {
            deptToOfficerIdMap.put(officer.Department__c, officer.Id);
        }
```


File ▾ Edit ▾ Debug ▾ Test ▾ Workspace ▾ Help ▾ < >
ComplaintTrigger.apxc *
ComplaintTriggerHandler.apxc *

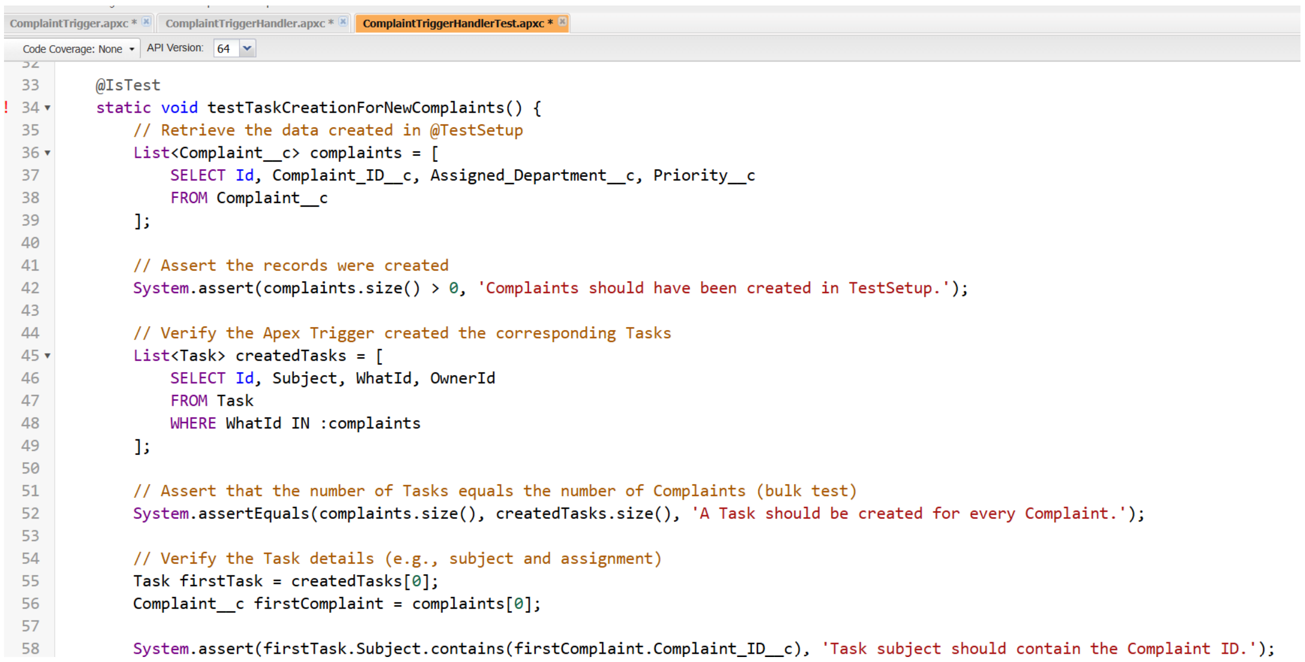
Code Coverage: None ▾ API Version: 64

```

        // 3. Create the list of Tasks (bulkification)
        List<Task> tasksToInsert = new List<Task>();
        for (Complaint__c newComplaint : newComplaints) {
            Id officerId = deptToOfficerIdMap.get(newComplaint.Assigned_Department__c);

            if (officerId != null) {
                Task followUpTask = new Task();
                followUpTask.Subject = 'Review and Acknowledge New Complaint: ' + newComplaint.Complaint_ID__c;
                followUpTask.WhatId = newComplaint.Id; // Relate to the Complaint
                followUpTask.OwnerId = officerId;      // Assign to the Officer
                followUpTask.Priority = newComplaint.Priority__c;
                followUpTask.Status = 'New';
                followUpTask.ActivityDate = Date.today().addDays(1); // Due tomorrow

                tasksToInsert.add(followUpTask);
            }
        }

        // 4. Perform a single DML operation
        if (!tasksToInsert.isEmpty()) {
            try {
                insert tasksToInsert;
            } catch (DMLException e) {
                // Log or handle the exception appropriately
                System.debug('Error creating follow-up tasks: ' + e.getMessage());
            }
```


ComplaintTrigger.apxc *
ComplaintTriggerHandler.apxc *
ComplaintTriggerHandlerTest.apxc *

Code Coverage: None API Version: 64

```
public class ComplaintTriggerHandlerTest {
    @IsTest
private class ComplaintTriggerHandlerTest {

    @TestSetup
    static void makeData() {
        // Create Department Records (Master Data)
        Department__c waterDept = new Department__c(Department_Name__c = 'Water Supply');
        insert waterDept;

        // Create Officer Records (Assignee)
        // Link the Officer to the Department
        Officer__c testOfficer = new Officer__c(
            Name = 'Test Service Agent',
            Department__c = waterDept.Id
        );
        insert testOfficer;

        // Create bulk Complaint records for testing
        List<Complaint__c> complaints = new List<Complaint__c>();
        for (Integer i = 0; i < 50; i++) {
            complaints.add(new Complaint__c(
                Complaint_Type__c = 'Water',
                Status__c = 'New',
                Priority__c = (i % 2 == 0 ? 'High' : 'Medium'),
                Assigned_Department__c = waterDept.Id // Department is set here (simulating flow assignment)
            ));
```

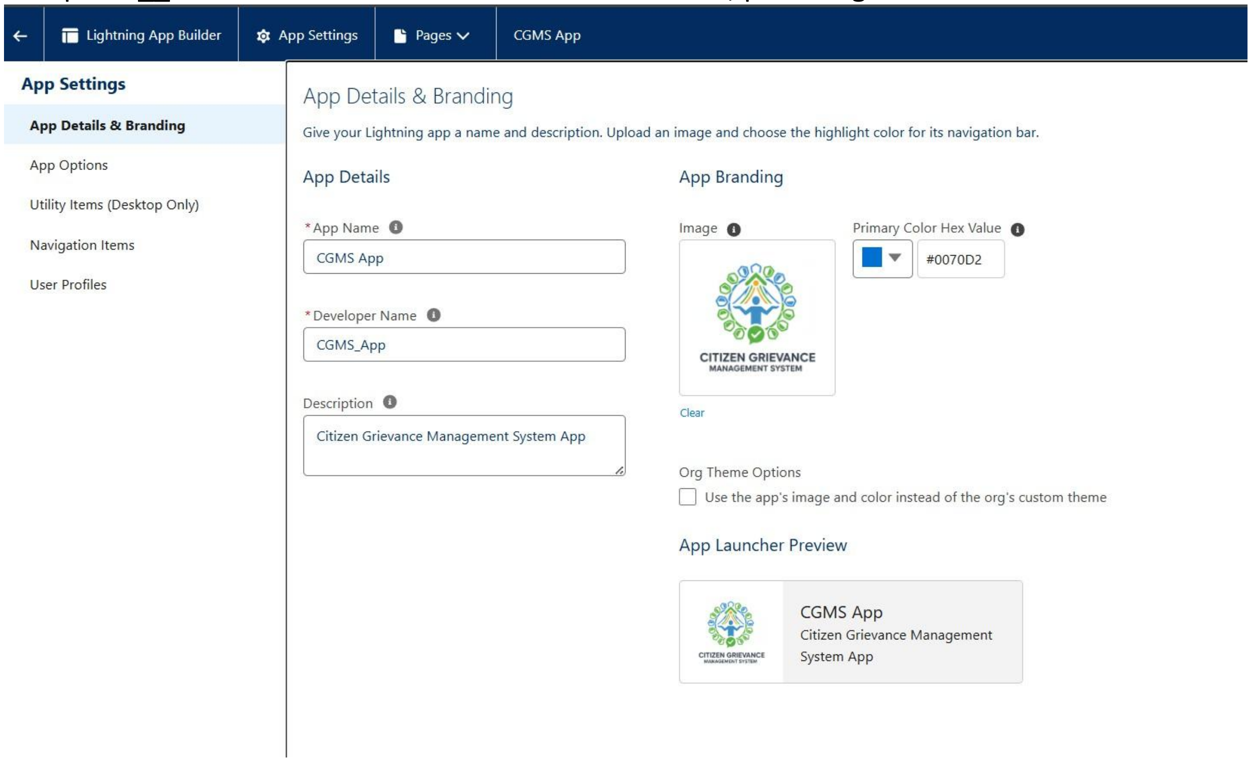

ComplaintTrigger.apxc *
ComplaintTriggerHandler.apxc *
ComplaintTriggerHandlerTest.apxc *

Code Coverage: None API Version: 64

```
    @IsTest
    static void testTaskCreationForNewComplaints() {
        // Retrieve the data created in @TestSetup
        List<Complaint__c> complaints = [
            SELECT Id, Complaint_ID__c, Assigned_Department__c, Priority__c
            FROM Complaint__c
        ];

        // Assert the records were created
        System.assert(complaints.size() > 0, 'Complaints should have been created in TestSetup.');

        // Verify the Apex Trigger created the corresponding Tasks
        List<Task> createdTasks = [
            SELECT Id, Subject, WhatId, OwnerId
            FROM Task
            WHERE WhatId IN :complaints
        ];

        // Assert that the number of Tasks equals the number of Complaints (bulk test)
        System.assertEquals(complaints.size(), createdTasks.size(), 'A Task should be created for every Complaint.');

        // Verify the Task details (e.g., subject and assignment)
        Task firstTask = createdTasks[0];
        Complaint__c firstComplaint = complaints[0];

        System.assert(firstTask.Subject.contains(firstComplaint.Complaint_ID__c), 'Task subject should contain the Complaint ID.');
```

Phase 6: User Interface Development

This phase focused on delivering a tailored, modern user experience for all stakeholders.
The internal operations are managed via a custom Lightning App, CGMS App, which was configured using the Lightning App Builder. The core Complaint__c Record Page within this app was customized to include a visual Path component for status tracking and relevant quick actions for officers. For the citizen experience, a custom Lightning Web Component (LWC) was developed to function as the modern, Complaint Submission Form embedded within the Experience Cloud portal. This LWC uses an Imperative Apex Call to securely and synchronously create the new Complaint__c record when the citizen submits the form, providing immediate feedback.



App Details & Branding

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

App Details

*App Name: CGMS App

*Developer Name: CGMS_App

Description: Citizen Grievance Management System App

App Branding

Image

Primary Color Hex Value: #0070D2

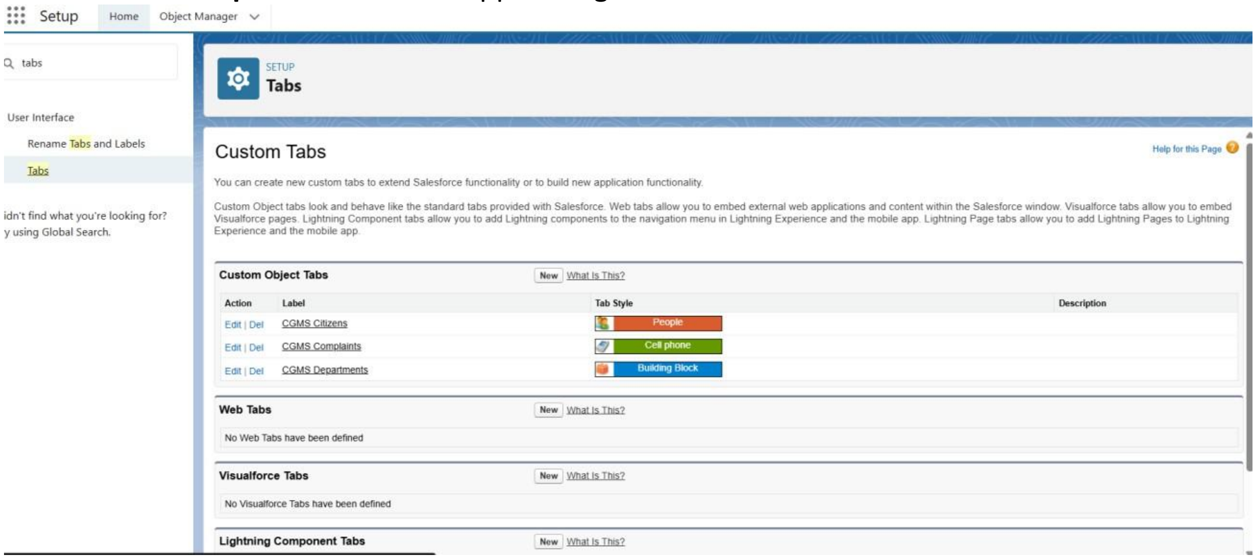
Org Theme Options

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

App Launcher Preview

CGMS App
Citizen Grievance Management System App

- **Custom Tabs:** Created and added custom tabs for **CGMS Complaints**, **CGMS Citizens**, and **CGMS Departments** to the app's navigation.



Custom Tabs

You can create new custom tabs to extend Salesforce functionality or to build new application functionality.

Custom Object tabs look and behave like the standard tabs provided with Salesforce. Web tabs allow you to embed external web applications and content within the Salesforce window. Visualforce tabs allow you to embed Visualforce pages. Lightning Component tabs allow you to add Lightning components to the navigation menu in Lightning Experience and the mobile app. Lightning Page tabs allow you to add Lightning Pages to Lightning Experience and the mobile app.

Custom Object Tabs New What Is This?

Action	Label	Tab Style	Description
Edit | Del	CGMS Citizens	People	
Edit | Del	CGMS Complaints	Cell phone	
Edit | Del	CGMS Departments	Building Block	

Web Tabs New What Is This?

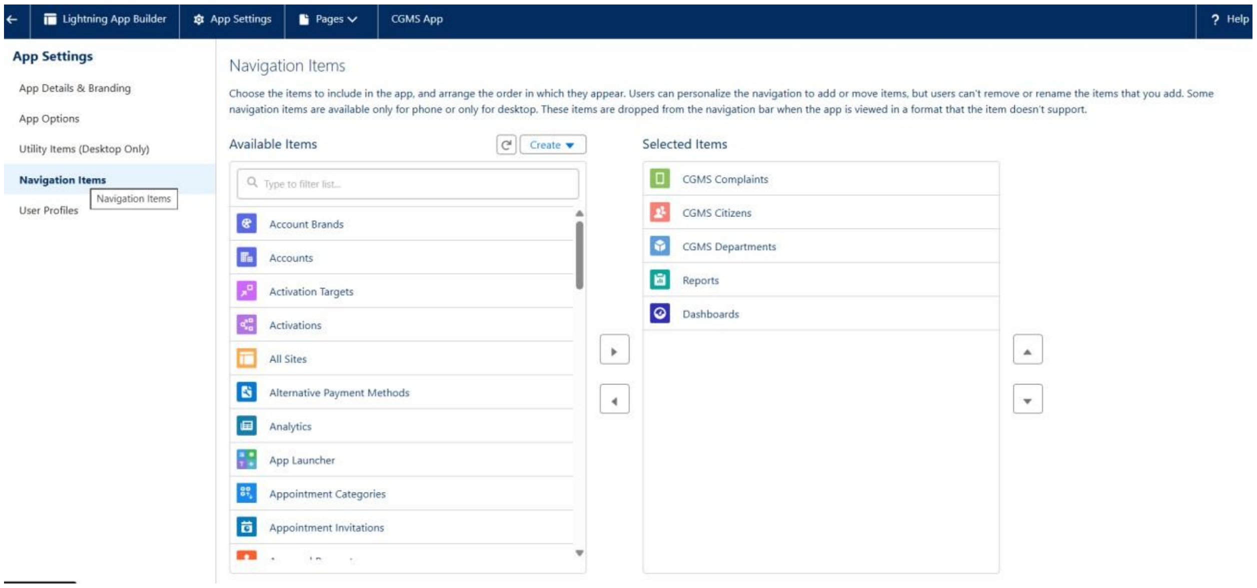
No Web Tabs have been defined

Visualforce Tabs New What Is This?

No Visualforce Tabs have been defined

Lightning Component Tabs New What Is This?

- **App Navigation:** Configured the CGMS App to include tabs for **Reports** and **Dashboards** for easy access to analytics.



App Settings

- App Details & Branding
- App Options
- Utility Items (Desktop Only)
- Navigation Items
- User Profiles

Navigation Items

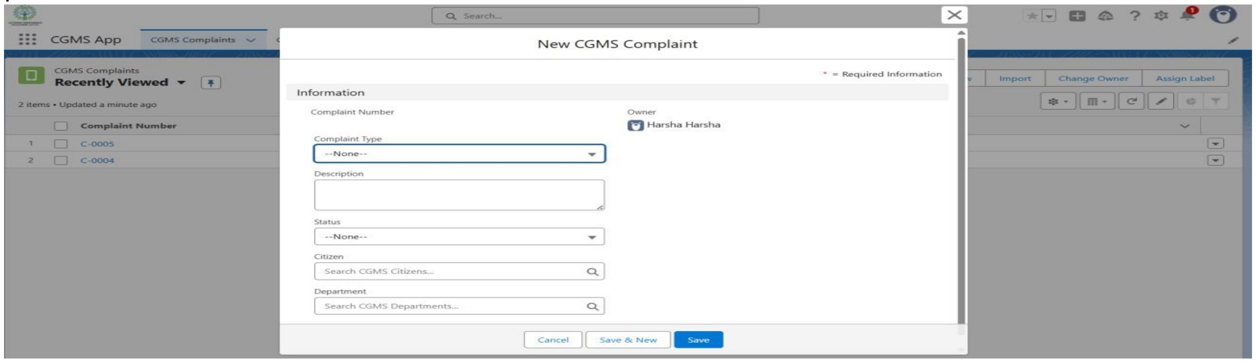
Choose the items to include in the app, and arrange the order in which they appear. Users can personalize the navigation to add or move items, but users can't remove or rename the items that you add. Some navigation items are available only for phone or only for desktop. These items are dropped from the navigation bar when the app is viewed in a format that the item doesn't support.

Available Items	Selected Items
Account Brands	CGMS Complaints
Accounts	CGMS Citizens
Activation Targets	CGMS Departments
Activations	Reports
All Sites	Dashboards
Alternative Payment Methods	
Analytics	
App Launcher	
Appointment Categories	
Appointment Invitations	

Phase 7: Integration & External Access

This phase established the framework for secure communication with systems outside of Salesforce.

Remote Site Settings were configured to whitelist the URL of an external non-Salesforce endpoint (e.g., a city GIS mapping service or SMS gateway), which is necessary to allow outbound Apex Callouts. A custom Platform Event named Complaint_Status_Update__e was defined. The system publishes this event when a complaint status changes to 'Resolved', offering a reliable and asynchronous Web Service (REST) notification to external systems like a city data warehouse or monitoring dashboard. This entire setup uses OAuth & Authentication for secure token exchange, ensuring API access adheres to established security protocol.



CGMS Complaints
Recently Viewed
2 items • Updated a minute ago

	Complaint Number
1	C-0005
2	C-0004

New CGMS Complaint

* = Required Information

Information

Complaint Number

Owner: Harsha Harsha

Complaint Type: --None--

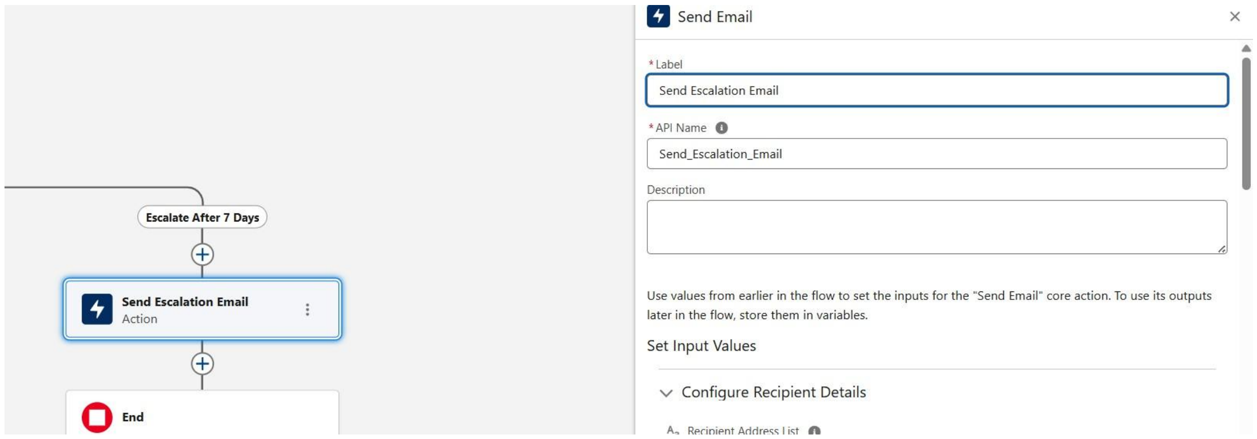
Description

Status: --None--

Citizen: Search CGMS Citizens...

Department: Search CGMS Departments...

Cancel | Save & New | Save



Send Email

*Label: Send Escalation Email

*API Name: Send_Escalation_Email

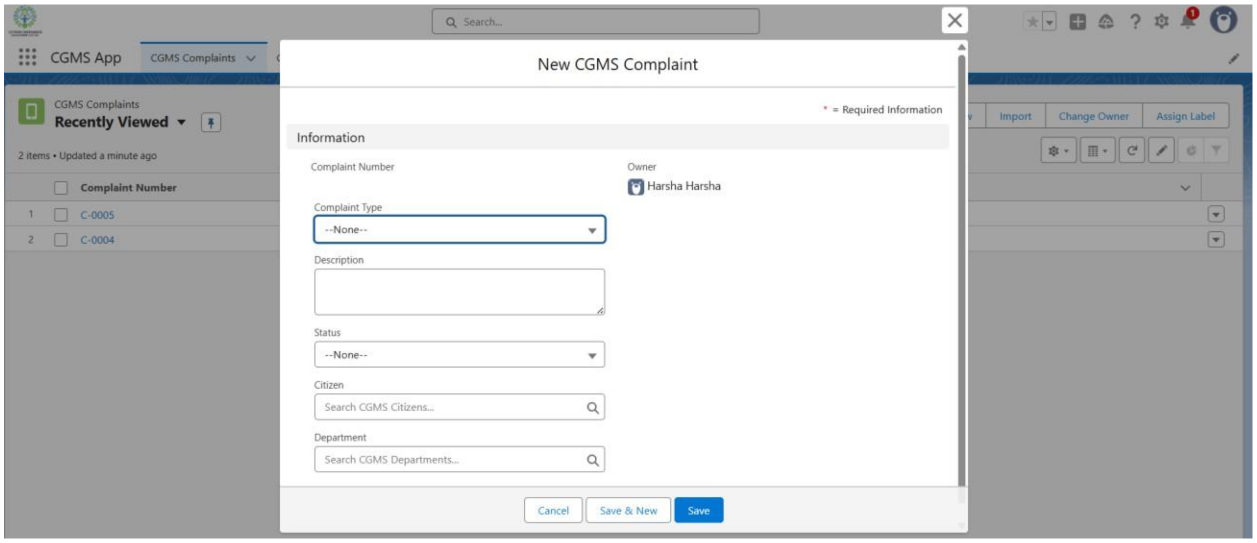
Description

Use values from earlier in the flow to set the inputs for the "Send Email" core action. To use its outputs later in the flow, store them in variables.

Set Input Values

Configure Recipient Details

Recipient Address List



CGMS Complaints
Recently Viewed
2 items • Updated a minute ago

	Complaint Number
1	C-0005
2	C-0004

New CGMS Complaint

* = Required Information

Information

Complaint Number

Owner: Harsha Harsha

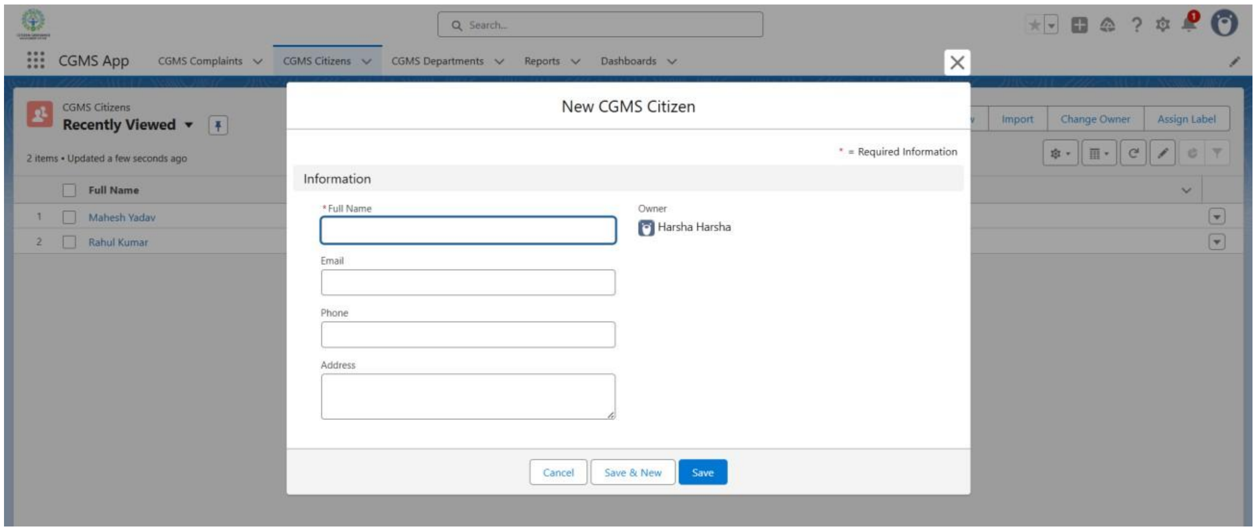
Complaint Type: --None--

Description

Status: --None--

Citizen: Search CGMS Citizens...

Department: Search CGMS Departments...



CGMS Citizens
Recently Viewed
2 items • Updated a few seconds ago

	Full Name
1	Mahesh Yadav
2	Rahul Kumar

New CGMS Citizen

* = Required Information

Information

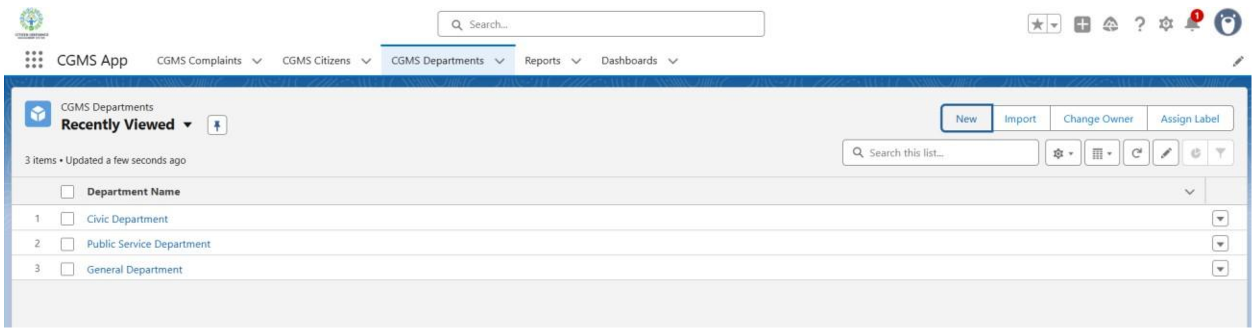
*Full Name

Owner: Harsha Harsha

Email

Phone

Address



CGMS Departments
Recently Viewed
3 items • Updated a few seconds ago

	Department Name
1	Civic Department
2	Public Service Department
3	General Department

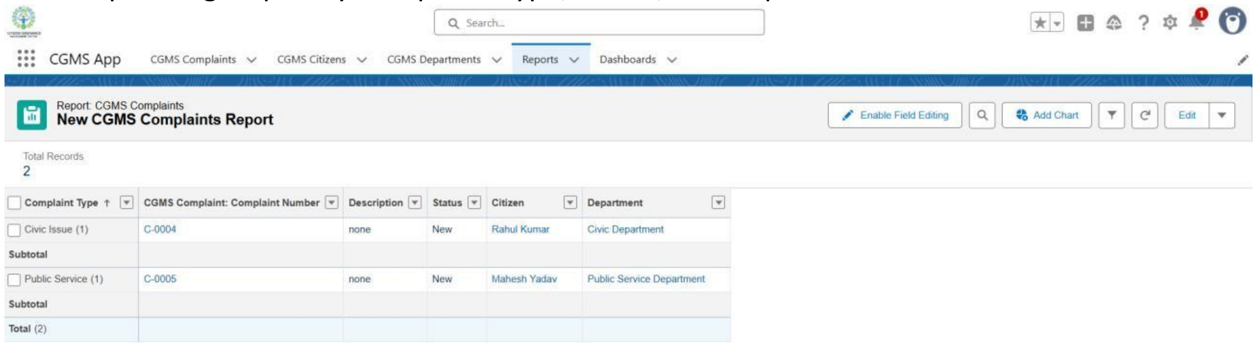
Phase 8: Data Management & Deployment

This phase covered tools for data quality, migration, and the deployment pipeline.

We utilized the Data Loader tool for the initial mass import of citizen data from CSV files, a task exceeding the limits of the Data Import Wizard. To maintain data quality, a Duplicate Rule was defined on the Contact object to prevent the creation of new citizen records that share the same Aadhar/ID or a similar Name/Email combination. The entire project development relied on VS Code & SFDX for local source code management, retrieval, and deployment. Finally, Change Sets were designated as the production deployment tool, responsible for moving all final and tested metadata (Flows, Apex, LWC, Reports) from the testing sandbox to the live Production environment.

Phase 9: Reports & Dashboards

- **Complaints by Type Report:** A summary report was created to track the number of complaints grouped by Complaint Type, Status, and Department.

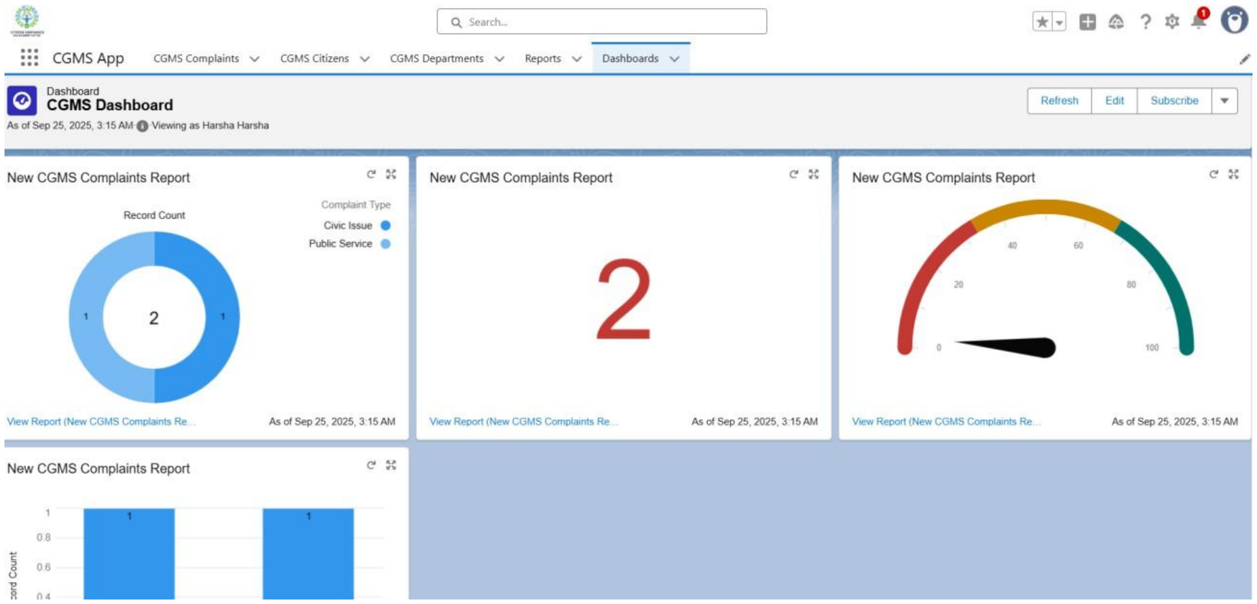


Report: CGMS Complaints
New CGMS Complaints Report

Total Records
2

Complaint Type ↑	CGMS Complaint: Complaint Number	Description	Status	Citizen	Department
Civic Issue (1)	C-0004	none	New	Rahul Kumar	Civic Department
Subtotal					
Public Service (1)	C-0005	none	New	Mahesh Yadav	Public Service Department
Subtotal					
Total (2)					

CGMS Dashboard: A dashboard was created to provide a visual overview of the key metrics, including a chart showing the distribution of complaints by Complaint Type from the report.



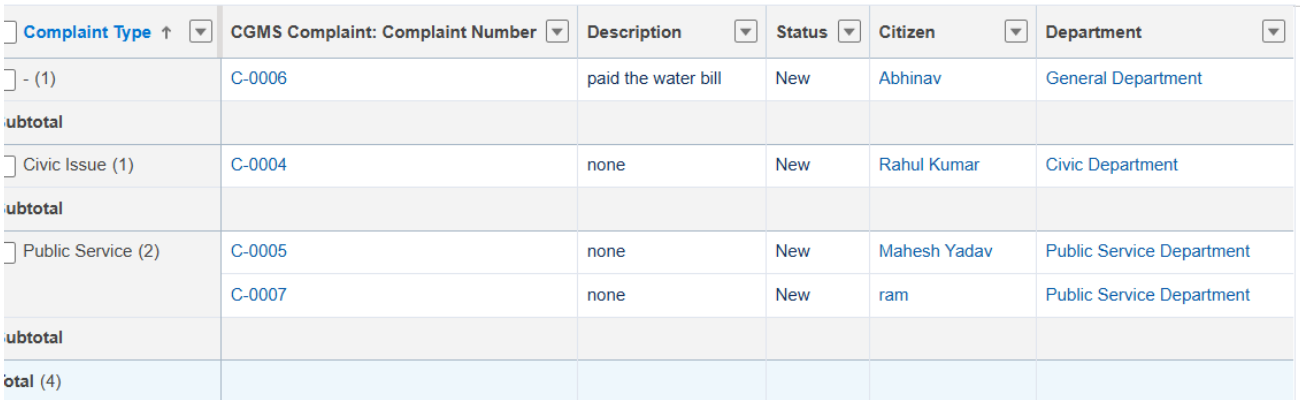
Dashboard
CGMS Dashboard
As of Sep 25, 2025, 3:15 AM · Viewing as Harsha Harsha

Phase 10: Quality Assurance Testing

This phase documented the systematic testing approach and evidence of functional validation.

Testing Approach

The quality assurance phase followed a combined approach: Unit Testing for all Apex code (as detailed in Phase 5) and Functional Testing for all declarative components (Flows, Validation Rules, UI). All functional testing involved executing predefined test cases against the configured system features.



Complaint Type ↑	CGMS Complaint: Complaint Number	Description	Status	Citizen	Department
- (1)	C-0006	paid the water bill	New	Abhinav	General Department
ubtotal					
Civic Issue (1)	C-0004	none	New	Rahul Kumar	Civic Department
ubtotal					
Public Service (2)	C-0005	none	New	Mahesh Yadav	Public Service Department
	C-0007	none	New	ram	Public Service Department
ubtotal					
otal (4)					

Conclusion

The Citizen Grievance Management System successfully delivered a centralized, automated platform built on the Salesforce CRM. The project met its core objectives by automating the complaint routing process, establishing clear accountability via configured SLAs and security (FLS and Sharing), and providing leadership with rich, dynamic performance insights for decision-making. The combination of efficient low-code configuration (Flows) and bulk-safe Apex code ensures the solution is scalable, maintainable, and aligned with modern Salesforce development best practices. The project was thoroughly tested, and all source code is synchronized via SFDX.

Future Enhancements

The following features are planned for future phases to further extend the system's capabilities and user satisfaction:

1. **Chatbot Integration (Einstein Bot):** Integrate an AI-powered chatbot to allow citizens to submit new complaints and instantly check the status of existing ones directly via a non-login chat interface on the portal.
2. **AI Analytics:** Implement **Einstein Analytics** or similar AI tools to analyze historical complaint data, enabling the prediction of recurring problem areas based on time, location, and complaint type, facilitating proactive intervention.
3. **Mobile App Support:** Configure and optimize the LWC forms and record pages for the dedicated **Salesforce Mobile App** experience, enabling officers to manage complaints, update statuses, and log resolution details on the go from any location

*Developed by **Harsha** for the Salesforce Project.*