# ****Phase 1: Problem Understanding & Industry Analysis****

## ****Problem Statement: Internal Helpdesk CRM****

**Prepared by:** **MACHAPATHRI VAMSHI**

### ****Requirement Gathering****

**Core Functionality:**  
Employees should be able to raise IT support tickets for issues such as password resets, software installation, or hardware problems. Tickets should automatically get assigned to the right IT support staff based on category, and the status should be tracked until resolution.

**Reporting & KPIs:**  
The system should measure average resolution time, ticket volume per category per month, and percentage of tickets resolved within SLA timelines.

**Automation:**  
Notifications should be sent to employees when tickets are created, updated, or resolved. Escalations should occur automatically if SLA timelines are breached.

### ****Stakeholder Analysis****

* **Employees:** Need a simple way to raise tickets and get updates quickly.
* **IT Support Team:** Need an organized queue of tickets to work on and update status easily.
* **IT Manager:** Needs dashboards and reports to monitor SLA performance and team workload.
* **Management:** Wants visibility into major recurring issues to take preventive actions.

### ****Business Process Mapping****

**Step 1:** Employee logs in and submits a ticket.  
**Step 2:** System auto-assigns ticket to appropriate support staff based on category.  
**Step 3:** Support staff works on the issue and updates ticket status.  
**Step 4:** Ticket is marked closed once resolved, and notification is sent to employee.  
**Step 5:** Resolution time and ticket data are captured for reports.

**Simple Process Flow Diagram (describe or draw):**

[Submit Ticket] → [Auto Assign] → [Work on Ticket] → [Close Ticket] → [Report Metrics]

### ****Industry-Specific Use Case Analysis****

An internal helpdesk CRM is critical for any corporate environment to ensure smooth IT operations. It helps reduce downtime, ensures SLA compliance, and improves employee satisfaction. For cybersecurity, it tracks security-related tickets like MFA issues or phishing reports, enabling faster incident response.

### ****AppExchange Exploration****

Explore ITSM and helpdesk apps on AppExchange such as **Service Cloud**, **Spoke Desk**, or **Milestones PM** to understand best practices in case management, escalation rules, and SLA tracking. These apps can provide inspiration for dashboards, reports, and automation flows.

# **Phase 2: Org Setup & Configuration**

# 1. Project Goal

Prepare Salesforce org with users, roles, profiles, permission sets, sharing settings, and email templates in preparation for building the Helpdesk Ticket object.

### Step 1: Log in to Salesforce Org

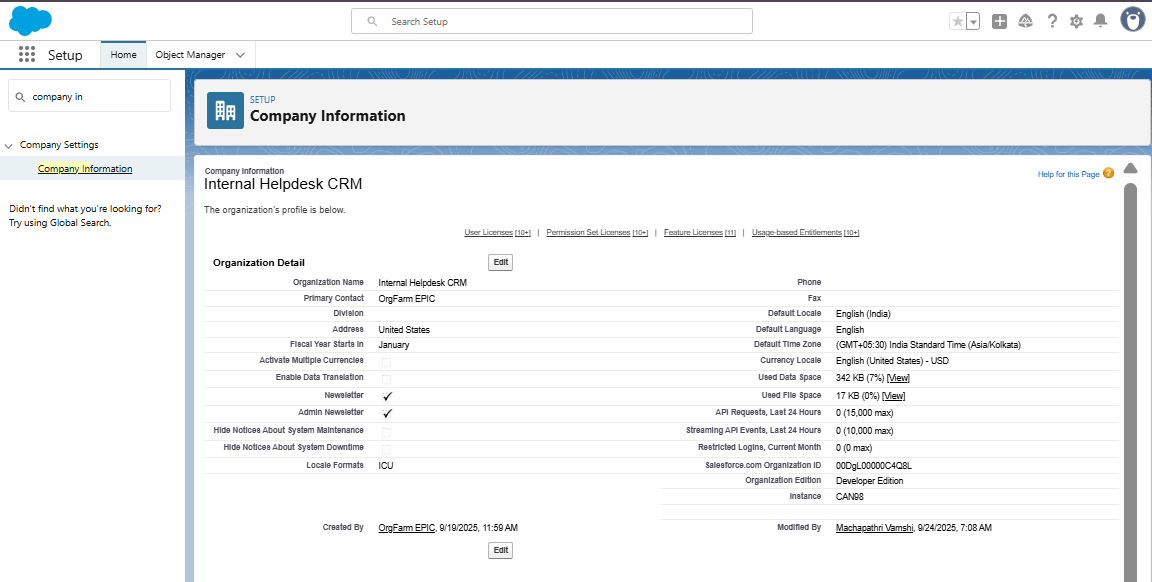
* Go to [https://login.salesforce.com](https://login.salesforce.com" \t "_new)
* Enter username and password
* Click **Log In**

## Step 2: Set Company Information

1. Setup → Quick Find → **Company Information** → Click
2. Verify/update:

* Company Name
* Primary Contact
* Default Locale: English (India)
* Default Time Zone: Asia/Kolkata

1. Click **Edit** → update → **Save**



## Step 3: Define Business Hours

1. Setup → Quick Find → **Business Hours** → Click
2. Click **New Business Hours**:

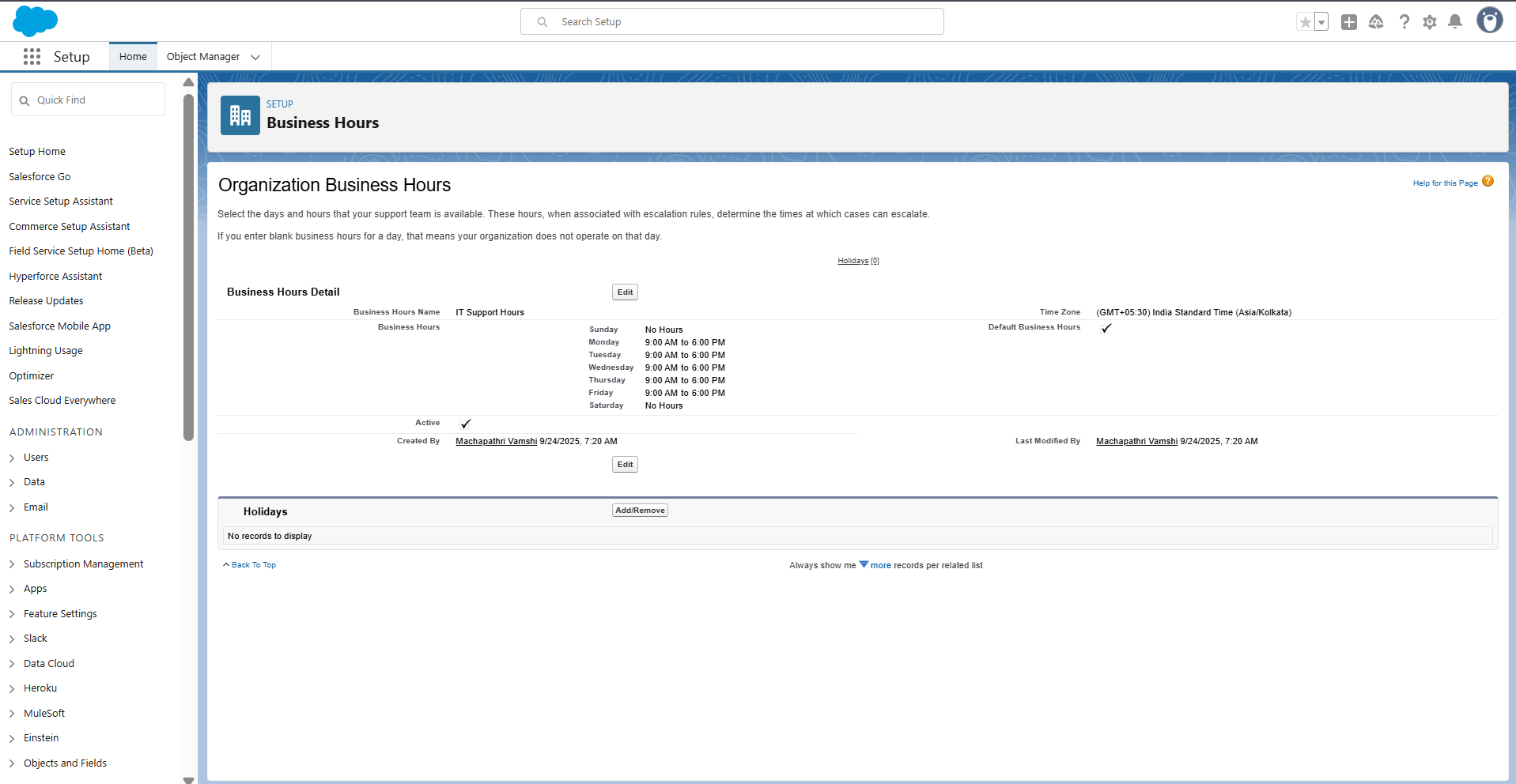
Name: IT Support Hours

Default: yes

Time Zone: Asia/Kolkata

Open: 09:00 → Close: 06:00 (Mon-Fri)

1. Click Save

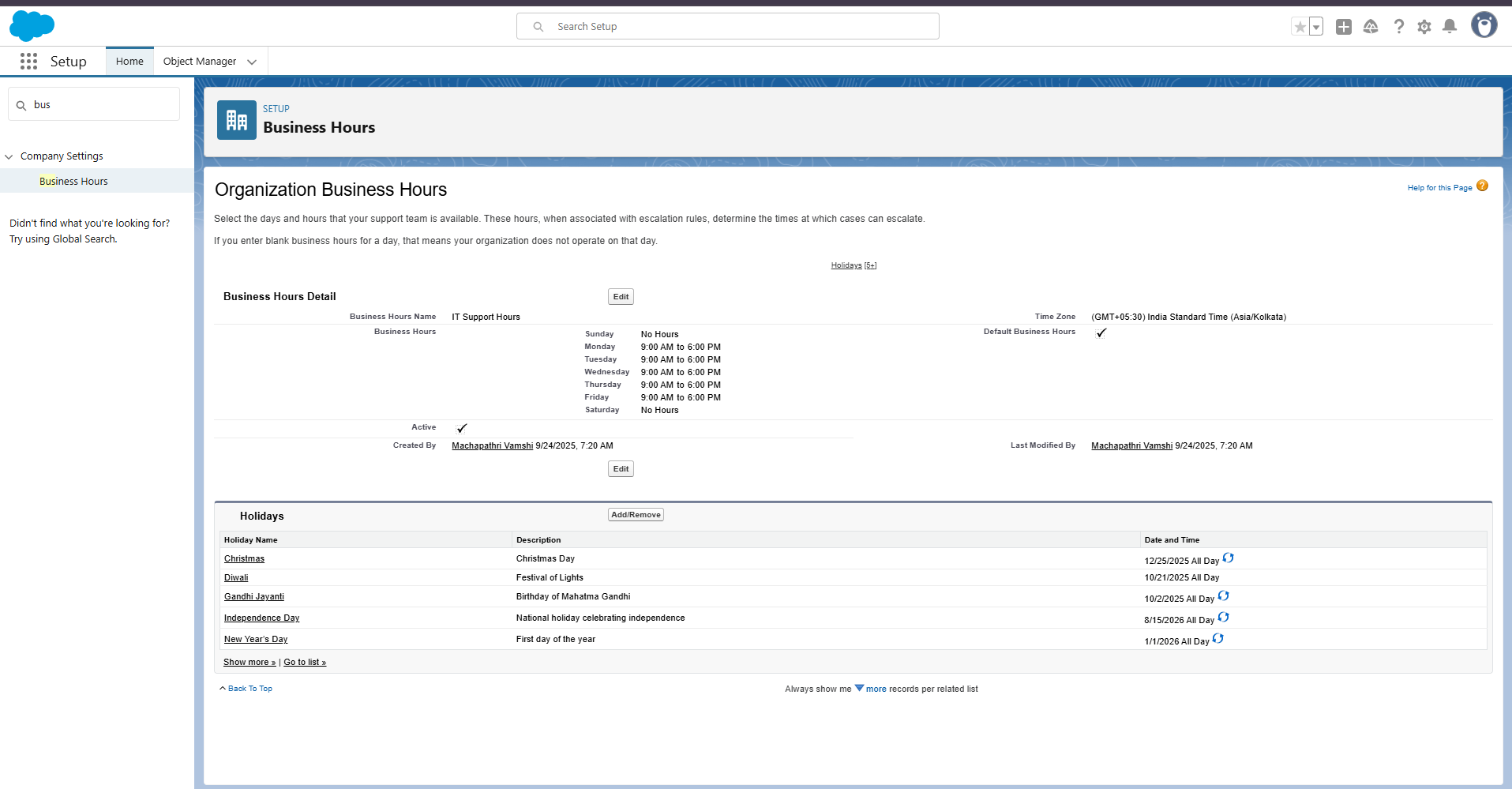


## Step 4: Add Holidays

1. Setup → Quick Find → **Holidays** → Click
2. Click **New Holiday**:

* Holiday Name: Republic Day
* Date: 26-Jan

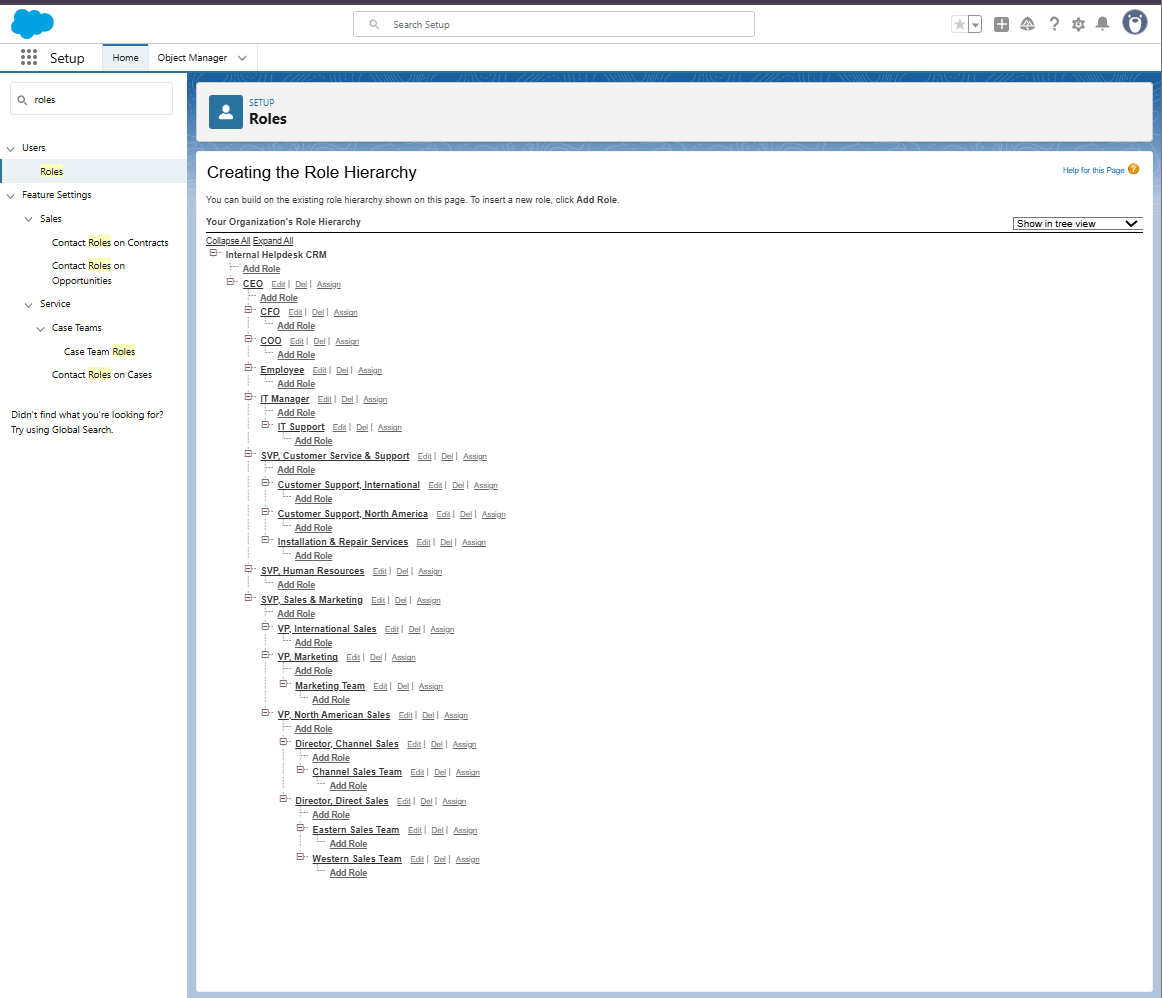
1. Click Save
2. Click **Add Business Hours** → add IT Support Hours



## Step 5: Create Role Hierarchy

1. Setup → Quick Find → Roles → Click Set Up Roles
2. Click Expand All
3. Add roles:

* CEO
* IT Manager
* IT Support
* Employee



## Step 6: Create Users

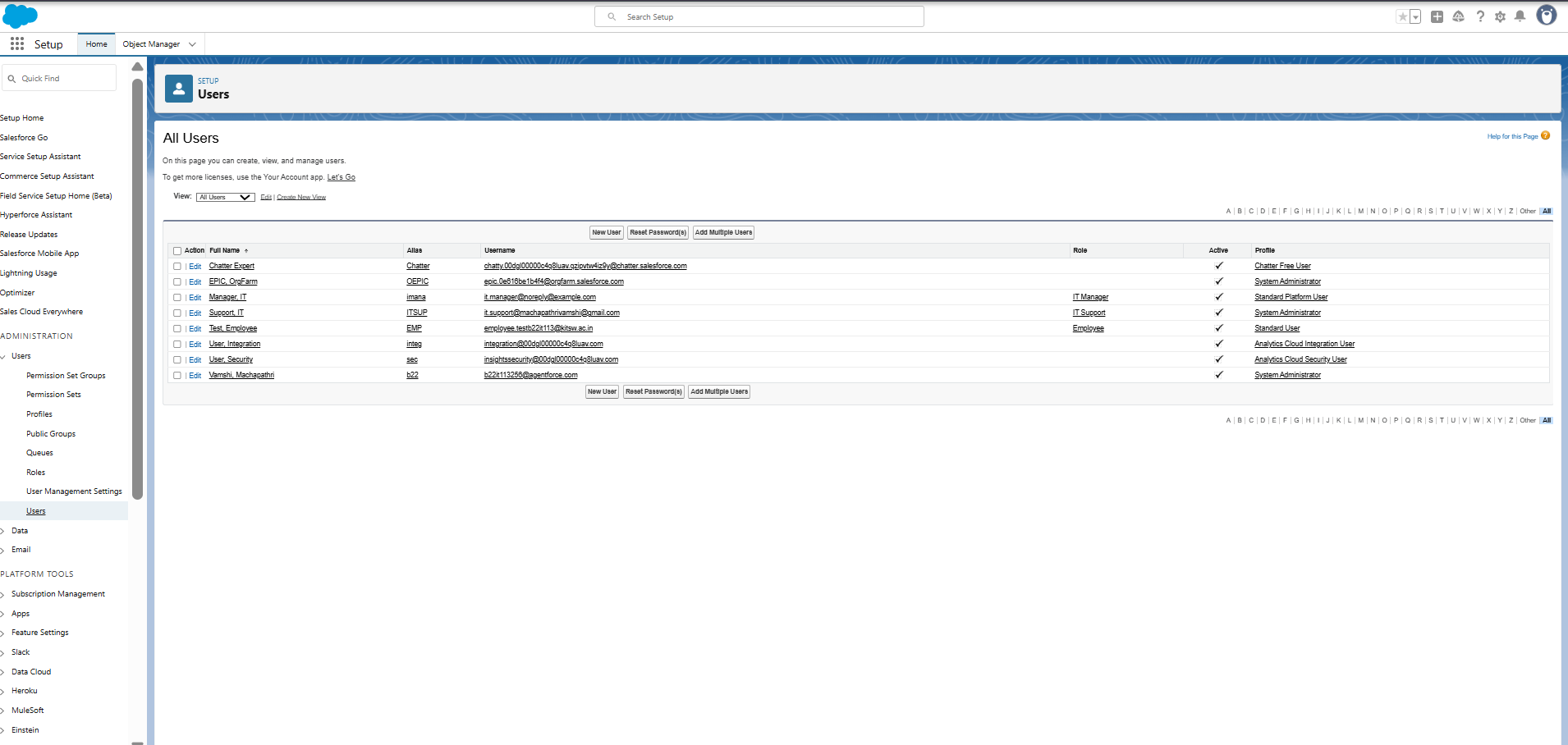
1. Setup → Quick Find → Users → Click New User
2. Employee User:

* First Name: Employee | Last Name: Test
* Alias: EMP
* Email: working email
* Username: unique (employee.test@internalhelpdesk.com)
* Profile: Standard User
* Role: Employee → Save

1. IT Support User:

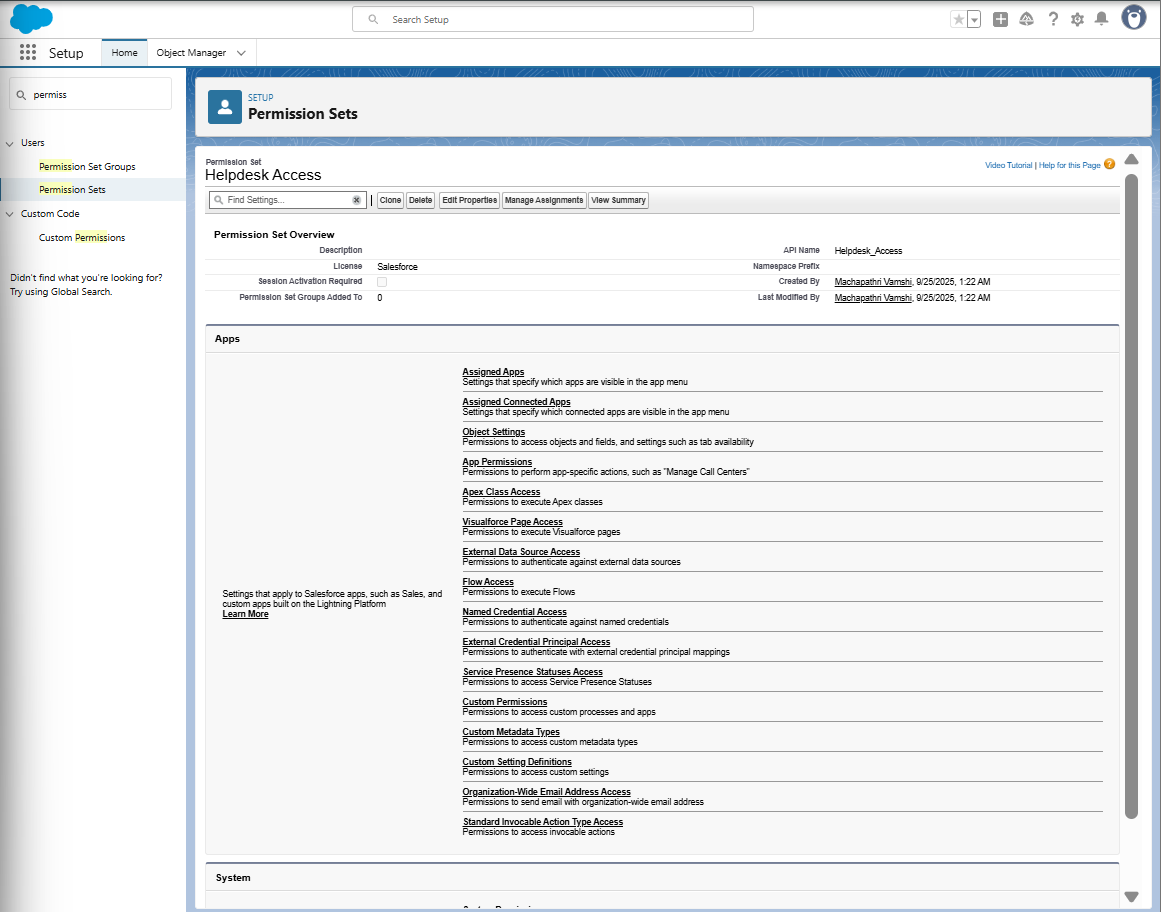
* Profile: System Administrator
* Role: IT Support → Save

1. (Optional) IT Manager user with role IT Manager



## Step 7: Create Permission Set

1. Setup → Quick Find → Permission Sets → Click New
2. Label: Helpdesk Access
3. API Name: auto-fills
4. Click Save
5. Assign object permissions after Phase 3

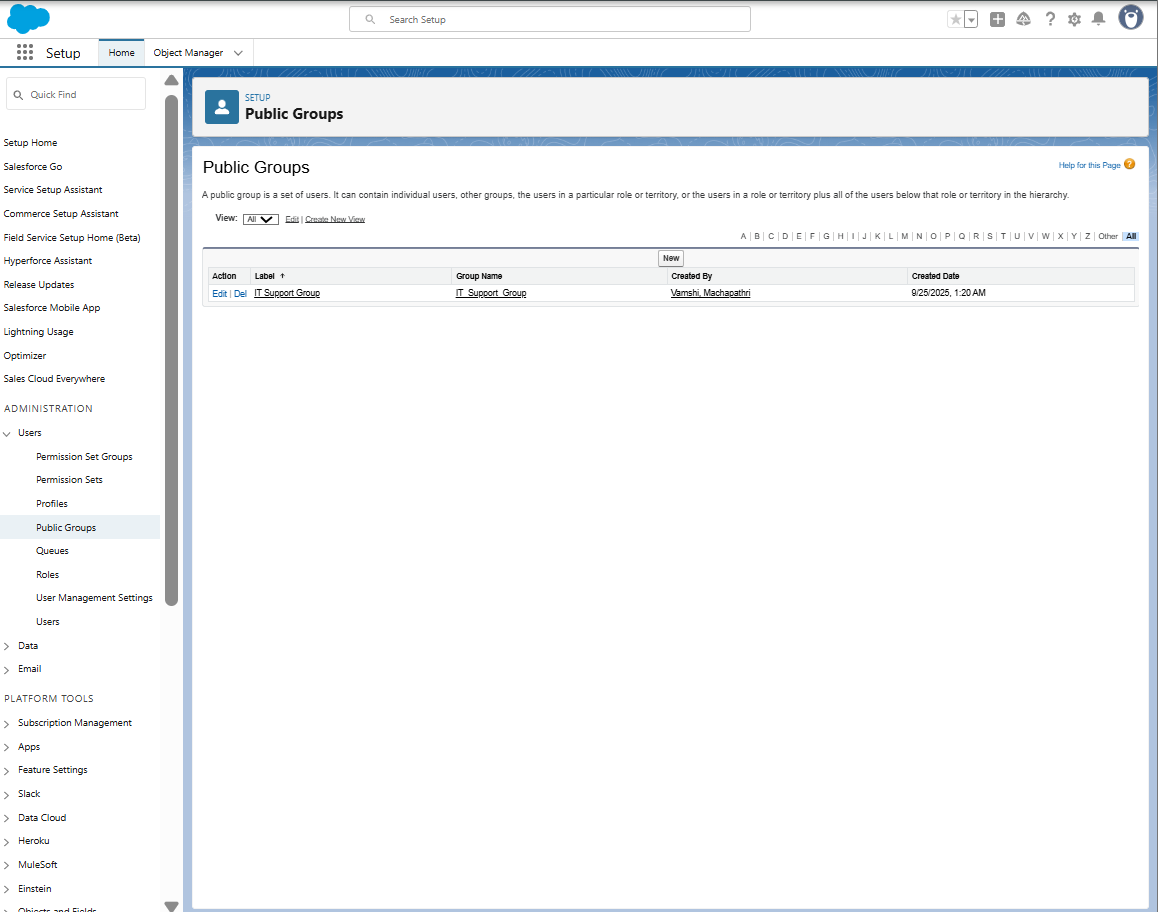


## Step 8: Prepare Org-Wide Defaults (OWD)

1. Setup → Quick Find → Sharing Settings → Click
2. Click Edit
3. Note: Helpdesk Ticket object will be Private after Phase 3
4. Click Save

## Step 9: Create Public Group

1. Setup → Quick Find → Public Groups → Click New
2. Label: IT Support Group
3. Add IT Support User → Save



## Step 10: Enable Admin Login Access

1. Setup → Quick Find → Login Access Policies → Click
2. Check Administrators Can Log in as Any User → Save



## Step 11: Create Email Templates

1. Setup → Quick Find → Email Templates → Click
2. New Email Template (Classic):

* Folder: Unfiled Public Email Templates
* Available For Use: ✔️
* Template Name: Ticket Submitted
* Subject: Your Helpdesk Ticket {!Helpdesk\_Ticket\_\_c.Name} has been submitted
* Text/HTML Body Example:

Hello {!Helpdesk\_Ticket\_\_c.Requester\_\_c},

Your ticket "{!Helpdesk\_Ticket\_\_c.Description\_\_c}" has been created successfully.

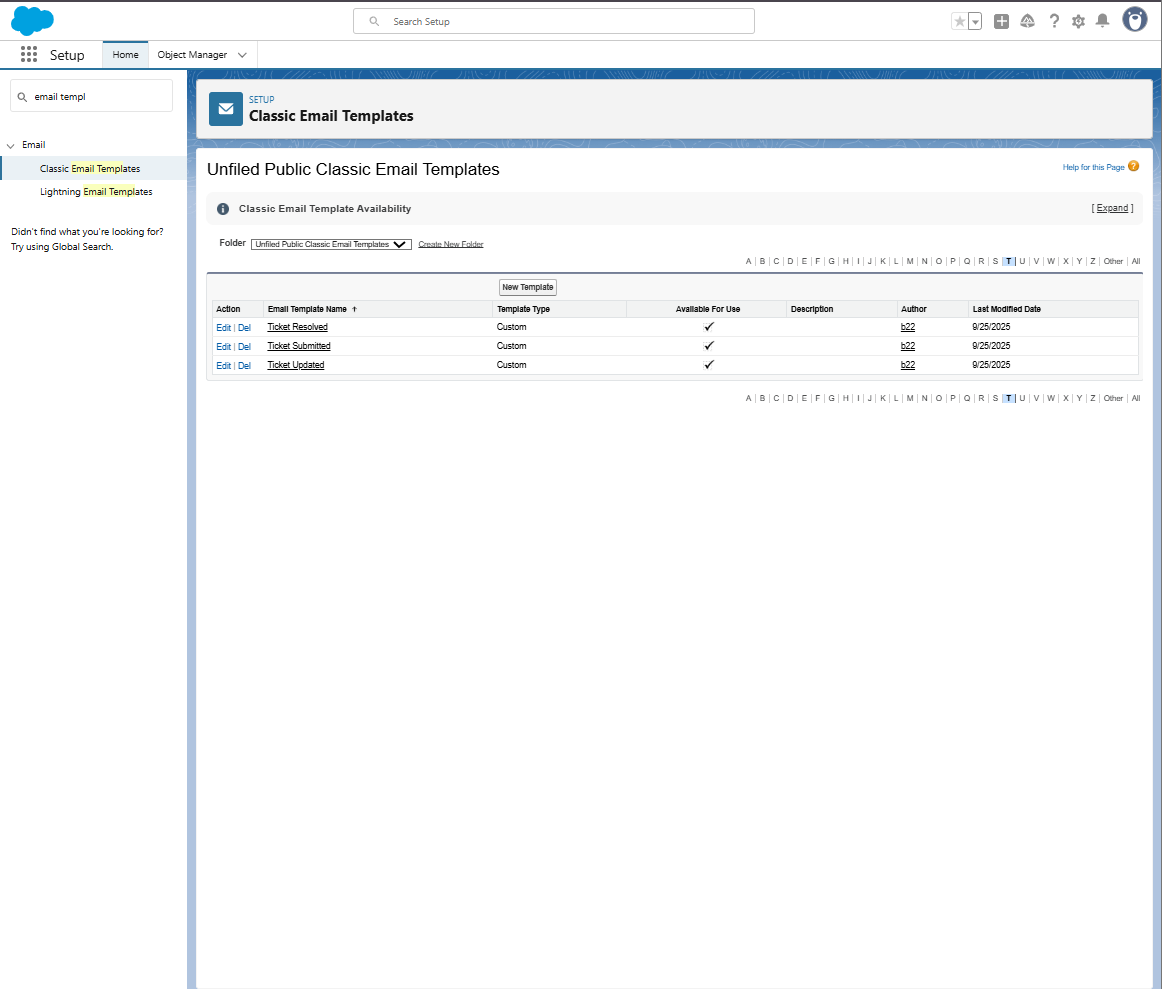
Ticket Number: {!Helpdesk\_Ticket\_\_c.Name}

Status: {!Helpdesk\_Ticket\_\_c.Status\_\_c}

Our IT Support team will review and get back to you soon.

Thanks,

IT Helpdesk Team

1. Save ✅
2. Repeat for Ticket Updated and Ticket Resolved
3. 

## 3. Final Confirmation Checklist

* Log in to Salesforce Org
* Set Company Information
* Define Business Hours
* Add Holidays
* Create Role Hierarchy
* Create Users
* Create Permission Set
* Prepare Org-Wide Defaults
* Create Public Group
* Enable Admin Login Access
* Create Email Templates (Ticket Submitted, Updated, Resolved)
* Assign object permissions after Phase 3

# ****Phase 3: Data Modeling & Relationships**** **Objective**:**** Build the Helpdesk Ticket object with fields, layouts, and relationships to manage internal IT tickets.

## ****Step 1: Create Custom Object – Helpdesk Ticket****

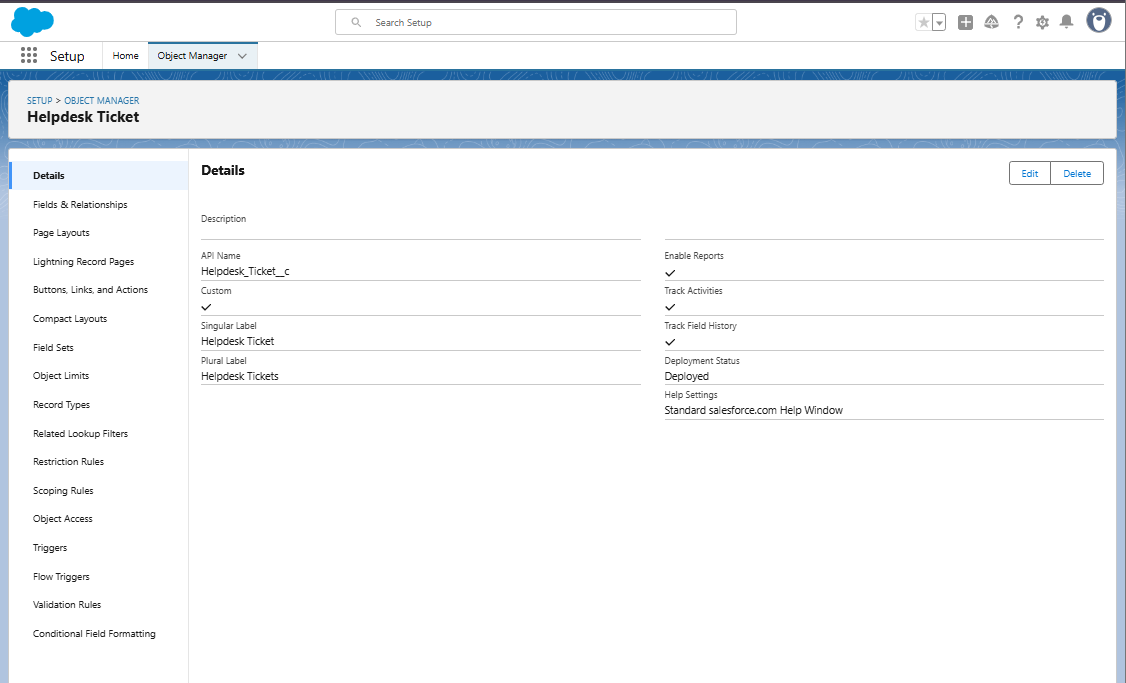
**Purpose:** Create a custom object to store all helpdesk tickets.

**Steps:**

1. Go to **Setup ⚙️ → Object Manager → Create → Custom Object**.
2. Fill details:

* **Label:** Helpdesk Ticket
* **Plural Label:** Helpdesk Tickets
* **Object Name:** Helpdesk\_Ticket
* **Record Name:** Auto Number → Format: HT-{0000}
* **Optional:** Description for documentation

1. Click **Save** ✅



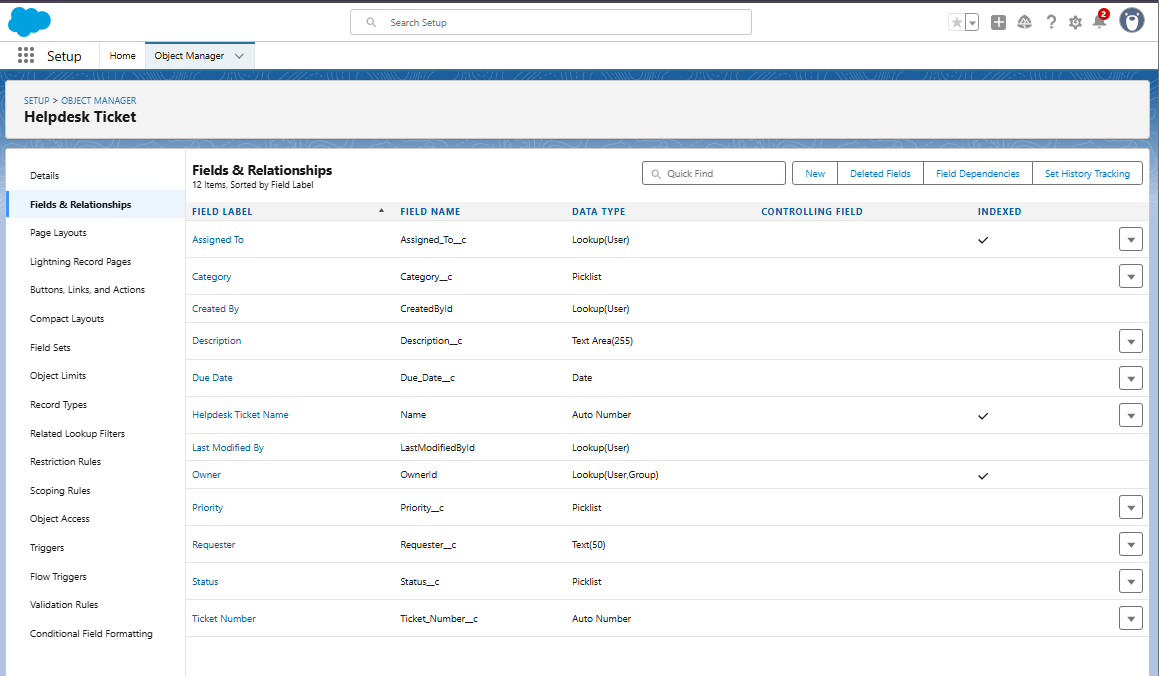
## ****Step 2: Add Custom Fields****

**Purpose:** Capture ticket information.

| **Field Label** | **Data Type** | **Details** | **Required** |
| --- | --- | --- | --- |
| Ticket Number | Auto Number | Format: HT-{0000} | checked |
| Description | Text Area | 255 characters | checked |
| Requester | Text | 50 characters | checked |
| Assigned To | Lookup (User) | – | Optional |
| Status | Picklist | New, In Progress, On Hold, Resolved, Closed | checked |
| Priority | Picklist | Low, Medium, High, Urgent | checked |
| Category | Picklist | Hardware, Software, Network, Other | Optional |
| Due Date | Date | – | Optional |
| Resolution Notes | Long Text Area | 500 characters | Optional |

**Steps to Create Fields:**

1. Object Manager → Helpdesk Ticket → **Fields & Relationships → New**.
2. Select **Data Type** → Next.
3. Fill **Field Details** → Next.
4. Set **Field-Level Security** → Next.
5. Add to **Page Layout** → Save.
6. Repeat for all fields.



## Step 3: Configure Page Layout

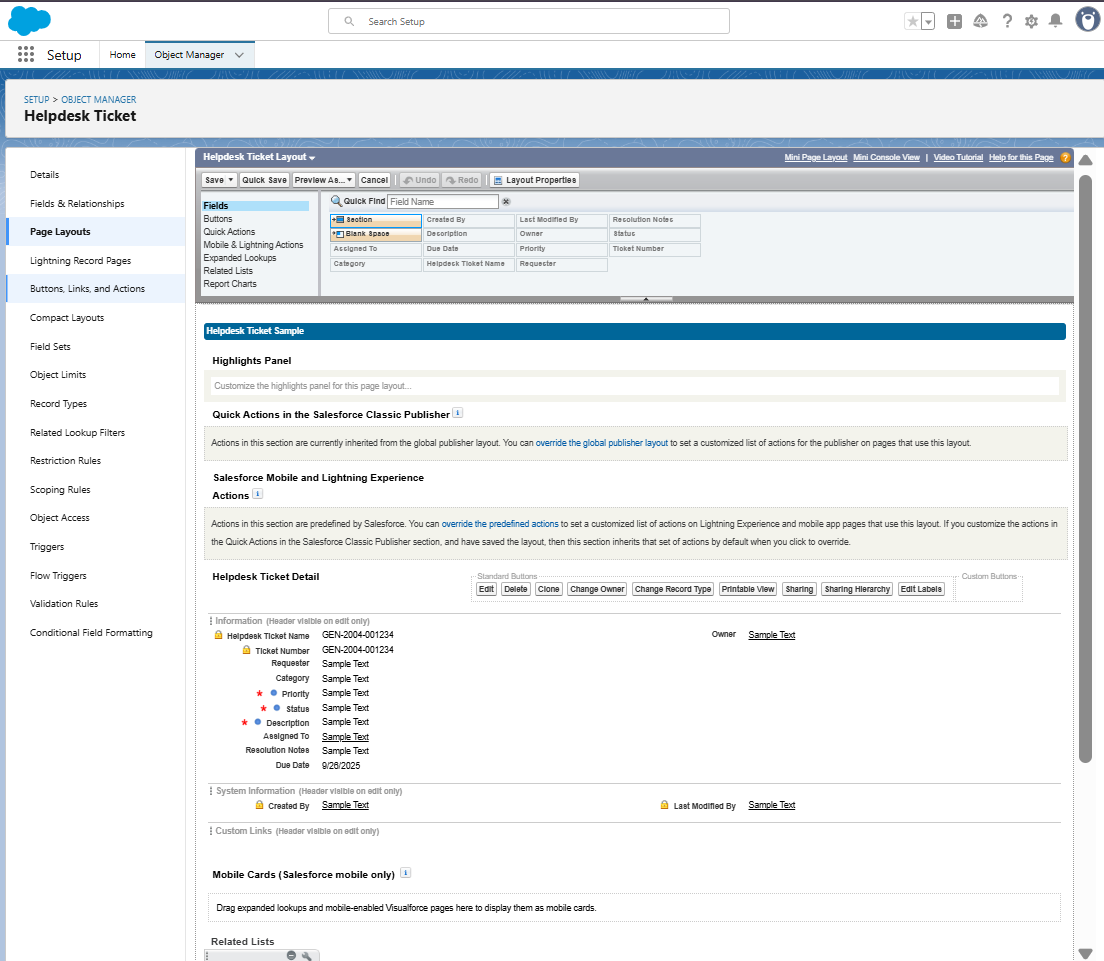
Purpose: Organize fields, sections, and related lists on the record page.

## Steps:

1. Object Manager → Helpdesk Ticket → Page Layouts → Edit.
2. Organize sections:

* **Ticket Information**: Ticket Number, Description, Requester, Assigned To
* Details: Status, Priority, Category, Due Date
* Resolution: Resolution Notes
* System Information: Created By, Last Modified By, Owner

1. Configure Related Lists: Open Activities, Activity History
2. Use Highlights Panel for key fields: Ticket Number, Status, Priority, Assigned To
3. Save ✅



## Step 4: Enable Field History Tracking

Purpose: Track changes to critical fields.

Steps:

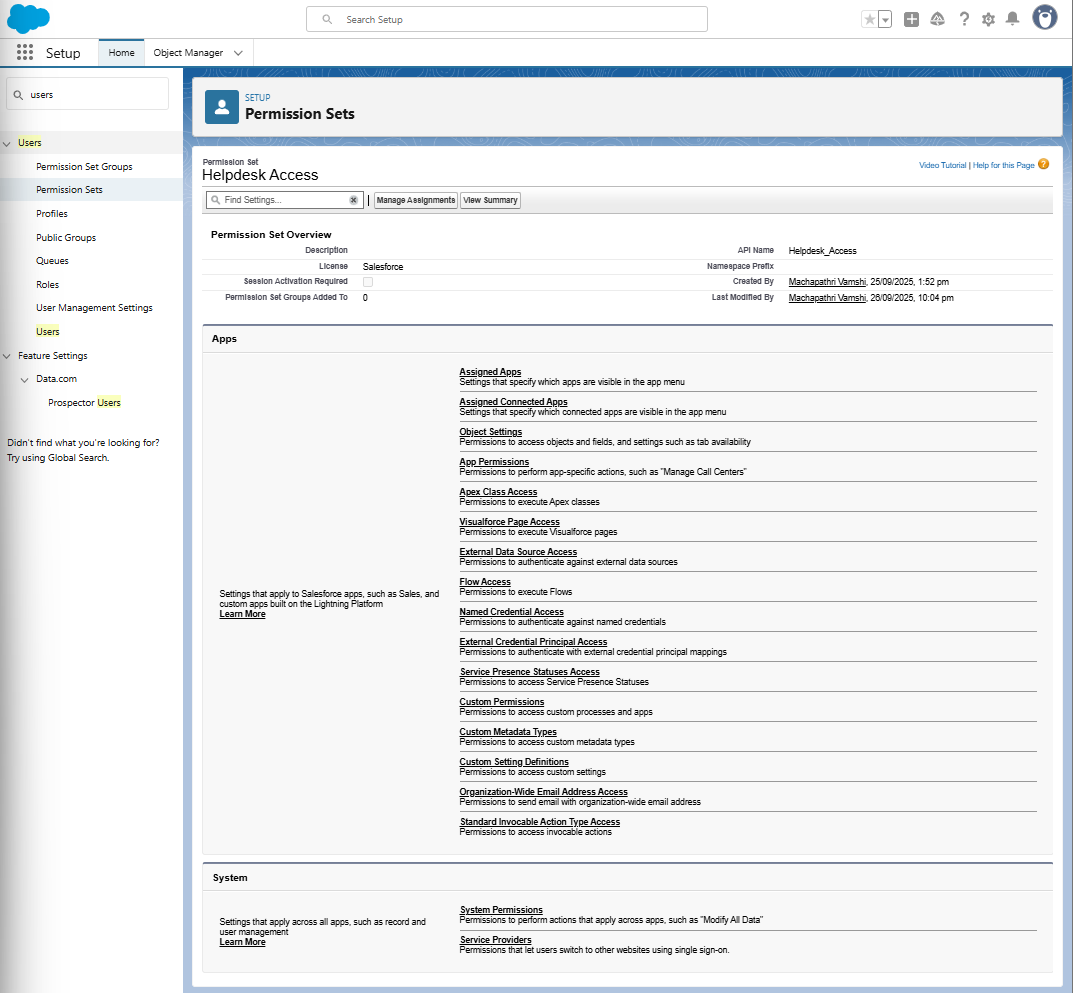
1. Object Manager → Helpdesk Ticket → Fields & Relationships → Set History Tracking.
2. Select fields: Assigned To, Description, Status, Priority, Category, Due Date, Resolution Notes
3. Enable Track old and new values → Save ✅
4. In Page Layout → Drag Helpdesk Ticket History related list → Save

## Step 5: Permission Set – Helpdesk Access

Purpose**:** Control object and field permissions for users.

## Steps:

1. Setup → Quick Find → **Permission Sets → Helpdesk Access**.
2. Click **Object Settings → Helpdesk Ticket → Edit**.
3. Grant: **Read, Create, Edit** (do not give Delete unless needed).
4. Click **Field Permissions** → adjust visibility for sensitive fields like Resolution Notes
5. Assign to users: **Manage Assignments → Add Assignments → Select Employee users → Assign**



# Step 6: Org-Wide Defaults (OWD) — set Private

1. Setup → Quick Find → **Sharing Settings** → click.
2. Under **Organization-Wide Defaults**, find **Helpdesk Ticket** → set **Default Internal Access** = **Private**.
3. Click **Save**

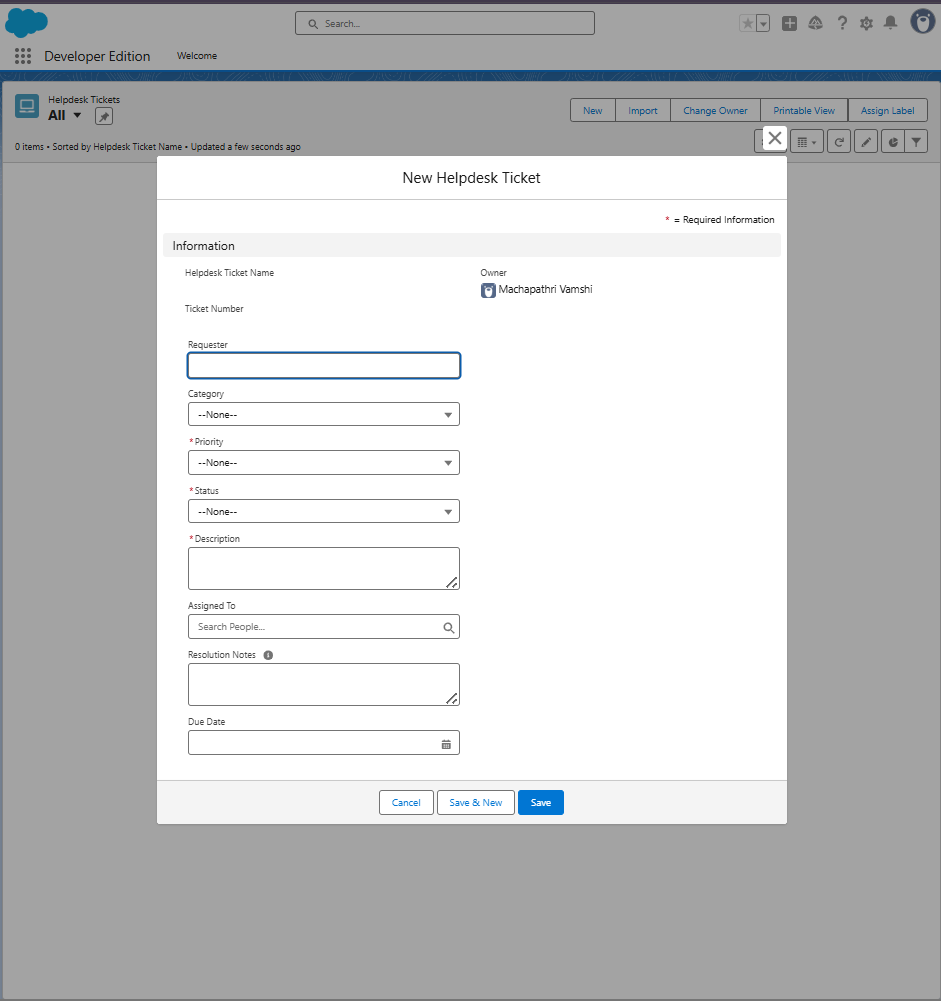
## **Screenshot 2025-09-26 225021**

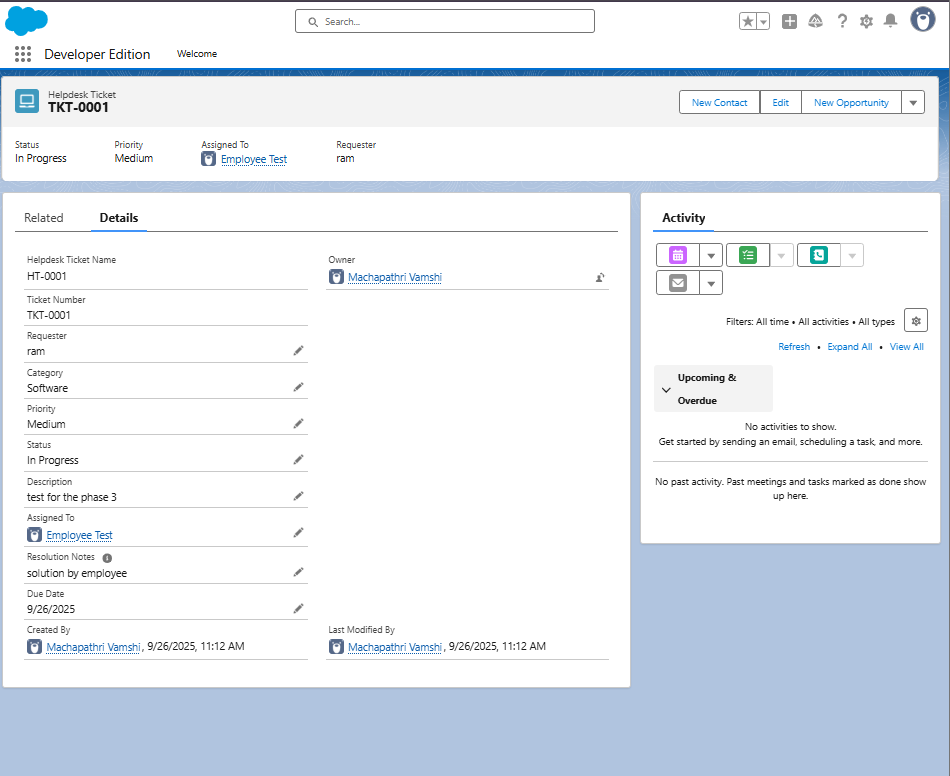
## ****Step 7: Test Object Access****

Purpose**:** Ensure correct access and functionality.

## Steps:

1. Log in as **Employee user** → Create ticket → check all fields
2. Log in as **IT Support user** → View/update tickets → verify access
3. Log in as **System Administrator** → Confirm full access





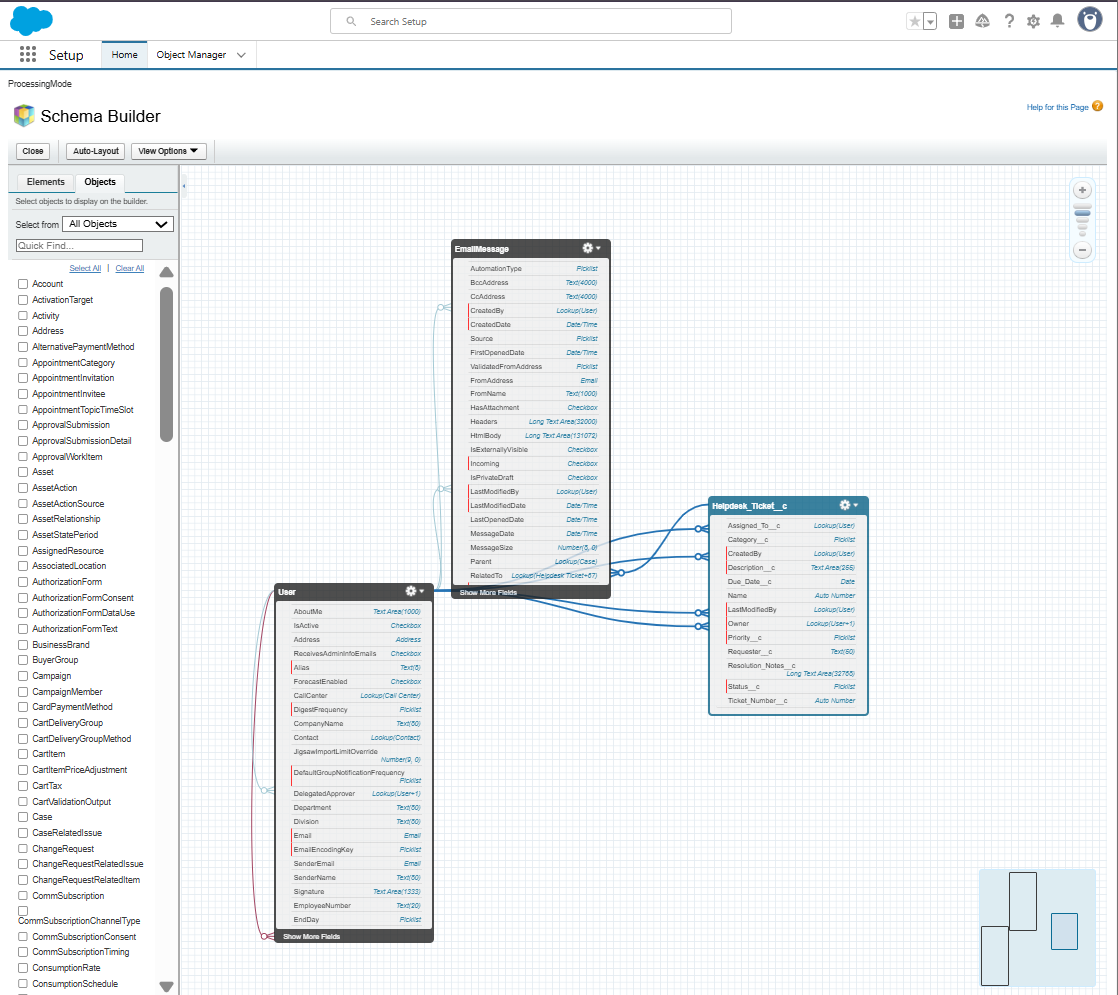
# Optional Advanced Features

# Step 8: Schema Builder

Purpose**:** Visualize objects, fields, and relationships.

## Steps:

1. Setup → Quick Find → **Schema Builder → Click**
2. Select objects: **Helpdesk Ticket, User, Role** (optional), related objects
3. Drag and drop to organize layout for better visualization
4. Take screenshots for documentation



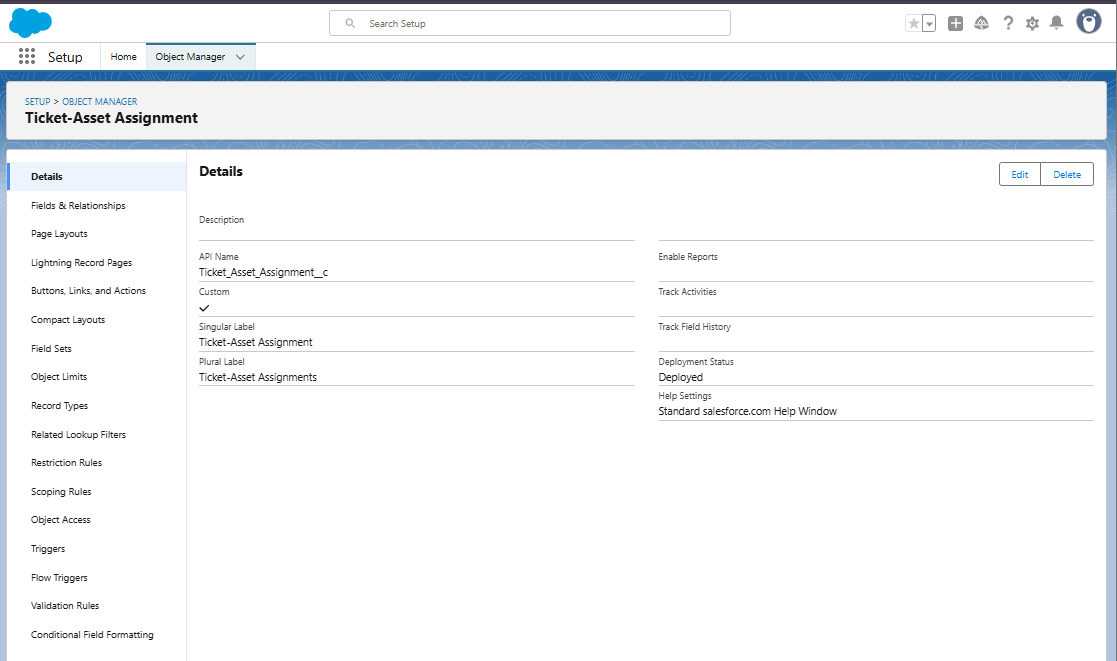
# Step 9: Junction Objects (Optional)

Purpose**:** Create many-to-many relationships.

## Steps:

1. Setup → Object Manager → Create → Custom Object → Name: Ticket-Asset Assignment
2. Record Name: Auto Number → TAA-0001 → Save
3. Create **Master-Detail Relationships**:

* Helpdesk Ticket → Save
* Asset → Save



## Step 10: Validation Rule — require Resolution Notes when Status = Resolved

1. Object Manager → Helpdesk Ticket → **Validation Rules** → **New**.
2. Rule Label: Require\_Resolution\_Notes\_When\_Resolved
3. Formula (copy exactly):

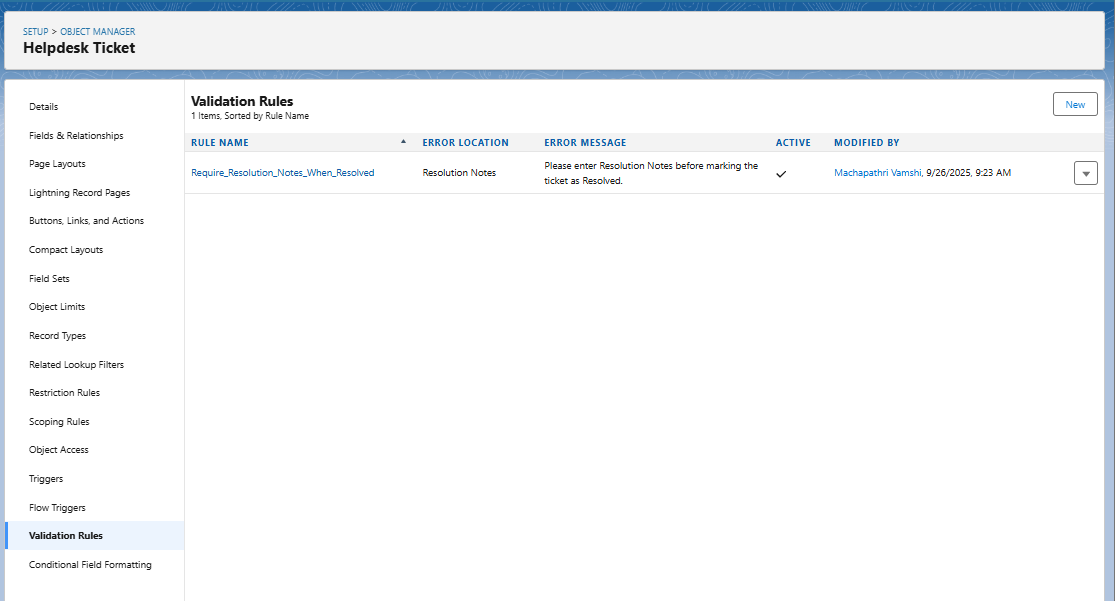
AND(

ISPICKVAL(Status\_\_c, "Resolved"),

ISBLANK( Resolution\_Notes\_\_c )

)

1. Error Message: Please enter Resolution Notes before marking the ticket as Resolved.
2. Error Location: **Field** → choose Resolution Notes → Save.



# ****Phase 3 Completion – Data Modeling & Relationships****

* Custom Object: **Helpdesk Ticket** created with Ticket Number (Auto Number).
* Custom Fields: Description, Requester, Assigned To, Status, Priority, Category, Due Date, Resolution Notes.
* Page Layouts: Configured with sections, highlights panel, and related lists.
* Field History Tracking: Enabled for key fields.
* Permission Set: **Helpdesk Access** created and assigned.
* Field-Level Security: Resolution Notes restricted for Employee users.
* Schema Builder: Used to visualize objects and relationships.
* Validation Rules: Configured (e.g., Due Date ≥ Today).
* Automation: Flows / Process Builder for ticket emails.
* Testing: Verified Employee, IT Support, and System Admin access.

# Phase 4: Process Automation (Admin)

## Objective

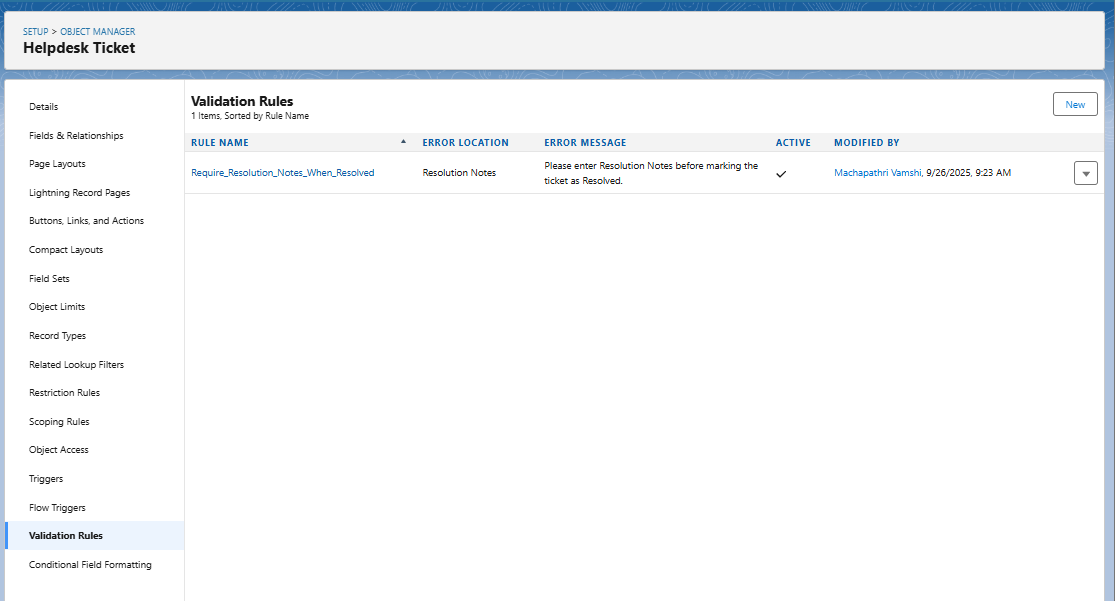
To automate Helpdesk Ticket operations such as email alerts, field updates, and validations using Salesforce automation tools — including Flows, Validation Rules, Process Builder, and Email Alerts.

## Step 1: Validation Rule

Goal: Ensure all required data is entered before saving records.

Example Rule – Due Date cannot be past:

1. Go to Setup → Object Manager → Helpdesk Ticket → Validation Rules → New.
2. Enter Rule Name: Validate\_Due\_Date.
3. Enter Error Condition Formula: Due\_Date\_\_c < TODAY()
4. Enter Error Message: “Due Date cannot be earlier than today.”
5. Location: Top of Page.
6. Click Save.



## Step 2: Create Record-Triggered Flow (Ticket Status Email Flow)

1. Setup → Quick Find → Flows → New Flow.
2. Choose Record-Triggered Flow.
3. Object: Helpdesk Ticket.
4. Trigger: A record is created or updated.
5. Optimize for: Actions and Related Records.
6. Click Done.

## Screenshot 2025-09-27 113537

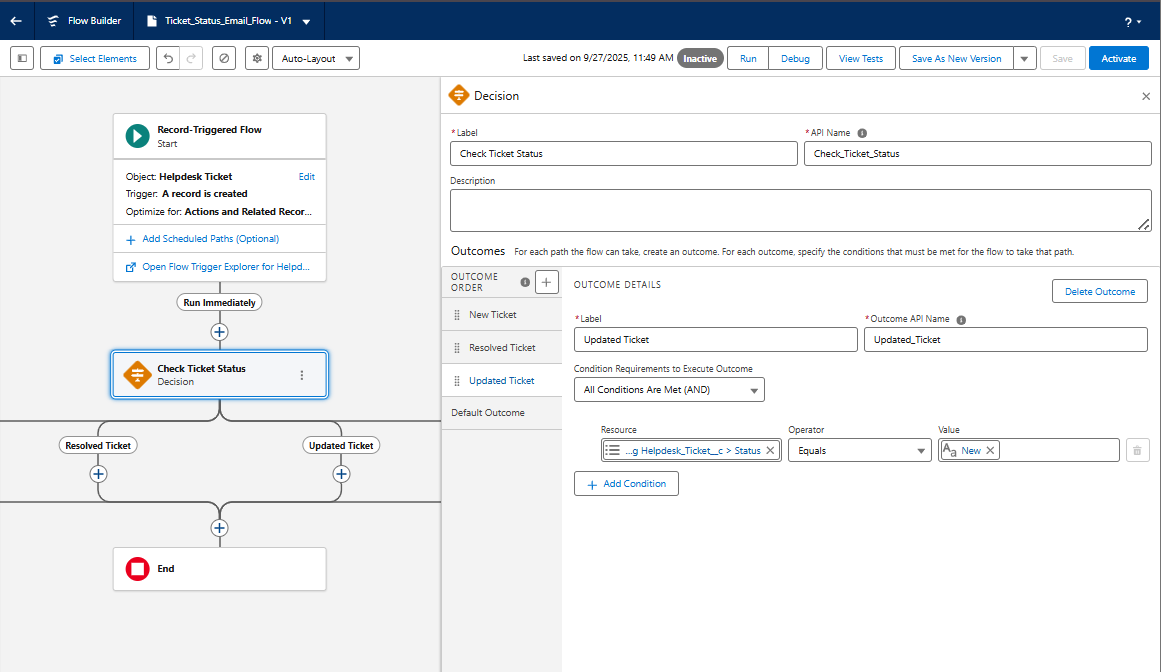
## Step 3: Add Decision Element

Goal: Decide which email to send based on Status.

From Flow Builder → Add Decision → Label it 'Check Ticket Status'.

Outcomes:

|  |  |
| --- | --- |
| Outcome Name | Condition |
| Ticket Submitted | Status\_\_c Equals 'New' |
| Ticket Updated | Status\_\_c Equals 'In Progress' OR Status\_\_c Equals 'On Hold' |
| Ticket Resolved | Status\_\_c Equals 'Resolved' |



## Step 4: Create Email Templates

Go to Setup → Email Templates → New Email Template. Create 3 templates:

1. Ticket Submitted  
2. Ticket Updated  
3. Ticket Resolved

Use merge fields like:

Hello {!Helpdesk\_Ticket\_\_c.Requester\_User\_\_c.Name},  
  
Your ticket "{!Helpdesk\_Ticket\_\_c.Description\_\_c}" has been updated.  
  
Status: {!Helpdesk\_Ticket\_\_c.Status\_\_c}

## Step 5: Configure Email Actions

In Flow → after Decision Node → Add Action → Send Email (Core Action)

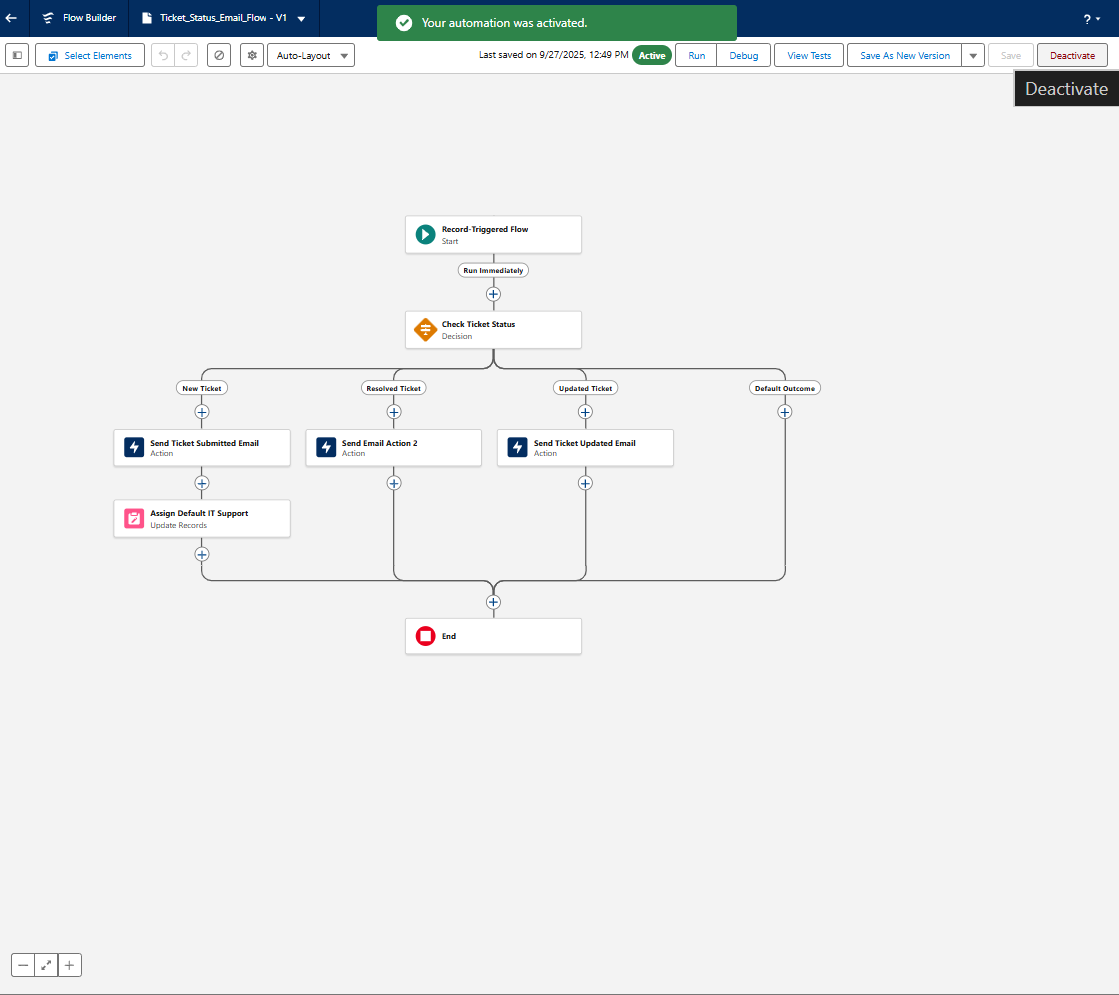
Example: Send Ticket Submitted Email

|  |  |
| --- | --- |
| Field | Value |
| Label | Send Ticket Submitted Email |
| Recipient ID | Triggering Record → Requester\_User\_\_c |
| Sender Type | Current User |
| Email Template ID | Ticket Submitted |
| Related Record ID | $Record.Id |
| Log Email on Send | Checked |

Repeat for Ticket Updated and Ticket Resolved emails.

## Step 6: Save and Activate Flow

Click Save → Name Flow 'Ticket\_Status\_Email\_Flow' → Activate.



## Step 7: Testing the Automation

1. Log in as Employee → Create a ticket with Status = New → Check email inbox.
2. Update ticket to In Progress → Check Ticket Updated email.
3. Update ticket to Resolved → Check Ticket Resolved email.

## IMG_256Step 8: Optional Automations

Workflow Rule: Send alerts for high priority tickets.

Process Builder: Automate field updates or task creation.

Custom Notifications: Push alerts to IT Support when ticket is created.

## Step 9: Summary

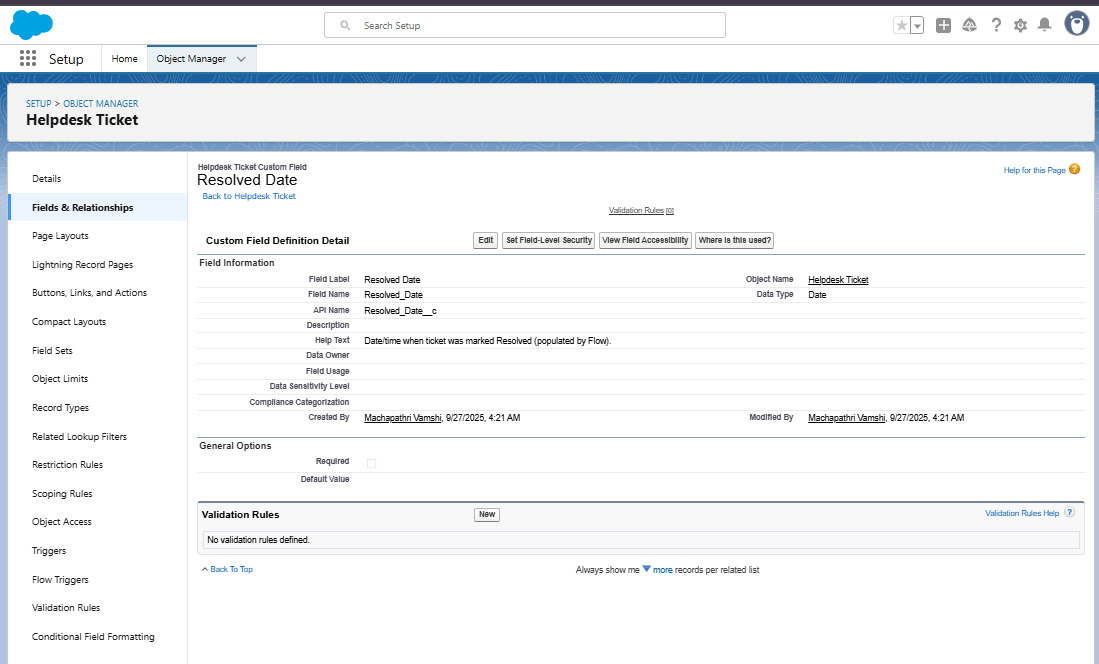
|  |  |
| --- | --- |
| Tool | Purpose |
| Validation Rule | Prevent invalid data |
| Record-Triggered Flow | Automate email alerts |
| Email Templates | Predefined messages |
| Decision Element | Control logic |
| Workflow Rule / Process Builder | Legacy automation options |

## Phase 4 Completion

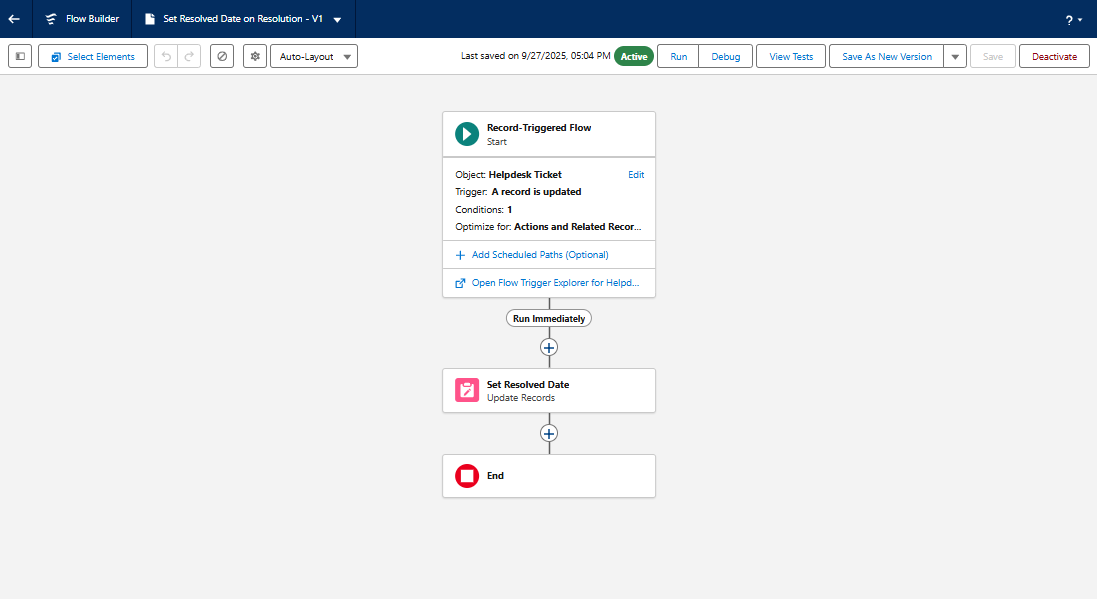
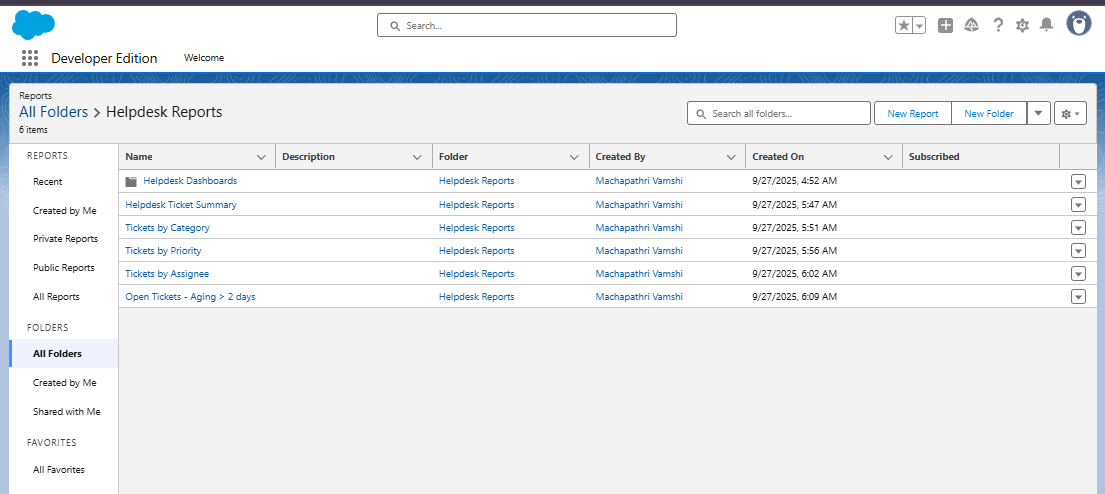
* Validation Rules configured
* Flow created & activated
* Email Templates linked
* Testing completed successfully
* Screenshot Placeholder: Flow Overview & Run History

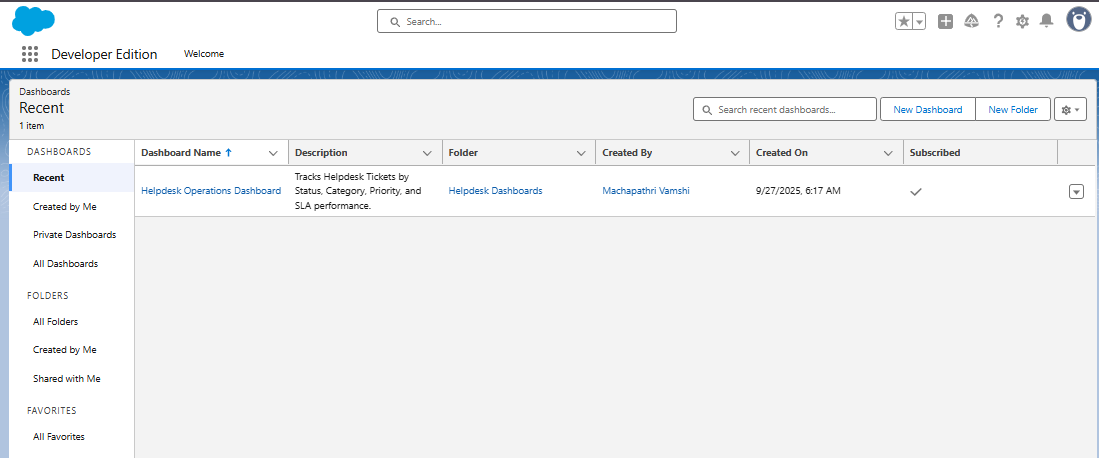
# Phase 5 — Reports & Dashboards

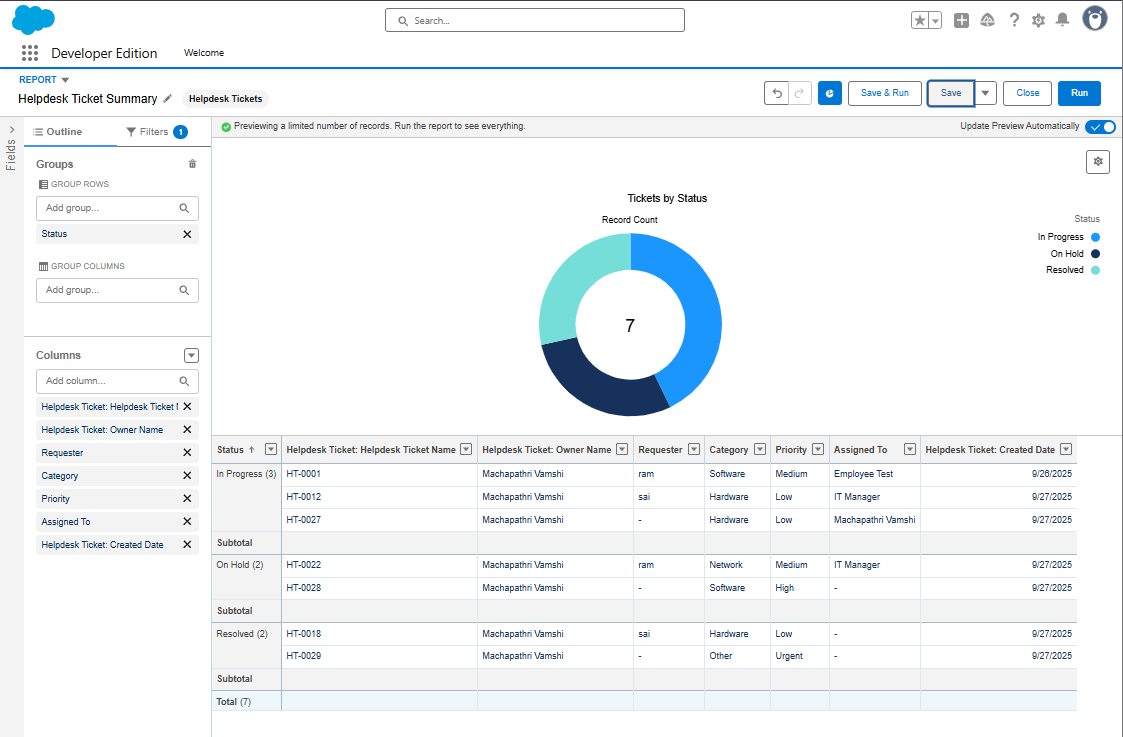
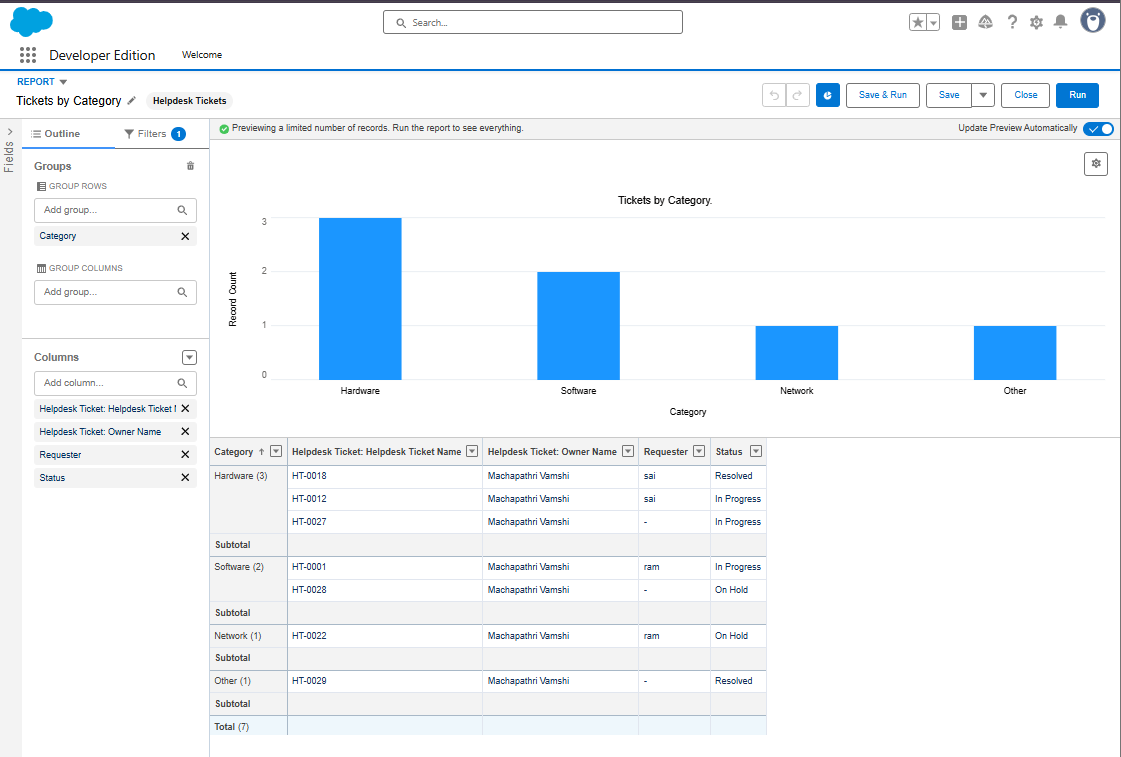
PRE-CHECK (do this first)  
Log in as System Administrator.  
Confirm Helpdesk Ticket object exists: Setup → Object Manager → Helpdesk Ticket.  
Confirm there are a few sample records (6–10) for testing.  
  
PART A — Add two helper fields (Age and Resolved Date + Resolution Hours Formula)  
**A1 — Create Resolved Date (Date/Time)**  
- Setup → Object Manager → Helpdesk Ticket → Fields & Relationships → New.  
- Data Type: Date/Time → Next.  
- Field Label: Resolved Date → API Name: Resolved\_Date\_\_c.  
- Help Text: Date/time when ticket was marked Resolved.  
- Visible on all layouts → Save.

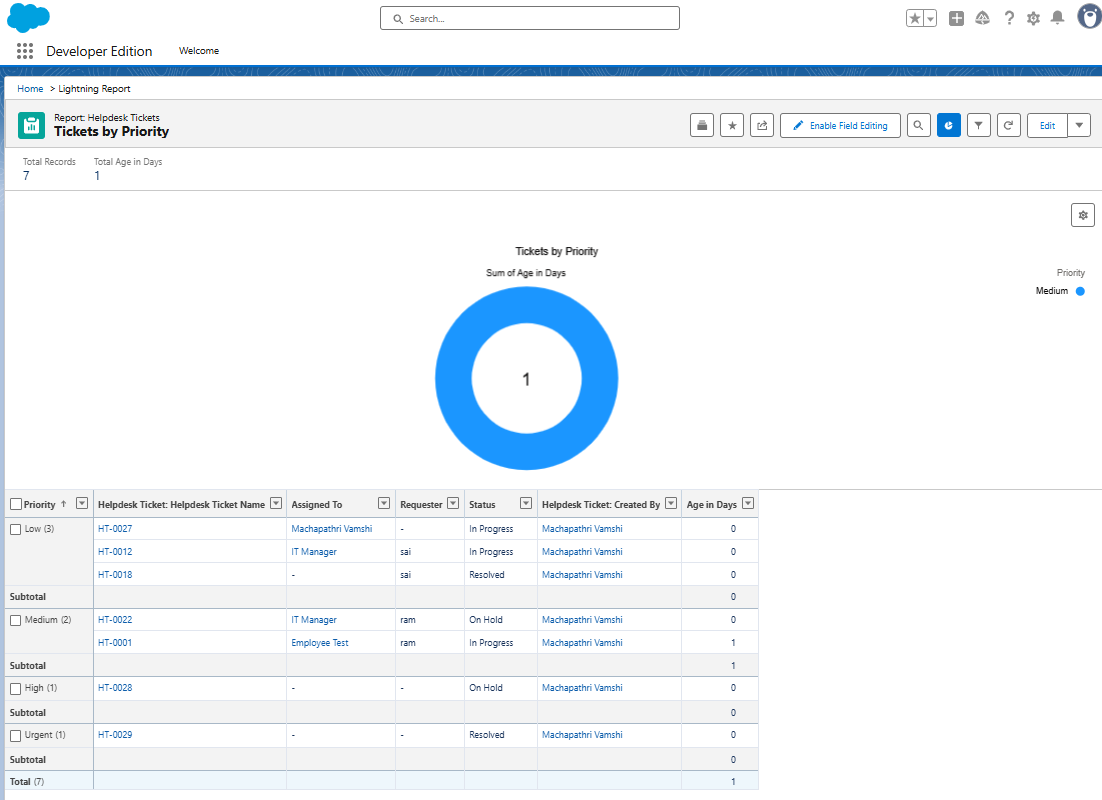
  
  
**A2 — Create Resolution Time (Hours) (Formula Number)**  
- Data Type: Formula → Return Type: Number (2 decimals).  
- Formula: **IF(  
 ISBLANK(Resolved\_Date\_\_c),  
 0,  
 (Resolved\_Date\_\_c - CreatedDate) \* 24  
 )  
- Check syntax and save.**  
  
A3 — Create Age in Days (Formula Number)  
Formula: **TODAY() - DATEVALUE(CreatedDate)  
Save and add to layout.**

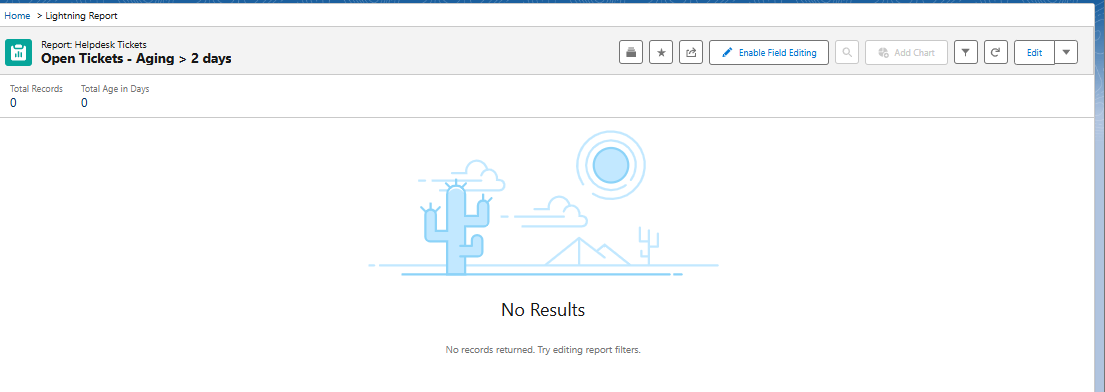
PART B — Flow to populate Resolved Date  
- Setup → Flows → New Flow → Record-Triggered Flow.  
- Object: Helpdesk Ticket.  
- Condition: Status = Resolved AND Resolved\_Date\_\_c IS BLANK.  
- Action: Update Record → Resolved\_Date\_\_c = $Flow.CurrentDateTime.  
- Save and Activate.

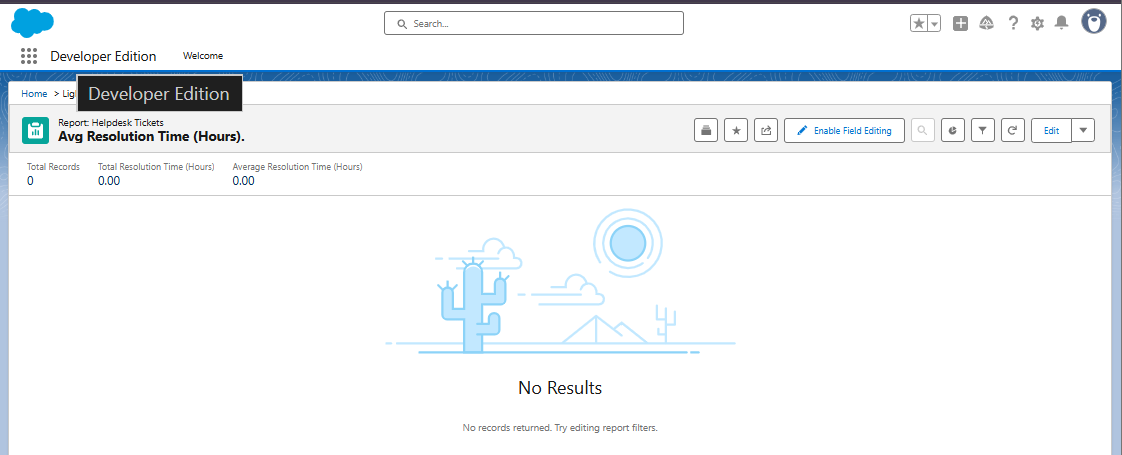
  
  
PART C — Create Report Folders  
C1 — Reports Folder: Helpdesk Reports  
C2 — Dashboard Folder: Helpdesk Dashboards  


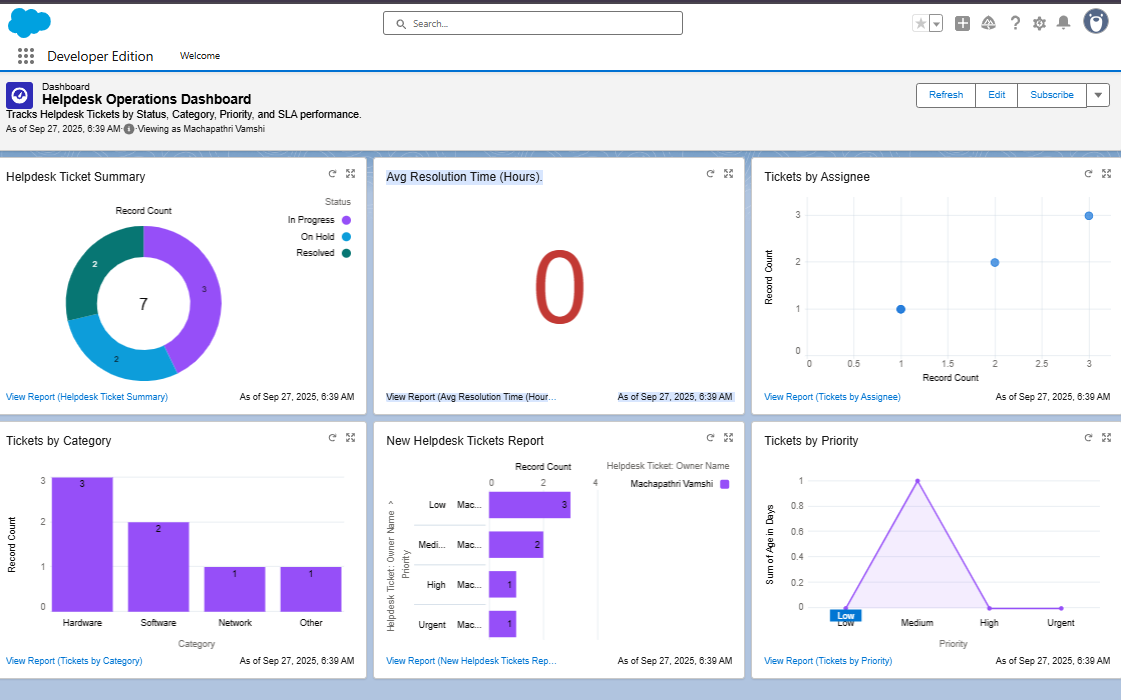


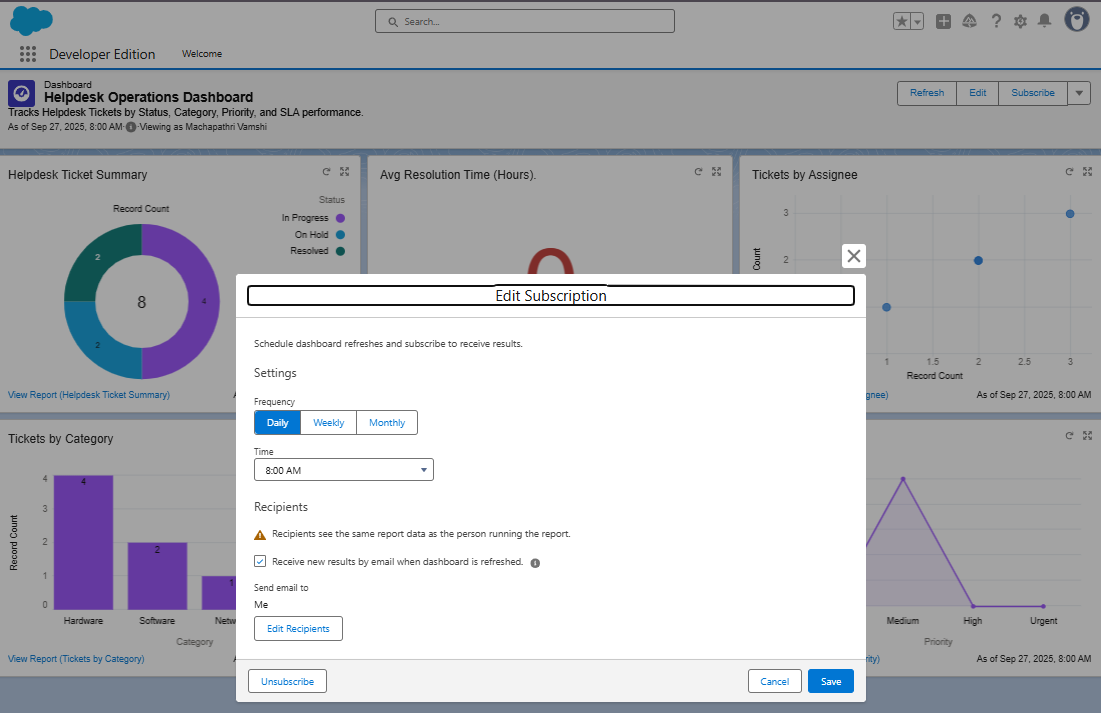
PART D — Build Reports  
D1 — Helpdesk Ticket Summary (by Status)  
- Group Rows by Status.  
- Add Donut Chart.  
  
D2 — Tickets by Category  
- Group Rows by Category.  
- Add Bar Chart.  
  
D3 — Tickets by Priority  
- Group Rows by Priority.  
- Add Donut Chart.

  
  
D4 — Tickets by Assignee  
- Group Rows by Assigned To.  
- Add Horizontal Bar Chart.

  
  
D5 — Open Tickets Aging (>2 Days)  
- Filter: Status not Resolved, Age\_in\_Days\_\_c > 2.  
- Add Table format.  
  
D6 — Average Resolution Time (Hours)  
- Filter: Resolved Date not Blank.  
- Add Summary on Resolution Time (Hours) column.  
- Add Gauge Chart.

  
  
PART E — Create Dashboard  
E1 — Dashboard Name: Helpdesk Operations Dashboard.  
E2 — Add Components:  
 1. Tickets by Status (Donut)  
 2. Tickets by Category (Bar)  
 3. Tickets by Assignee (Horizontal Bar)  
 4. Open Tickets Aging (Table)  
 5. Avg Resolution Time (Metric)  
 6. Tickets by Priority (Stacked Bar)  
E3 — Add Filters: Status, Assigned To.

  
  
PART F — Schedule Refresh  
- Dashboard → Subscribe → Daily 8 AM.  
- Reports → Subscribe → Daily 9 AM.

  
  
PART G — Sharing & Permissions  
- Share “Helpdesk Reports” and “Helpdesk Dashboards” folders with IT Support Group.  
.  
  
FINAL CHECKLIST  
✓ Created Resolved\_Date\_\_c (Date/Time)  
✓ Created Resolution\_Time\_Hours\_\_c (Formula)  
✓ Created Age\_in\_Days\_\_c (Formula)  
✓ Flow activated  
✓ Reports & Dashboards created  
✓ Dashboard filters added  
✓ Dashboard scheduled  
✓ Tests verified

# **Phase 5 – Apex Programming (Developer)**

Objective:  
To develop backend business logic using Apex classes, triggers, collections, SOQL, SOSL, asynchronous processing, and test classes to automate Helpdesk Ticket workflows efficiently.

Step 1 — Create Apex Class  
Path: **Setup → Developer Console → File → New → Apex Class  
Class Name: TicketHelper**

public with sharing class TicketHelper {

public static String greetUser(String userName) {

return 'Welcome to Helpdesk, ' + userName + '!';

}

public static List<Helpdesk\_Ticket\_\_c> getOpenTickets() {

return [SELECT Id, Name, Status\_\_c FROM Helpdesk\_Ticket\_\_c WHERE Status\_\_c != 'Closed'];

}

}



Step 2 — Apex Trigger (Before Insert / After Update)  
Trigger Name: HelpdeskTicketTrigger

trigger HelpdeskTicketTrigger on Helpdesk\_Ticket\_\_c (before insert, after update) {

if (Trigger.isBefore && Trigger.isInsert) {

for (Helpdesk\_Ticket\_\_c t : Trigger.new) {

if (t.Priority\_\_c == null) t.Priority\_\_c = 'Medium';

if (t.Status\_\_c == null) t.Status\_\_c = 'New';

}

}

if (Trigger.isAfter && Trigger.isUpdate) {

for (Helpdesk\_Ticket\_\_c t : Trigger.new) {

if (t.Status\_\_c == 'Resolved' && t.Resolved\_Date\_\_c == null) {

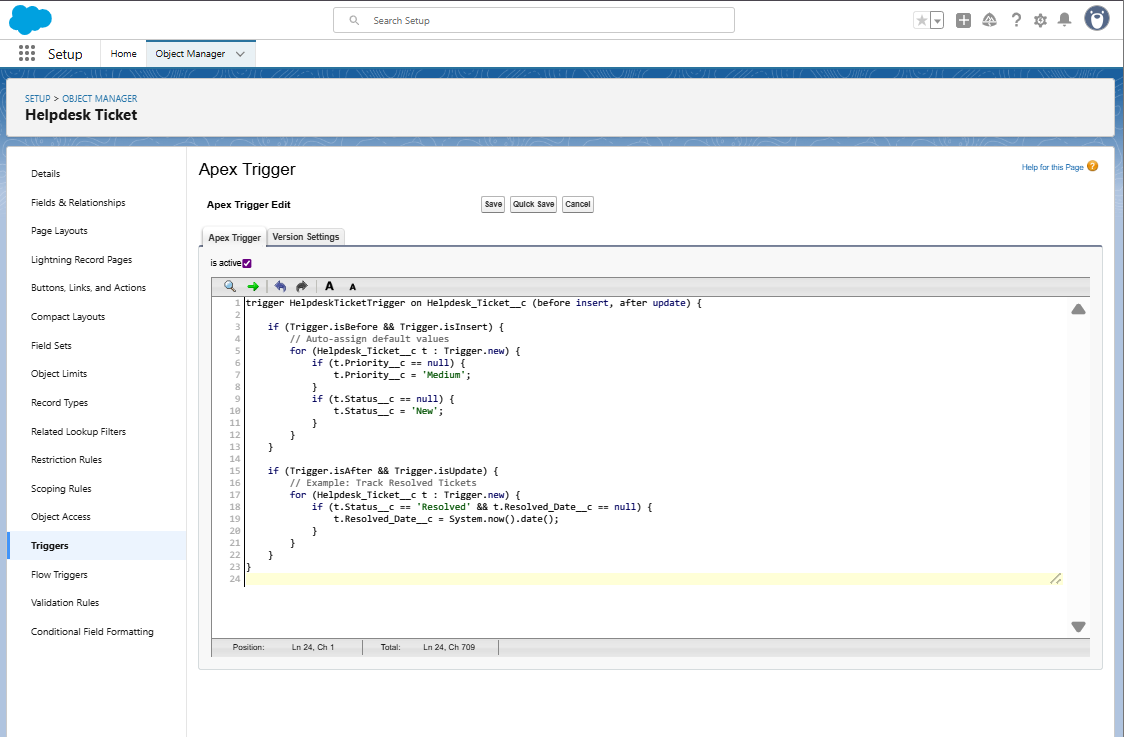
t.Resolved\_Date\_\_c = System.today();

}

}

}

}



Step 3 — Trigger Design Pattern (Handler Class)  
Class Name: HelpdeskTicketHandler

public with sharing class HelpdeskTicketHandler {

public static void beforeInsert(List<Helpdesk\_Ticket\_\_c> newList) {

for (Helpdesk\_Ticket\_\_c t : newList) {

if (t.Priority\_\_c == null) t.Priority\_\_c = 'Medium';

}

}

public static void afterUpdate(List<Helpdesk\_Ticket\_\_c> newList) {

for (Helpdesk\_Ticket\_\_c t : newList) {

if (t.Status\_\_c == 'Resolved' && t.Resolved\_Date\_\_c == null) {

t.Resolved\_Date\_\_c = Date.today();

}

}

}

}

## Trigger Updated:

trigger HelpdeskTicketTrigger on Helpdesk\_Ticket\_\_c (before insert, after update) {

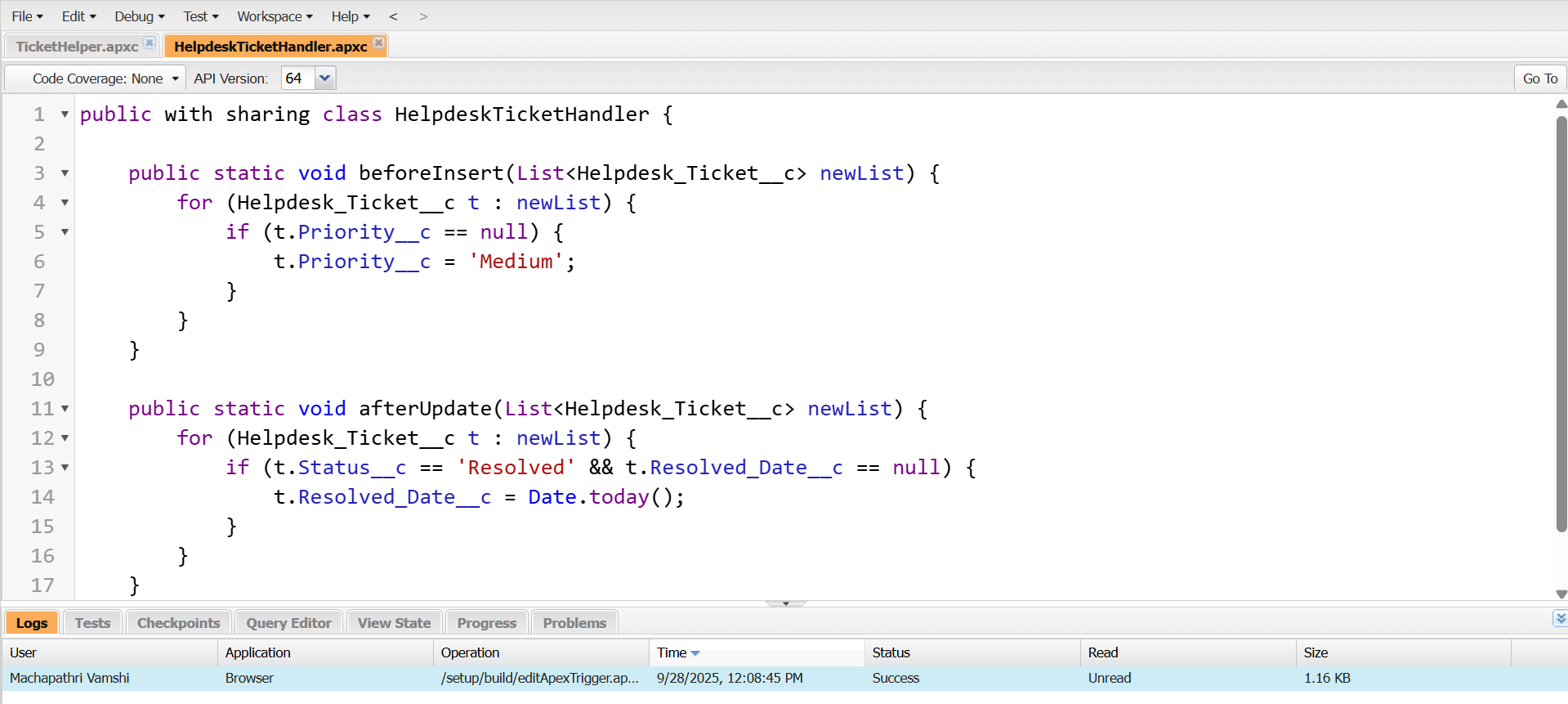
if (Trigger.isBefore && Trigger.isInsert)

HelpdeskTicketHandler.beforeInsert(Trigger.new);

if (Trigger.isAfter && Trigger.isUpdate)

HelpdeskTicketHandler.afterUpdate(Trigger.new);

}



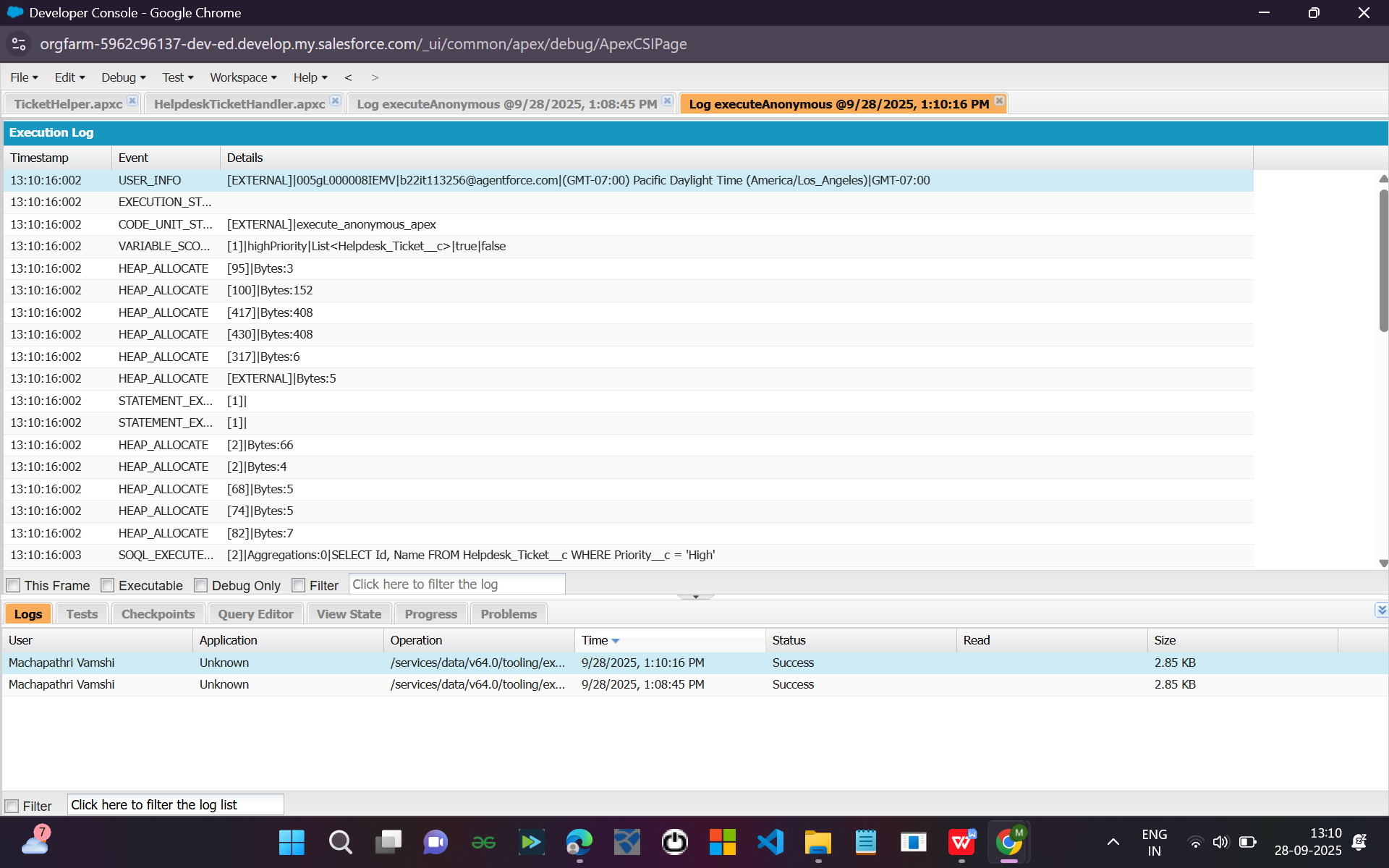
## Step 4 — SOQL & SOSL Queries

List<Helpdesk\_Ticket\_\_c> highPriority = [SELECT Id, Name FROM Helpdesk\_Ticket\_\_c WHERE Priority\_\_c = 'High'];

System.debug('High priority tickets: ' + highPriority);

List<List<SObject>> searchList = [FIND 'Network\*' IN ALL FIELDS RETURNING Helpdesk\_Ticket\_\_c(Name, Category\_\_c)];

System.debug(searchList);



## Step 5 — Collections (List, Set, Map)

List<String> ticketNames = new List<String>{'HT-001', 'HT-002'};

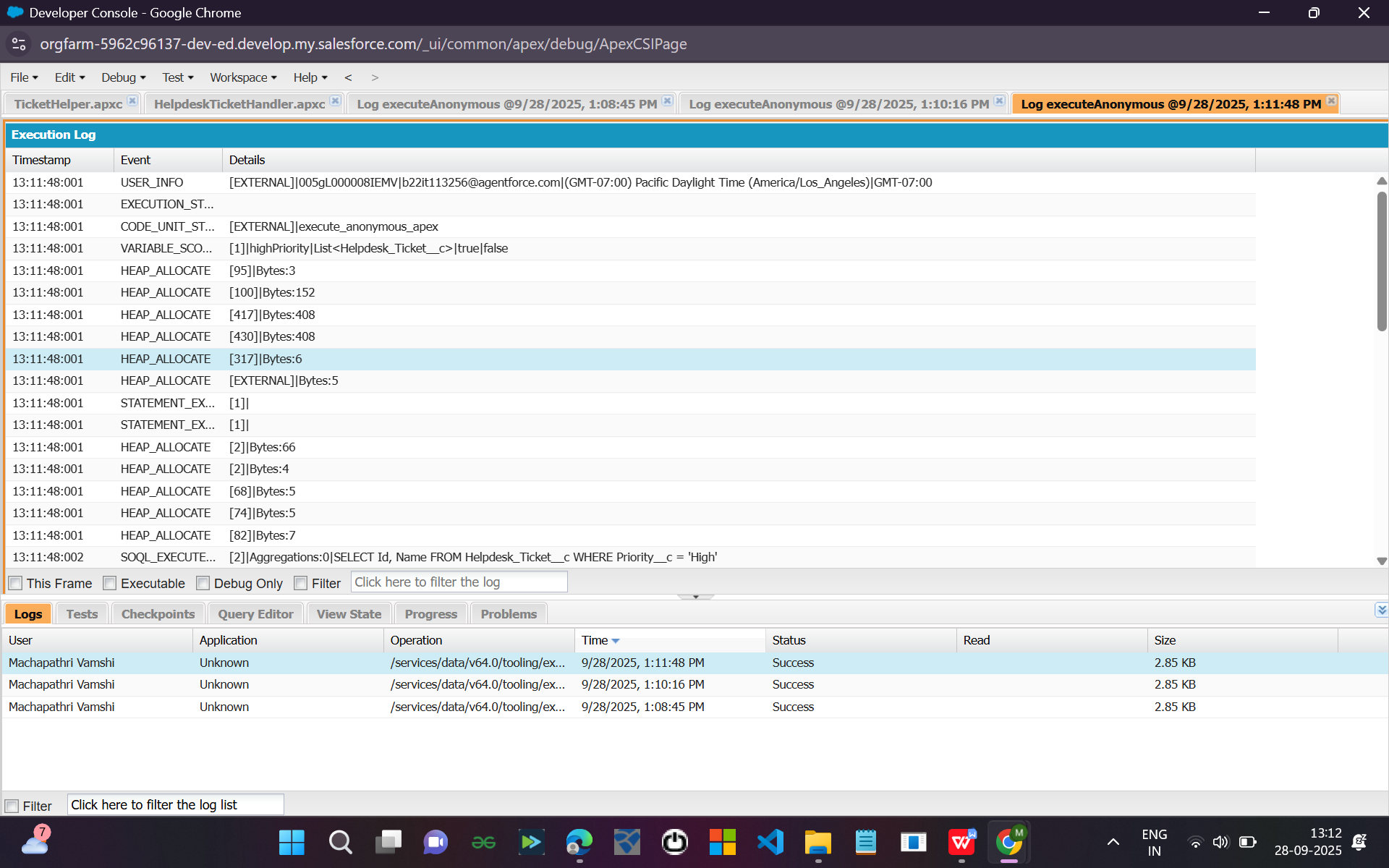
Set<String> categories = new Set<String>{'Network', 'Access'};

Map<Id, Helpdesk\_Ticket\_\_c> ticketMap = new Map<Id, Helpdesk\_Ticket\_\_c>([SELECT Id, Name FROM Helpdesk\_Ticket\_\_c]);

System.debug(ticketNames);

System.debug(categories);

System.debug(ticketMap.keySet());



## Step 6 — Control Statements (If, For, While)

for (Helpdesk\_Ticket\_\_c t : [SELECT Status\_\_c FROM Helpdesk\_Ticket\_\_c]) {

if (t.Status\_\_c == 'Resolved') {

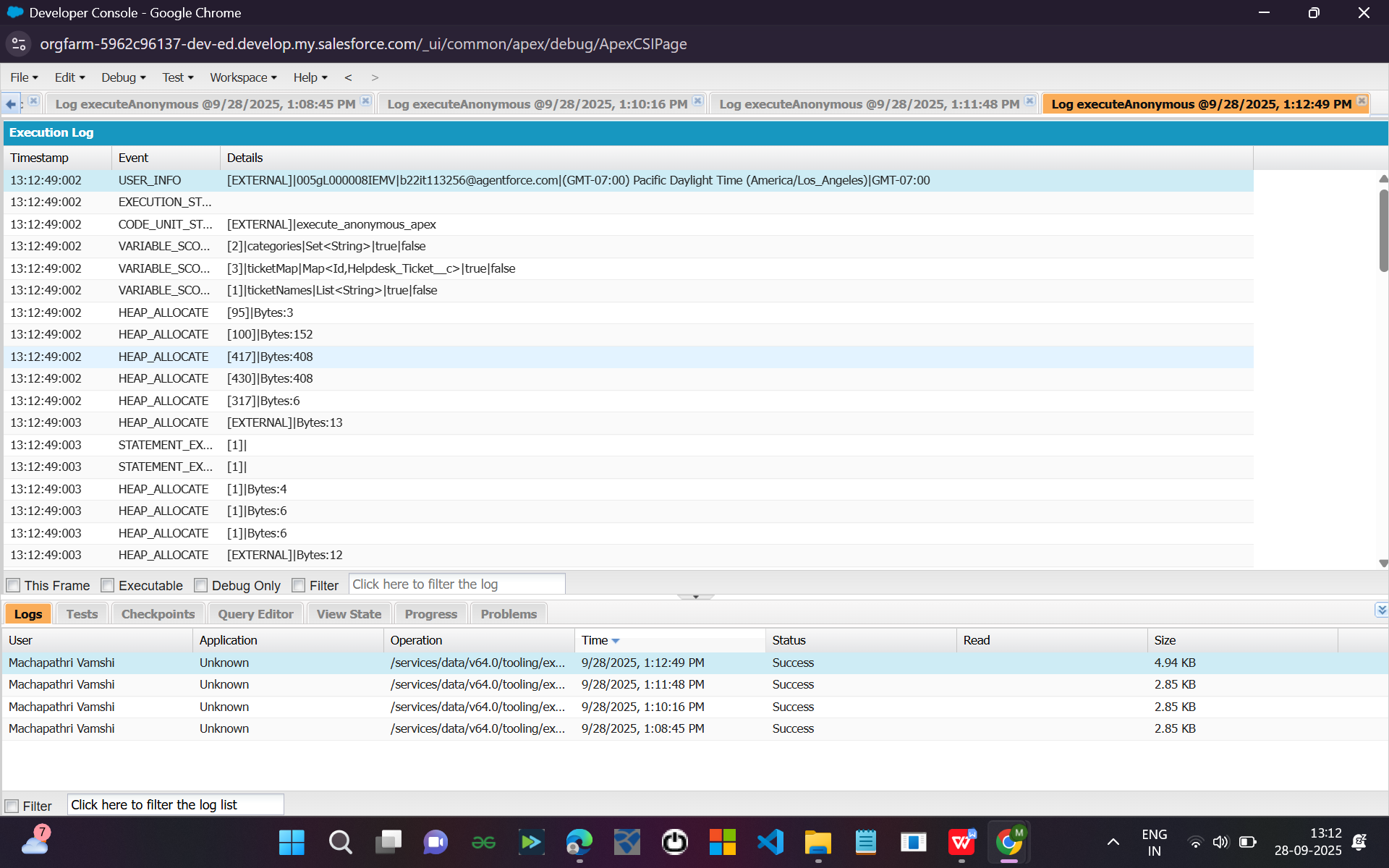
System.debug('Resolved Ticket Found!');

} else {

System.debug('Pending Ticket');

}

}



## Step 7 — Future Methods

public with sharing class TicketAsyncHelper {

@future

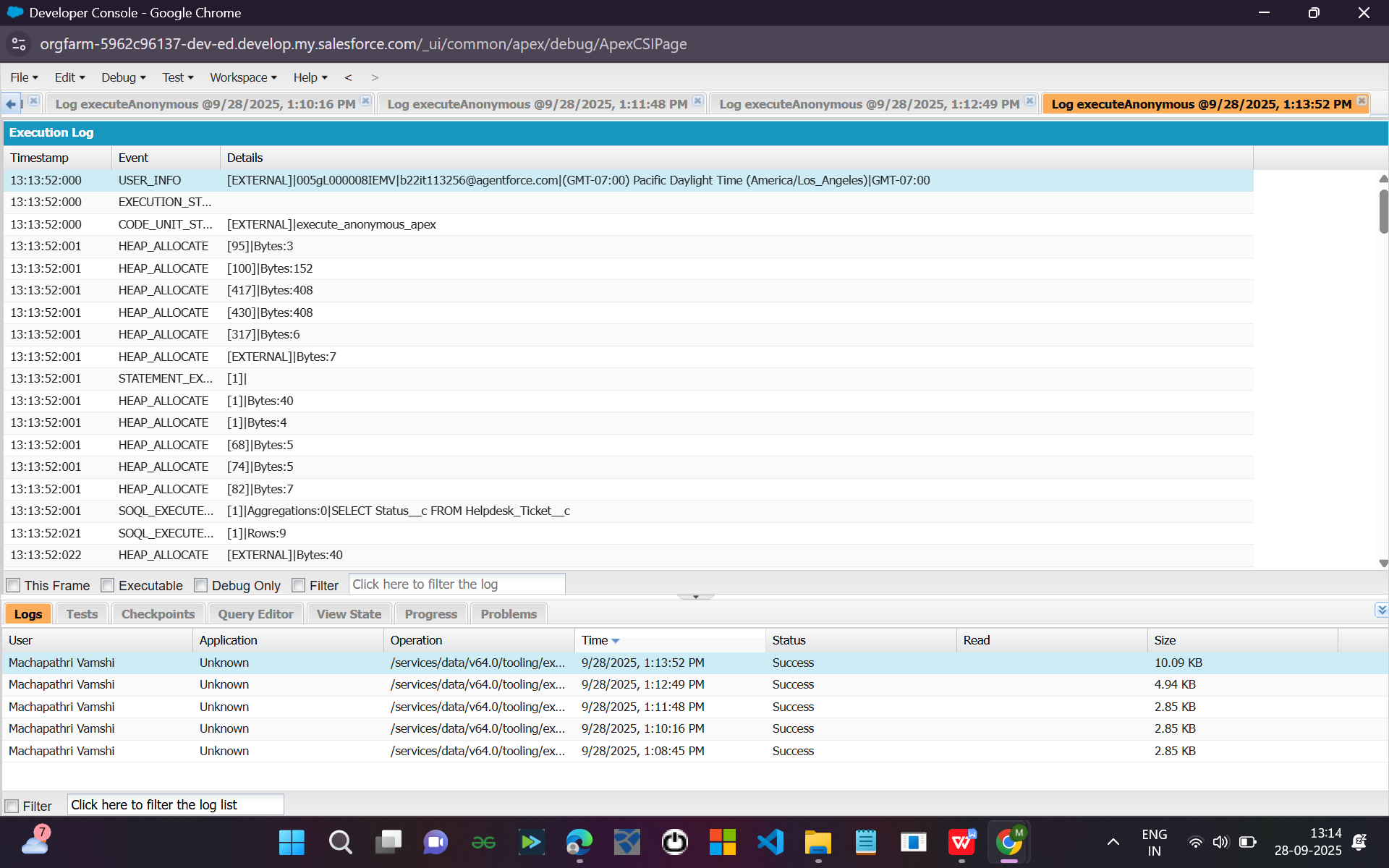
public static void notifyTeam(String message) {

System.debug('Async Notification Sent: ' + message);

}

}

TicketAsyncHelper.notifyTeam('New Ticket Created');



## Step 8 — Batch Apex

public class TicketBatchProcessor implements Database.Batchable<SObject> {

public Database.QueryLocator start(Database.BatchableContext bc) {

return Database.getQueryLocator('SELECT Id FROM Helpdesk\_Ticket\_\_c WHERE Status\_\_c = 'Resolved'');

}

public void execute(Database.BatchableContext bc, List<Helpdesk\_Ticket\_\_c> scope) {

for (Helpdesk\_Ticket\_\_c t : scope) {

System.debug('Processing Ticket: ' + t.Id);

}

}

public void finish(Database.BatchableContext bc) {

System.debug('Batch Completed');

}

}

Database.executeBatch(new TicketBatchProcessor());

## Step 9 — Queueable Apex

public class TicketQueueJob implements Queueable {

public void execute(QueueableContext qc) {

System.debug('Queue Job Executed');

}

}

ID jobId = System.enqueueJob(new TicketQueueJob());

## Step 10 — Scheduled Apex

public class TicketScheduler implements Schedulable {

public void execute(SchedulableContext sc) {

System.debug('Daily Ticket Job Executed');

}

}

Schedule via Setup → Apex Classes → Schedule Apex

## Step 11 — Exception Handling

try {

List<Helpdesk\_Ticket\_\_c> t = [SELECT Id FROM Helpdesk\_Ticket\_\_c WHERE Name = 'Invalid'];

} catch (Exception e) {

System.debug('Error: ' + e.getMessage());

}

## Step 12 — Test Class

@isTest

public class TicketHelperTest {

@isTest

static void testGreetUser() {

String result = TicketHelper.greetUser('Vamshi');

System.assertEquals('Welcome to Helpdesk, Vamshi!', result);

}

@isTest

static void testTriggerInsert() {

Helpdesk\_Ticket\_\_c t = new Helpdesk\_Ticket\_\_c(Category\_\_c='Network', Description\_\_c='Testing trigger');

insert t;

System.assertNotEquals(null, t.Priority\_\_c);

}

}