

# CRM Project – HR Recruitment Process

## Phase 1: Problem Understanding & Industry Analysis

### Goal:

Identify why organizations require a Salesforce-based HR Recruitment CRM and analyze the recruitment challenges in modern enterprises.

### Problem Statement:

Organizations face delays in filling critical job positions due to fragmented recruitment processes, reliance on emails and spreadsheets, and poor visibility into candidate pipelines. These manual methods increase the risk of miscommunication, missed follow-ups, and inconsistent candidate evaluations. High attrition rates or rapid business expansion amplify these challenges, leading to prolonged vacancies, higher recruitment costs, and reduced productivity.

### Solution:

A Salesforce-based HR Recruitment CRM will centralize job requisitions, candidate profiles, and interview schedules. It automates workflows such as candidate status tracking, interview reminders, and offer approvals. Dashboards and reports will give HR managers and recruiters real-time insights to optimize hiring cycles, improve candidate experience, and ensure data security.

### Stakeholders:

- ✓ HR Manager → Creates and manages job openings, monitors KPIs, and approves offers.
- ✓ Recruiters → Source candidates, update statuses, and coordinate interviews.
- ✓ Department Heads → Approve job requisitions and participate in interview panels.
- ✓ IT Admin → Maintains CRM configuration, permissions, and integrations.
- ✓ Candidates → Apply for positions and track their application status.

### Business Process Flow:

Job Request Raised → HR Creates Job Opening → Recruiters Source Candidates  
(LinkedIn, Referrals, Portals) → Screening and Shortlisting → Interview Scheduling &  
Feedback → Department Head Approval → Offer Generation & Communication →

Candidate Onboarding → Job Opening Closed.

**KPIs:**

- Time-to-Hire (average days to fill a role)
- Offer Acceptance Rate
- Candidate Conversion Rate (screened to hired)
- Recruiter Productivity (applications processed per recruiter)
- Employee Retention Rate post-hire

**Requirement Gathering Highlights:**

- Functional Requirements: Job creation, candidate management, interview scheduling, approvals, and dashboards.
- Non-Functional Requirements: Secure access, scalability for high-volume hiring, and mobile-friendly UI.
- Pain Points Identified: Lack of automated notifications, inefficient approval workflows, and scattered candidate data.

**Industry-Specific Use Case Analysis:**

- Benchmarked against popular ATS tools (e.g., Workday, LinkedIn Recruiter).
- Identified trends like AI-powered resume screening and automated reminders.
- Ensured compliance with data protection regulations (e.g., GDPR for candidate data privacy).

**AppExchange Exploration:**

- Reviewed Salesforce AppExchange for HR solutions such as Applicant Tracking Systems and Resume Parsers.
- Found potential integrations like LinkedIn Connector for Salesforce and resume parsing utilities.
- Selected components that could complement our custom recruitment workflow.

## Phase 2: Org Setup & Configuration

### Goal:

Prepare the Salesforce org for implementing the HR Recruitment Process CRM by configuring company settings, user roles, permissions, and foundational security measures.

### Salesforce Edition:

- Developer Edition Org used for development and testing.
- Includes custom objects, Flows, Apex, Approval Processes, and Lightning components.
- Sandbox is used for testing automation and deployment before production.

### Company Profile Setup:

- Navigated to Setup → Company Information.
- Updated organization name to “HR Recruitment CRM”.
- Currency: INR, Locale: India, Time zone: IST.
- Ensured reports and currency fields reflect correct region settings.

The screenshot shows the Salesforce Setup interface with the 'Company Information' page open. The left sidebar shows 'Company Settings' with 'Company Information' selected. The main content area displays the 'Organization Detail' section for 'HR Recruitment CRM'. Key details include:

- Organization Name: HR Recruitment CRM
- Primary Contact: OrgFarm EPIC
- Division: United States
- Fiscal Year Starts In: January
- Default Locale: English (India)
- Default Language: English
- Default Time Zone: (GMT+05:30) India Standard Time (Asia/Kolkata)
- Currency Locale: Hindi (India) - INR
- Used Data Space: 362 KB (7%)
- Used File Space: 17 KB (0%)
- API Requests, Last 24 Hours: 37 (15,000 max)
- Streaming API Events, Last 24 Hours: 0 (10,000 max)
- Restricted Logins, Current Month: 0 (0 max)
- Salesforce.com Organization ID: 00DgL000007exVB
- Organization Edition: Developer Edition
- Instance: CAN98

At the bottom, it shows 'Created By: OrgFarm EPIC 7/21/2025, 3:33 PM' and 'Modified By: Tammisetti Venkatarao 9/19/2025, 11:15 AM'. A 'User Licenses' table is also visible at the bottom of the page.

## Fiscal Year Settings & Business Hours:

- Fiscal Year: Standard, starting from January.
- Defined Business Hours: Monday–Friday, 9 AM–6 PM.
- Added Key Holidays: Republic Day, Diwali, Independence Day.

## User Setup & Licenses:

- Created test users for role-based testing:
  - Admin User – Full system access.
  - HR Manager – Approve offers and manage openings.
  - Recruiter – Manage candidate records and schedule interviews.
  - Department Head – View job openings and approve requisitions.
- Assigned Standard User licenses and mapped them to roles.

Action	Full Name	Alias	Username	Role	Active	Profile
<input type="checkbox"/>	Admin User	admin	admin2908@mail.com		<input checked="" type="checkbox"/>	Standard Platform User
<input type="checkbox"/>	Chatter Expert	chatter	chatter.359e00000007exbuav.c631c0vh50@chatter.salesforce.com		<input checked="" type="checkbox"/>	Chatter Free User
<input type="checkbox"/>	Department Head	dehead	deheadm2908@mail.com		<input checked="" type="checkbox"/>	System Administrator
<input type="checkbox"/>	EPIC_OrgAdmin	GEPE	eoic.56d98f44c28d@orgadmin.salesforce.com		<input checked="" type="checkbox"/>	Standard Platform User
<input type="checkbox"/>	HR Manager	hr.m	hr2908@mail.com		<input checked="" type="checkbox"/>	Standard Platform User
<input type="checkbox"/>	Recruiter	rscr	recruiter2908@mail.com		<input checked="" type="checkbox"/>	Analytics_Cloud_Integration_User
<input type="checkbox"/>	User Integration	integ	integration000400000007exbuav.com		<input checked="" type="checkbox"/>	Analytics_Cloud_Security_User
<input type="checkbox"/>	User Security	ses	inashishsecurity.020000000007exbuav.com		<input checked="" type="checkbox"/>	System Administrator
<input type="checkbox"/>	Venkatarao_Tammineni	tha	thammisettyenkka53743@mailforce.com		<input checked="" type="checkbox"/>	Chatter Free User

## Profiles:

- Recruiter Profile → CRUD on Candidate & Interview objects, Read-only on Job Opening.
- HR Manager Profile → Full access on Job Openings, Candidates, and Interviews.
- Department Head Profile → Read/Edit on Job Openings and Interviews.
- Admin Profile → Full system access.

The screenshot shows the Salesforce Setup interface under the Profiles section. A search bar at the top finds 'prof'. The main content area displays the 'HR Manager' profile. It includes a summary of permissions like Apex Class Access, Visualforce Page Access, External Data Source Access, Named Credential Access, External Credential Principal Access, and various custom settings. Below this is a 'Profile Detail' section with fields for Name (HR Manager), User License (Salesforce), Description, and Created By (Tammisetti Venkatarao). The 'Page Layouts' section lists standard object layouts for Global, Email Application, Home Page Layout, Account, and Alternative Payment Method, along with their respective location group assignments.

This screenshot shows the 'Recruiter Profile' details in the Salesforce Setup interface. Similar to the HR Manager profile, it lists various permissions and the 'Recruiter Profile' itself. The 'Page Layouts' section shows that the Global layout is assigned to the 'Not Assigned' location group, while other object layouts like Home Page Default, Account Layout, and Alternative Payment Method Layout are assigned to the 'Macro' location group.

## Roles & Role Hierarchy:

- Admin → HR Manager → Recruiter / Department Head.
- Ensures managers can view and approve records owned by subordinates.

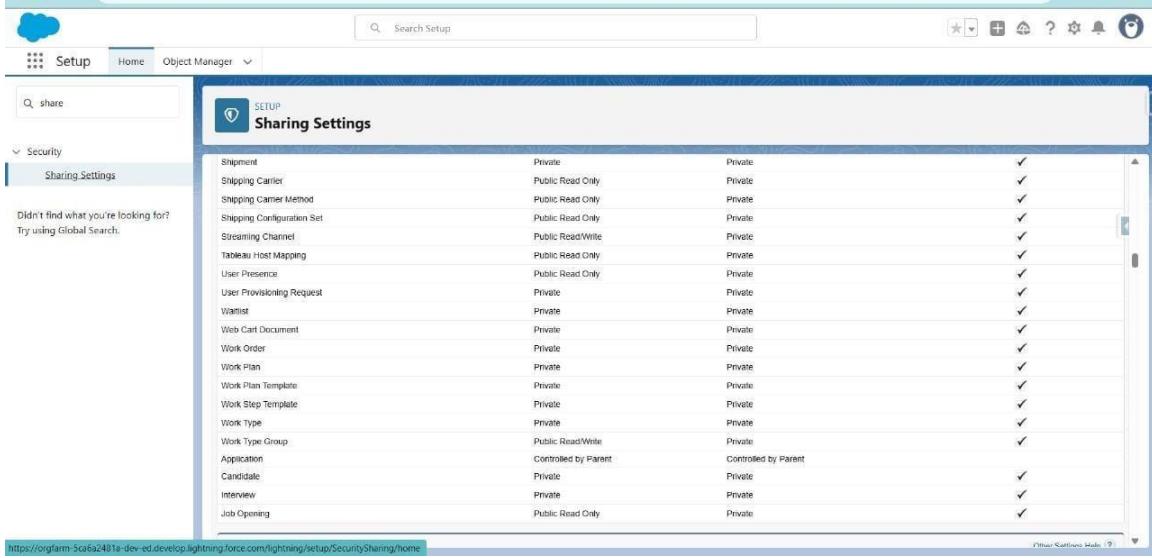
## Permission Sets:

- Created a “Candidate Data Access” permission set to grant temporary or additional permissions.
- Assigned permission sets for testing advanced access scenarios.

## OWD (Organization-Wide Defaults) & Sharing Rules:

- Sharing Settings:
  - Job Openings = Private.
  - Candidates = Private.

- Interviews = Controlled by Parent (Candidate).
- Sharing Rules:
  - Share Candidate records owned by Recruiters with HR Managers (Read/Write).
  - Share Job Openings with Department Heads for visibility.



The screenshot shows the Salesforce Sharing Settings page. The left sidebar has a 'Sharing Settings' section selected under 'Security'. The main area displays a table of sharing rules for various objects. The columns are Object, Default sharing, and Specific sharing. A 'Controlled by Parent' row is present for Application and Interview objects.

Object	Default sharing	Specific sharing
Shipment	Private	Private
Shipping Carrier	Public Read Only	Private
Shipping Carrier Method	Public Read Only	Private
Shipping Configuration Set	Public Read Only	Private
Streaming Channel	Public Read/Write	Private
Tableau Host Mapping	Public Read Only	Private
User Presence	Public Read Only	Private
User Provisioning Request	Private	Private
Waitlist	Private	Private
Web Cart Document	Private	Private
Work Order	Private	Private
Work Plan	Private	Private
Work Plan Template	Private	Private
Work Step Template	Private	Private
Work Type	Private	Private
Work Type Group	Public Read/Write	Private
Application	Controlled by Parent	Controlled by Parent
Candidate	Private	Private
Interview	Private	Private
Job Opening	Public Read Only	Private

## Login Access Policies:

- Default policies retained (Admins can log in as users to troubleshoot issues).
- Enabled IP restrictions for added security.

## Sandbox Usage & Deployment Basics:

- Created a Developer Sandbox for testing automation and approval flows.
- Deployment Basics:
  - Used Change Sets to migrate configurations to production.
  - Validated deployments before applying to the live environment.

**Deliverables:**

- Configured Salesforce org with company profile, business hours, and fiscal year.
- Defined user roles, profiles, and permission sets.
- Established OWD, sharing rules, and login policies for secure data access.
- Sandbox created and deployment steps documented.

## Phase 3: Data Modeling & Relationships

### Goal:

Build a robust data model that represents jobs, candidates, applications, interviews, and related entities. Create relationships (master-detail, lookup, junction), record types, page layouts, compact layouts, and ensure the model supports reporting and automation.

### 1. Create Custom Objects (core objects)

**Purpose:** Create objects to store recruitment data.

#### Objects to create (recommended):

- **Job\_Opening\_c** — stores job requisitions.
- **Candidate\_c** — stores candidate profile & contact info.
- **Application\_c** — *junction object* between Candidate and Job Opening (one candidate can apply to many jobs, one job can have many candidates).
- **Interview\_c** — interview records linked to Application (or Candidate).
- **Resume\_c** (optional) — file/reference to candidate resume or parsed data.

#### Steps (example for Job\_Opening\_c):

1. Click **Setup** → enter **Object Manager** in Quick Find → **Object Manager**.
2. Click **Create** → **Custom Object**.
3. For **Label** enter: Job Opening  
**Plural Label:** Job Openings  
**Object Name (API):** Job\_Opening\_c  
**Record Name:** Job Opening Name (Auto-Number or Text)
4. Check **Allow Reports**, **Allow Activities**, and **Track Field History** (as needed).
5. Click **Save**.

Repeat the same for **Candidate\_c**, **Application\_c**, **Interview\_c**, etc.

The screenshot shows the Salesforce Object Manager page. At the top, there's a navigation bar with 'Setup' (selected), 'Home', and 'Object Manager'. A search bar says 'Search Setup' with a magnifying glass icon. To the right are various icons for filtering, sorting, and creating new objects. Below the navigation is a header for 'Object Manager' with a blue ribbon icon, a search bar ('job'), a 'Schema Builder' button, and a 'Create' button. A message indicates '1 Items, Sorted by Label'. A table lists one item:

LABEL	API NAME	TYPE	DESCRIPTION	LAST MODIFIED	DEPLOYED
Job Opening	Job_Opening_c	Custom Object		9/20/2025	✓

## 2. Add Fields to Objects (detailed examples)

**General:** For each object, create fields with clear API names and types.

### **Job\_Opening\_c — suggested fields**

- Job\_Code\_\_c — Auto Number JOB-{0000}
- Job\_Title\_\_c — Text (255)
- Department\_\_c — Picklist (e.g., Sales, Engineering, HR, Finance)
- Hiring\_Manager\_\_c — Lookup(User)
- Positions\_Open\_\_c — Number (Integer)
- Status\_\_c — Picklist (Draft, Open, Closed, On Hold)
- Location\_\_c — Text or Picklist
- Salary\_Range\_Low\_\_c & Salary\_Range\_High\_\_c — Currency

### **Candidate\_\_c — suggested fields**

- FirstName\_\_c, LastName\_\_c — Text (or use Contact standard object)
- Email\_\_c — Email (Required)
- Phone\_\_c — Phone
- Resume\_Link\_\_c — URL or File (use Files)
- Source\_\_c — Picklist (LinkedIn, Referral, Job Portal, Walk-in)
- Current\_Status\_\_c — Picklist (Applied, Screened, Interview, Offered, Hired, Rejected)
- Current\_Score\_\_c — Number (3,1) — optional scoring field.

### **Application\_\_c (junction between Candidate & Job)**

- Candidate\_\_c — Master-Detail → Candidate\_\_c
- Job\_Opening\_\_c — Master-Detail → Job\_Opening\_\_c
- Application\_Date\_\_c — Date
- Stage\_\_c — Picklist (Applied, Screening, Interviewing, Offer, Hired, Rejected)
- Resume\_Attached\_\_c — Checkbox

### **Interview\_\_c**

- Application\_\_c — Lookup(Application\_\_c) or Master-Detail to Application
- Interview\_Date\_\_c — Date/Time
- Panel\_\_c — Text / Related Users (could be a multi-select picklist of panel members or create Interview\_Panel\_\_c child records)
- Feedback\_\_c — Long Text Area
- Interview\_Result\_\_c — Picklist (Pass, Fail, Hold)

### **Steps to add a field (example Email on Candidate\_\_c):**

1. Setup → Object Manager → open **Candidate** → **Fields & Relationships** → **New**.
2. Choose **Email** type → Next.

Field Label: Email → Field Name: Email\_\_c → Set **Required** if desired → Next → Set field-level security → Next → Add to Page Layouts → Save.

The screenshot shows the Salesforce Setup interface with the following details:

- Setup** icon in the top left.
- Search bar: Search Setup.
- Top navigation: Setup, Home, Object Manager.
- Breadcrumbs: SETUP > OBJECT MANAGER.
- Section title: Interview.
- Left sidebar (under Fields & Relationships):
  - Details
  - Fields & Relationships** (selected)
  - Page Layouts
  - Lightning Record Pages
  - Buttons, Links, and Actions
  - Compact Layouts
  - Field Sets
  - Object Limits
  - Record Types
  - Related Lookup Filters
  - Restriction Rules
  - Scoping Rules
  - Object Access
  - Triggers
  - Flow Triggers
  - Validation Rules
  - Conditional Field Formatting
- Main content area:
 

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Application	Application_c	Lookup(Application)		✓
Created By	CreatedById	Lookup(User)		
Feedback	Feedback_c	Long Text Area(32768)		
Interview Date	Interview_Date_c	Date/Time		
Interview Name	Name	Text(80)		✓
Interview Result	Interview_Result_c	Picklist		
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		✓
Panel	Panel_c	Text(255)		

### 3. Create Relationships (lookup, master-detail, junction)

#### When to use which:

- **Master-Detail:** Use when child record should inherit parent sharing and be deleted when parent deleted (e.g., Application as master-detail both Candidate & Job).
- **Lookup:** Use when child should be independent (e.g., Interview may be a lookup to Application if you want interviews to survive application deletion).
- **Junction Object:** Use Application\_c to model many-to-many between Job and Candidate.

#### Step to create a Master-Detail (Application → Candidate):

1. Setup → Object Manager → **Application** → **Fields & Relationships** → New.
2. Choose **Master-Detail Relationship** → Next.
3. Related To: **Candidate** → Next.
4. Field Label: Candidate → Field Name: Candidate\_c → Next → Set sharing and behavior → Save.
5. Repeat for **Job\_Opening** master-detail.

**Result:** Application now shows as related list on both Candidate and Job Opening pages.

The screenshot shows the Salesforce Object Manager interface. On the left, a sidebar lists various setup options under 'Fields & Relationships'. The main content area is titled 'Application Custom Field Candidate' and displays the 'Custom Field Definition Detail' for the 'Candidate' field. Key details shown include:

- Field Information:** Field Label: Candidate, Field Name: Candidate, API Name: Candidate\_\_c, Data Type: Master-Detail, Object Name: Application.
- Master-Detail Options:** Related To: Candidate, Child Relationship Name: Applications, Shared Setting: Read/Write.
- Validation Rules:** No validation rules defined.

At the bottom right of the main area, there is a note: "Always show me ▾ more records per related list".

#### 4. Create Record Types & Picklist Variants

**Use case:** Show different fields or page layouts for **Internal Hiring vs External Hiring or Referral vs Open Application**.

**Steps:**

1. Setup → Object Manager → **Job Opening** → **Record Types** → New.
2. Select existing profile defaults to clone.
3. Enter Record Type Label: Internal and API name: Internal.
4. Repeat to create External.
5. For each Record Type, assign Page Layouts and set picklist value availability.

**Tip:** Use record types when fields, picklist values, or required fields differ by hiring type.

The screenshot shows the Salesforce Object Manager interface for a custom object named 'Job Opening'. On the left, a sidebar lists various setup options like Details, Fields & Relationships, Page Layouts, etc. The 'Record Types' option is selected. The main content area displays the 'External' record type configuration. It includes fields for Record Type Label (set to 'External'), Record Type Name (set to 'External'), and Namespace Prefix. The status is set to 'Active'. Below this, a table shows 'Picklists Available for Editing' with two entries: 'Department' and 'Status', both modified on 9/20/2025, 8:44 AM.

## 5. Page Layouts & Compact Layouts

**Page Layouts** control detail page appearance; **Compact Layouts** control highlights on record cards & mobile.

### Create/Edit Page Layout:

1. Setup → Object Manager → Job Opening → **Page Layouts** → New (or edit existing).
2. Drag/Drop fields, Related Lists (Applications, Interviews), Buttons.
3. Save.
4. Assign page layout to profiles and record types (Page Layout Assignment).

### Create Compact Layout:

1. Setup → Object Manager → Job Opening → **Compact Layouts** → New.
2. Choose fields shown in the highlights panel (Job Title, Status, Hiring Manager).

Save → Set as the org default or assign by record type.

SETUP > OBJECT MANAGER

## Job Opening

**Page Layouts**

- Lightning Record Pages
- Buttons, Links, and Actions
- Compact Layouts
- Field Sets
- Object Limits
- Record Types
- Related Lookup Filters
- Restriction Rules
- Scoping Rules
- Object Access
- Triggers
- Flow Triggers
- Validation Rules
- Conditional Field Formatting

**Fields**

Section	Hiring Manager	Last Modified By	Record Type
Blank Space	Job Code	Location	Salary Range High
Created By	Job Opening Name	Owner	Salary Range Low
Department	Job Title	Positions Open	Status

**Job Opening Sample**

Customize the highlights panel for this page layout...

**Quick Actions in the Salesforce Classic Publisher**

Actions in this section are currently inherited from the global publisher layout. You can [override the global publisher layout](#) to set a customized list of actions for the publisher on pages that use this layout.

**Salesforce Mobile and Lightning Experience Actions**

Actions in this section are predefined by Salesforce. You can [override the predefined actions](#) to set a customized list of actions on Lightning Experience and mobile app pages that use this layout. If you customize the actions in the Quick Actions in the Salesforce Classic Publisher section, and have saved the layout, then this section inherits that set of actions by default when you click to override.

SETUP > OBJECT MANAGER

## Job Opening

**Compact Layouts**

- Field Sets
- Object Limits
- Record Types
- Related Lookup Filters
- Restriction Rules
- Scoping Rules
- Object Access
- Triggers
- Flow Triggers
- Validation Rules
- Conditional Field Formatting

**Job Title**

[Back to Job Opening](#)

**Compact Layout Detail**

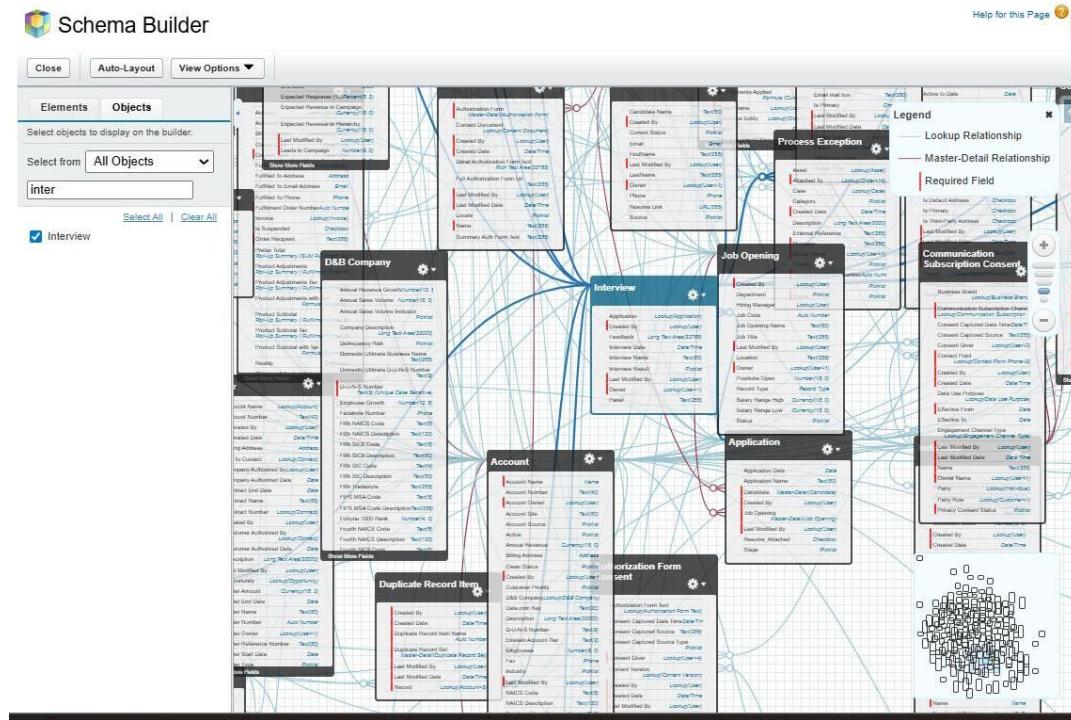
Label	Job Title	Object Name	
API Name	Job_Title	Job_Opening	
Included Fields	Job Title Status Hiring Manager		
Created By	Tammisetti Venkatarao, 9/20/2025, 8:53 AM	Modified By	Tammisetti Venkatarao, 9/20/2025, 8:53 AM

## 6. Schema Builder (visualize & adjust)

Steps:

1. Setup → enter **Schema Builder** in Quick Find → **Schema Builder**.
2. From the left pane, check the objects you created (Job\_Opening, Candidate, Application, Interview).
3. Drag objects onto canvas to view relationships.
4. You can also create fields or relationships from Schema Builder (click on object → Add field).

Use it to verify your ERD and to export/interpret model when writing documentation.



## 7. Create Tabs & Add to App

Steps:

1. Setup → Quick Find → **Tabs** → New (Custom Object Tabs).
2. Select **Job Opening** → Choose tab style → Next → Add to desired Apps (Recruitment App) → Save.
3. Repeat for Candidate and Application.

## 8. Search Layouts & List Views

- Setup → Object Manager → Candidate → **Search Layouts for Salesforce Classic / Search Layouts for Lightning Experience** → Configure fields that show in search results.

Create List Views for hiring managers: e.g., Open Applications, Interviews Today, Offers Pending.

## 9. Validation Rules (examples)

### Example 1 — Require Email or Phone for Candidate:

- Setup → Object Manager → Candidate → **Validation Rules** → New.
- Rule Name: `Require_Email_or_Phone`
- Formula:

`AND( ISBLANK( Email__c ), ISBLANK( Phone__c ) )`

Error Message: Either Email or Phone must be provided. → Error Location: Top of Page  
→ Save.

The screenshot shows the Salesforce Setup interface with the following details:

- Header:** Search Setup, Setup, Home, Object Manager.
- Breadcrumbs:** SETUP > OBJECT MANAGER Candidate
- Left sidebar:** Details, Fields & Relationships, Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Limits, Record Types, Related Lookup Filters, Restriction Rules, Scoping Rules, Object Access, Triggers, Flow Triggers, Validation Rules (selected), Conditional Field Formatting.
- Table:** Validation Rules (1 Items, Sorted by Rule Name)

RULE NAME	ERROR LOCATION	ERROR MESSAGE	ACTIVE	MODIFIED BY
Require_Email	Top of Page	Either Email or Phone must be provided. → Error Location: Top of Page → Save.	✓	Tammiseti Venkatarao, 9/20/2025, 9:12 AM

## 10. Duplicate Management (Matching & Duplicate Rules)

### Create Matching Rule (Candidate by Email & Phone):

1. Setup → Quick Find: **Matching Rules** → New Rule.
2. Object: Candidate. Matching Criteria: Email (Exact), Phone (Exact).
3. Save → Activate.

## Create Duplicate Rule:

1. Setup → Quick Find: **Duplicate Rules** → New Rule.
2. Object: Candidate. Rule Type: Use the matching rule just created.
3. Action on Create/Edit: Block or Allow and Report (choose Allow and Report during testing).
4. Save → Activate.

The screenshot shows the Salesforce Setup interface with the 'Duplicate' tab selected in the sidebar. The main content area displays the 'Duplicate Rules' page for 'Candidate Duplicate Rule'. Rule 2 is listed with the following details:

Rule Name	Rule 2	Order	1 of 1 [Reorder]
Description	Candidate Enforce sharing rules		
Action On Create	Allow	Operations On Create	<input checked="" type="checkbox"/> Alert <input checked="" type="checkbox"/> Report
Action On Edit	Allow	Operations On Edit	<input checked="" type="checkbox"/> Alert <input checked="" type="checkbox"/> Report
Alert Text	Use one of these records?		
Active	<input checked="" type="checkbox"/>		
Matching Rule	Rule 1 Mapped	Matching Criteria	(Candidate: Email EXACT MatchBlank = FALSE) AND (Candidate: Phone EXACT MatchBlank = FALSE)
Conditions			
Created By	Tammisetti Venkataram, 9/20/2025, 9:22 AM	Modified By	Tammisetti Venkataram, 9/20/2025, 9:25 AM

## 11. Import Sample Data

### Small import (Data Import Wizard):

1. Setup → Quick Find → **Data Import Wizard** → Launch Wizard.
2. Choose object: Candidates (or Accounts/Contacts if using Contact).
3. Upload CSV file with mapped headers (FirstName, LastName, Email, Phone, Source).
4. Map CSV columns to Salesforce fields → Start Import.
5. Review import results.

### Large imports (Data Loader):

- Use Data Loader to insert Application records or historical data; map lookup fields using external IDs or Salesforce IDs.

**Tip:** For Master-Detail relationships during import, either import parent records first (Candidate, Job) and use their Salesforce IDs in child records, or use External ID fields to match.

## 12. Security for Objects & Fields

- For each sensitive field (e.g., Salary\_Range), set **Field-Level Security**: Setup → Object Manager → Field → **Set Field-Level Security** — hide for recruiter if needed.
- Use **Permission Sets** to grant special access to certain users.

The screenshot shows the Salesforce Setup interface with the following details:

- Object Manager:** Job Opening
- Custom Field Definition Detail:**
  - Field Label:** Salary Range High
  - Field Name:** Salary\_Range\_High
  - API Name:** Salary\_Range\_High\_c
  - Description:** Help Text
  - Data Owner:** Field Usage
  - Data Sensitivity Level:** Compliance Categorization
  - Created By:** Tammiseti Venkatarao, 9/20/2025, 8:09 AM
  - Modified By:** Tammiseti Venkatarao, 9/20/2025, 8:09 AM
- General Options:**
  - Required:**
  - Default Value:** 0
- Currency Options:**
  - Length:** 18
  - Decimal Places:** 0
- Validation Rules:** No validation rules defined.

### 13. Use a Junction Object for Many-to-Many (Application example)

**Why Application\_c:** A Candidate can apply to multiple Job Openings and each Job Opening can have many Candidates.

- Create Application\_c → two master-detail fields: Candidate\_c and Job\_Opening\_c.
- This makes Applications appear in related lists on both Candidate and Job Opening pages and supports per-application stages.

The screenshot shows the Salesforce Setup interface with the following details:

- Header:** Includes the Salesforce logo, a search bar labeled "Search Setup", and various navigation icons.
- Breadcrumbs:** "SETUP > OBJECT MANAGER".
- Section:** "Application".
- Left Sidebar (Details):**
  - Fields & Relationships (selected)
  - Page Layouts
  - Lightning Record Pages
  - Buttons, Links, and Actions
  - Compact Layouts
  - Field Sets
  - Object Limits
  - Record Types
  - Related Lookup Filters
  - Restriction Rules
  - Scoping Rules
  - Object Access
  - Triggers
  - Flow Triggers
  - Validation Rules
  - Conditional Field Formatting
- Table Header:** "Fields & Relationships" with columns: FIELD LABEL, FIELD NAME, DATA TYPE, CONTROLLING FIELD, and INDEXED.
- Table Data:**

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Application Date	Application_Date_c	Date		▼
Application Name	Name	Text(80)	✓	▼
Candidate	Candidate_c	Master-Detail(Candidate)	✓	▼
Created By	CreatedById	Lookup(User)		
Job Opening	Job_Opening_c	Master-Detail(Job Opening)	✓	▼
Last Modified By	LastModifiedById	Lookup(User)		
Resume_Attached	Resume_Attached_c	Checkbox		▼
Stage	Stage_c	Picklist		▼

## Documenting the Model & Deliverables

### Deliverables for Phase 3:

- ERD (Entity Relationship Diagram) — can export from Schema Builder or draw in Visio/Miro.
- Object & field dictionary (table with Field Label, API Name, Type, Description, Required?).
- Record Type list & page layout assignment matrix (which profile sees which layout).
- Validation rules list with formulas and purpose.
- Duplicate/matching rules configuration.
- Sample CSV files used to import test data and import logs.
- Screenshots of key configurations (Object Manager, schema builder, page layout).

## Phase 4: Process Automation (Admin)

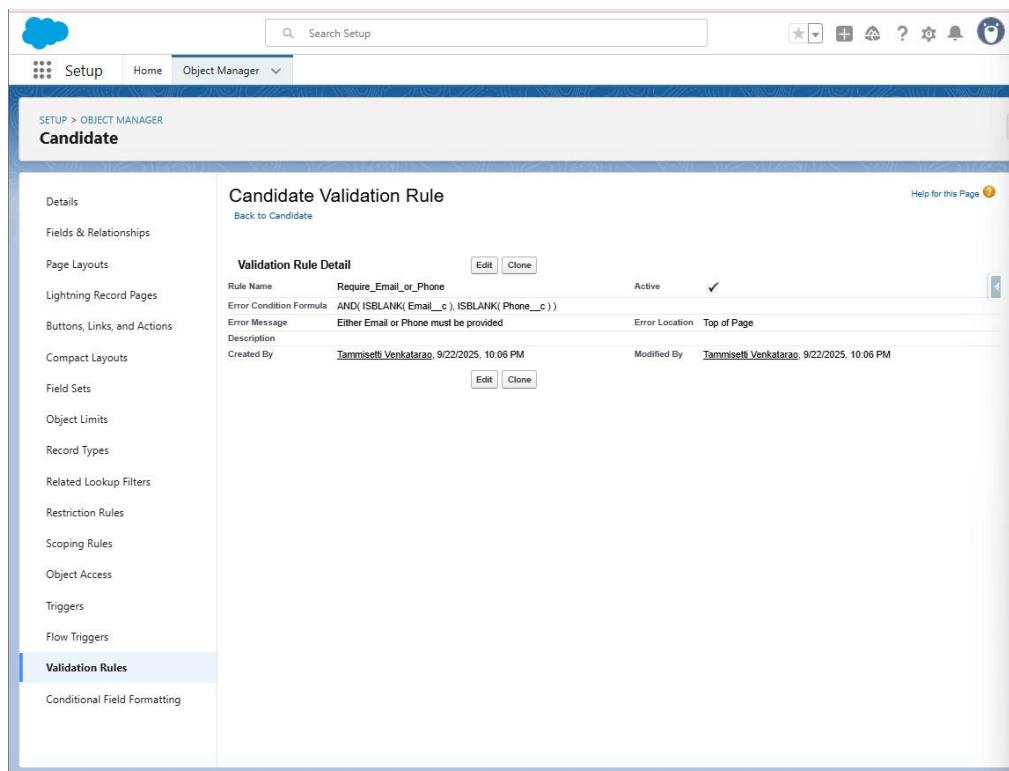
### Goal:

Automate recruitment workflows using declarative Salesforce tools (Validation Rules, Flows, Approval Processes, Email Alerts, Tasks, and Scheduled Automations). Ensure automations are maintainable, tested in Sandbox, and documented.

### 1) Validation Rules (examples)

#### Example A — Require Email OR Phone on Candidate

1. Click Setup → Object Manager → Candidate → Validation Rules → New.
2. Rule Name: Require\_Email\_or\_Phone
3. Error Condition Formula:  
$$\text{AND( ISBLANK( Email\_c ), ISBLANK( Phone\_c ) )}$$
4. Error Message: Either Email or Phone must be provided.
5. Error Location: Top of Page → Save.



## 2) Create Email Templates & Alerts (prepare first)

### Create a Lightning Email Template (Congratulate Hired)

1. App Launcher → **Email Templates** → **New Email Template**.

2. Name: Candidate\_Hired\_Congrats

3. Related Entity Type: **Application** (or Candidate)

4. Subject: Congratulations — Offer Accepted for  
{!Application\_\_c.Job\_Opening\_\_r.Job\_Title\_\_c}

5. Body (example):

Hi {!Application\_\_c.Candidate\_\_r.FirstName},

Congratulations! Your application for  
{!Application\_\_c.Job\_Opening\_\_r.Job\_Title\_\_c} has been successful.

Your expected joining date: {!Application\_\_c.Joining\_Date\_\_c}.

HR Manager: {!User.FirstName} {!User.LastName}

6. Save.

The screenshot shows the Salesforce Email Template page for the 'Candidate Hired Congrats' template. The page includes a header with navigation links like Sales, Home, Opportunities, Leads, Tasks, Files, Accounts, Contacts, and a search bar. Below the header, there's a toolbar with options like Edit, Clone, Delete, and a dropdown menu. The main content area displays the template details:

Details		Related	
<b>Information</b>			
Email Template Name	Candidate Hired Congrats	Related Entity Type	Application
Description	Email sent to candidate when an offer is accepted	Folder	Public Email Templates
Made in Email Template Builder	<input type="checkbox"/>		
<b>Message Content</b>			
Subject	Enhanced Letterhead		
HTML Value	<input type="text"/>		
<b>Additional Information</b>			
Created By	Last Modified By		
Tammisetti Venkatarao, 9/21/2025, 7:06 AM	Tammisetti Venkatarao, 9/21/2025, 7:10 AM		

## Create an Email Alert to use in Flows

1. Setup → **Email Alerts** → New Email Alert.
2. Description: Send Congratulation to Candidate on Hired
3. Object: **Application**
4. Email Template: Candidate\_Hired\_Congrats
5. Recipient Type: Related Contact / Email Field → choose Candidate Email (or related Contact)
6. Save.

The screenshot shows the Salesforce Setup interface. The top navigation bar includes a search bar, a gear icon for Setup, Home, and Object Manager. Below the navigation is a sidebar with links for Process Automation, Workflow Actions, and Email Alerts, with Email Alerts selected. The main content area displays a page titled "Email Alerts" with a sub-section for "Email Alert". The alert is titled "Send Congratulation to Candidate on Hired". The "Email Alert Detail" section shows the following configuration:

- Description: Send Congratulation to Candidate on Hired
- Unique Name: Send\_Congratulation\_to\_Candidate\_on\_Hired
- From Email Address: Current User's email address
- Recipients: [tammisetti Venkatara53@gmail.com](#)
- Additional Emails: [tammisetti Venkatara53@gmail.com](#)
- Created By: [Tammisetti Venkatara53](#), 9/21/2025, 7:13 AM
- Modified By: [Tammisetti Venkatara53](#), 9/21/2025, 7:13 AM

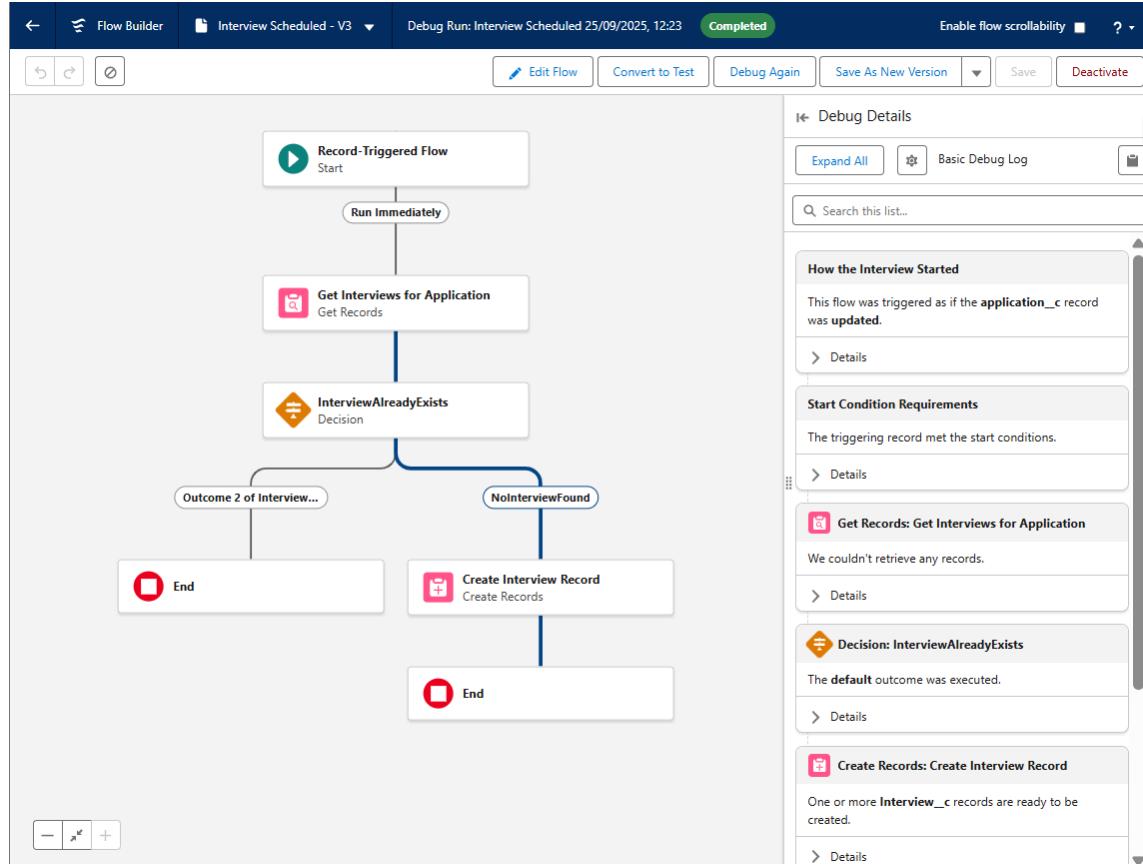
Below the detail section are four sections: "Rules Using This Email Alert", "Approval Processes Using This Email Alert", "Entitlement Processes Using This Email Alert", and "Flows Using This Email Alert". Each section contains a message indicating that no rules, approval processes, entitlement processes, or flows are currently associated with the alert.

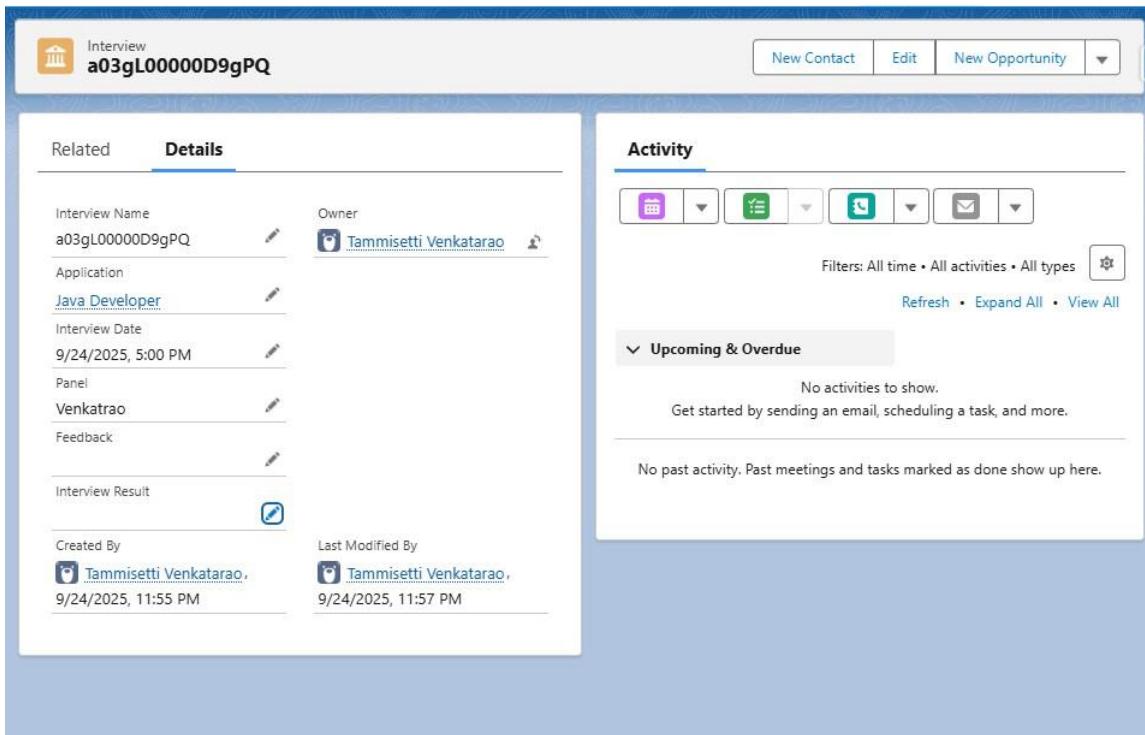
## 3) Record-Triggered Flow: Auto-create Interview when Stage = Interview Scheduled

1. Setup → **Flows** → New Flow → Record-Triggered Flow.
2. Object: **Application\_c**. Trigger: A record is updated.

3. Entry Condition: Stage\_c Equals Interview Scheduled. Optimize for: **Actions and Related Records**.
4. (Optional) Add **Decision**: InterviewAlreadyExists? to prevent duplicates.
5. Add **Create Records**: Object = **Interview\_\_c**. Map fields:
  - o Application\_\_c = \$Record.Id
  - o Interview\_Date\_\_c = \$Record.Interview\_Date\_\_c
  - o Interview\_Result\_\_c = Scheduled
  - o Panel\_\_c = \$Record.Panel\_\_c
6. (Optional) Add **Send Email Alert** or **Send Custom Notification** to notify panel/recruiter.
7. Save → Name: AutoCreateInterview\_On\_InterviewScheduled → **Activate**.

**Test:** In Sandbox, update an Application to Stage = Interview Scheduled; verify Interview appears in related list and notification/email is received.

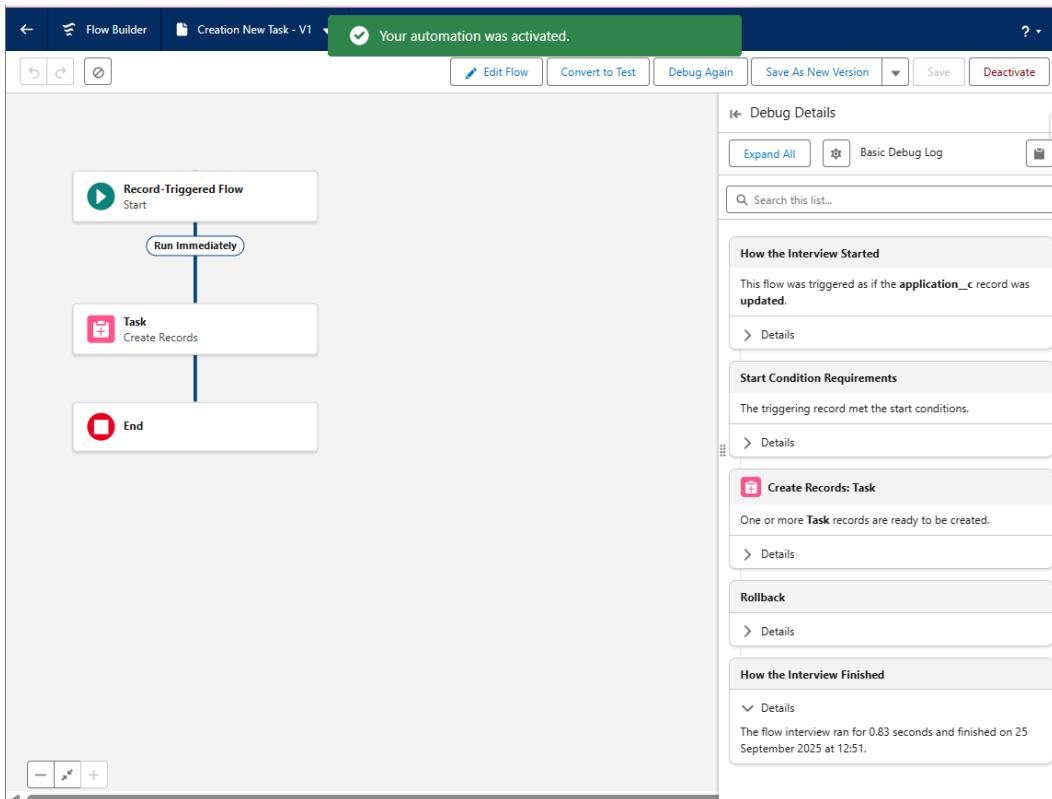




#### 4) Record-Triggered Flow: Auto-create Task for Recruiter on Stage = Screening

1. Setup → Flows → New → Record-Triggered Flow.
2. Object: **Application\_\_c** → Trigger when record is created or updated.
3. Entry Condition: Stage\_\_c Equals Screening.
4. Add **Create Records** → Object: **Task**. Fields:
  - Subject: Screen Candidate - {!\$Record.Candidate\_\_r.Name}
  - WhoId / WhatId: set to Candidate or Application as appropriate
  - OwnerId: \$Record.OwnerId (or specific recruiter)
  - Status: Not Started
  - Priority: Normal
5. Save & Activate.

**Test:** Change Stage to Screening → confirm Task appears under related Tasks.



Sales Home Opportunities Leads Tasks Files Accounts Contacts Campaigns Dashboards Applications More

Application Salesforce

New Contact Edit New Opportunity

Related	Details
Application Name	Salesforce
Candidate	Gopi
Job Opening	Salesforce
Application Date	9/25/2025
Stage	Screening
Resume_Attached	<input checked="" type="checkbox"/>
Created By	Tammiseti Venkatarao, 9/24/2025, 10:07 AM
Last Modified By	Tammiseti Venkatarao, 9/25/2025, 12:22 AM

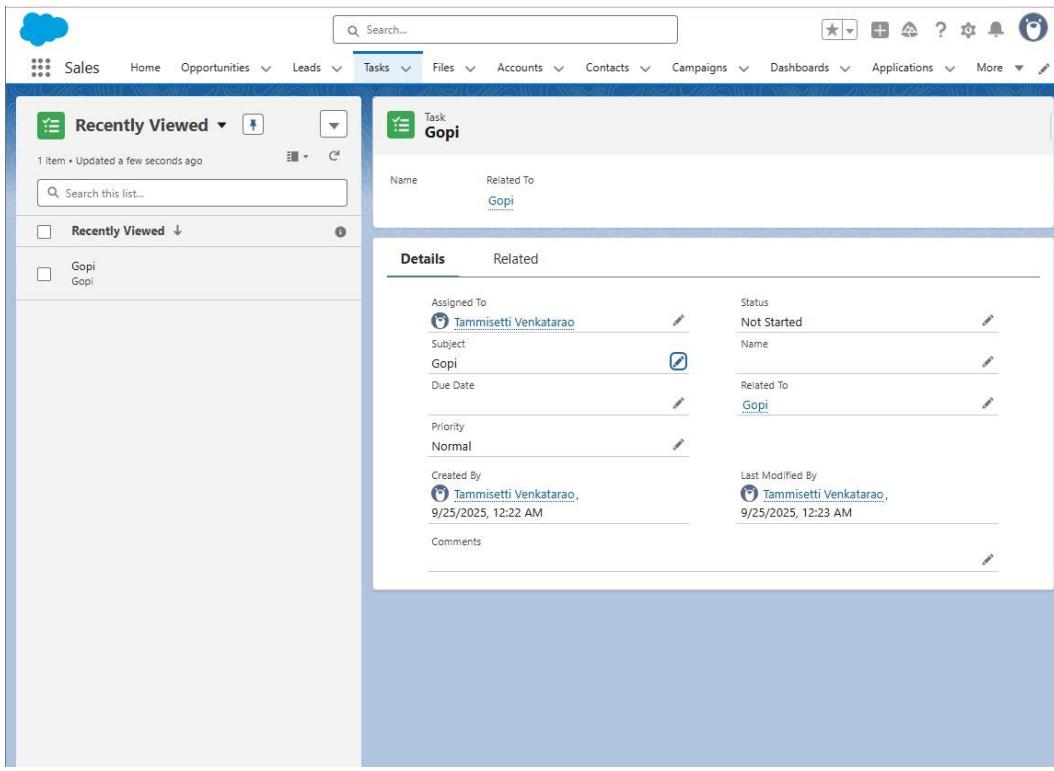
Activity

Filters: All time • All activities • All types

Upcoming & Overdue

No activities to show. Get started by sending an email, scheduling a task, and more.

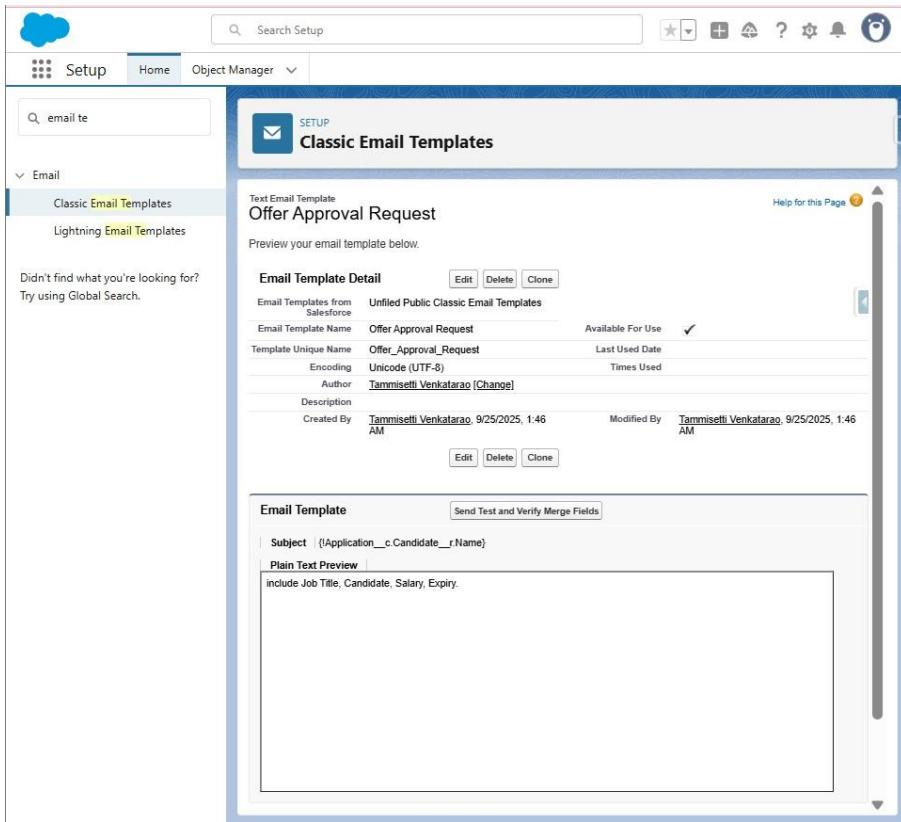
No past activity. Past meetings and tasks marked as done show up here.



## 5) Approval Process: Offer Approval (Department Head)

### Create Email Template (Offer Approval Request)

1. Setup → Email Templates → New Email Template.
  - Name: Offer\_Approval\_Request
  - Subject: Approval Required: Offer for {!Application\_\_c.Candidate\_\_r.Name}
  - Body: include Job Title, Candidate, Salary, Expiry.
2. Save.



## Create Approval Process

1. Setup → Approval Processes → Select object Application → Create New Approval Process → Standard Wizard.
2. Name: Offer Approval Process.
3. Entry Criteria: ISPICKVAL(Stage\_\_c, "Offer") (or add threshold: AND(ISPICKVAL(Stage\_\_c, "Offer"), Salary\_Offered\_\_c > 50000)).
4. Approver: Record Owner's Manager or a specified Department Head.
5. Initial Submitters: Application Owner.
6. Initial Submission Actions: Field Update → Approval\_Status\_\_c = Pending.
7. Final Approval Actions: Field Update → Offer\_Status\_\_c = Approved; Send Email Alert to Candidate.
8. Final Rejection Actions: Field Update → Offer\_Status\_\_c = Rejected; Send Email Alert.
9. Activate Approval Process.

**Optional:** Create a Record-Triggered Flow that submits the Application for approval automatically when Stage changes to Offer.

**Test:** Move Application Stage to Offer → approver receives email and can approve or reject; check field updates.

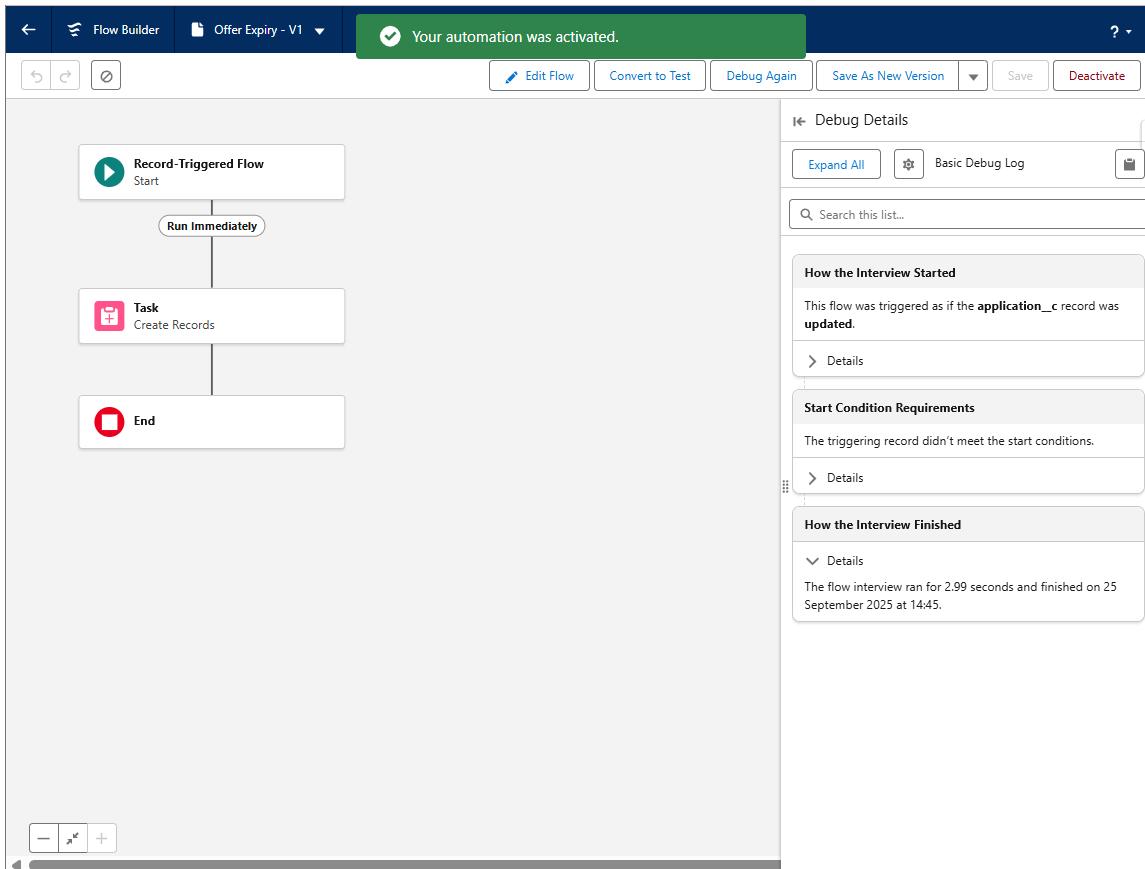
Field	Value
Unique Name	Offer_Approval_Process
Description	ISPICKVAL(Stage__c, "Offer")
Record Editability	Administrator OR Current Approver
Approval Assignment Email Template	Offer Approval Request
Initial Submitters	Candidate Owner
Created By	Tammisetti Venkatarao
Modified By	Tammisetti Venkatarao
Next Automated Approver Determined By	
Manager of Record Owner	

## 6) Scheduled Path in Record-Triggered Flow (Reminders / SLA)

**Use case:** Remind recruiter/candidate 3 days before Offer expiry or remind if no response.

1. Create Record-Triggered Flow on **Application\_c** (trigger on create or update).
2. Add **Scheduled Path**: Run after **Offer\_Expiry\_Date\_c - 3 Days**.
3. In scheduled path add **Decision**: If **Stage\_c = Offer AND Offer\_Accepted\_c = false**, then **Create Task** or **Send Email Alert** to candidate & recruiter.
4. Save & Activate.

**Test:** Create an Application with Offer Expiry in near future and debug scheduled path or wait for execution.



## 7) Auto-update Job Opening Positions (Roll-up)

### Option A — Roll-Up Summary (if Master-Detail exists)

1. If Application\_\_c is master-detail to Job\_Opening\_\_c, create **Roll-Up Summary** on Job\_Opening\_\_c:
  - Summarized Object: Application\_\_c
  - Roll-up Type: COUNT where Stage\_\_c = Hired.

The screenshot shows the Salesforce Setup interface with the following details:

- Setup** icon in the top left.
- Search Setup** bar at the top.
- Object Manager** tab selected in the top navigation.
- Job Opening** object selected in the main content area.
- Fields & Relationships** section selected in the sidebar.
- Custom Field Definition Detail** for **Number of Hired Applications**.
- Field Information** section:
 

Field Label	Number of Hired Applications
Field Name	Number_of_Hired_Applications
API Name	Number_of_Hired_Applications_c
Description	(empty)
Help Text	(empty)
Data Owner	(empty)
Field Usage	(empty)
Data Sensitivity Level	(empty)
Compliance Categorization	(empty)
- Created By**: Tammisetti Venkatarao, 9/25/2025, 2:25 AM
- Modified By**: Tammisetti Venkatarao, 9/25/2025, 2:25 AM
- Roll-Up Summary Options** section:
 

Data Type	Roll-Up Summary
Summarized Object	Application
Filter Criteria	Stage EQUALS Hired
Summary Type	COUNT
- Related Lookup Filters**, **Restriction Rules**, **Scoping Rules**, **Object Access**, **Triggers**, **Flow Triggers**, **Validation Rules**, and **Conditional Field Formatting** sections listed in the sidebar.

**Test:** Mark Application as Hired → Job Opening counters update.

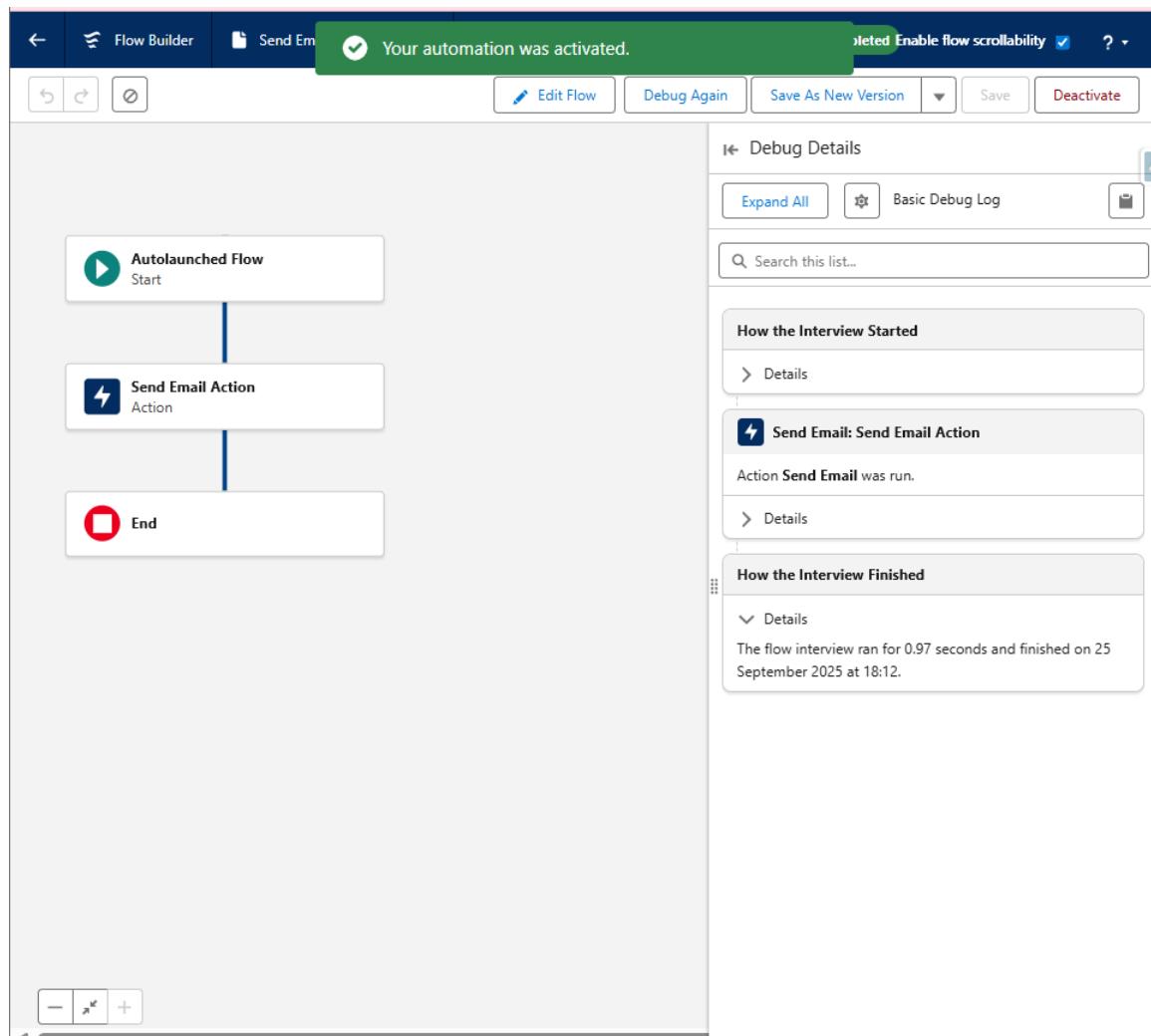
## 8) Auto-convert Leads to Candidates (incoming from LinkedIn / Site)

1. Create Record-Triggered Flow on **Lead** when Status = Closed - Converted or a custom checkbox **Convert\_to\_Candidate\_c** = true.
2. Add **Create Records** → Object: **Candidate\_c**. Map fields from Lead: FirstName, LastName, Email, Phone, Source = LinkedIn.
3. Optionally create **Application\_c** to link Candidate → Job\_Opening if Lead included interested job.
4. Update Lead to mark converted. Save & Activate.

**Note:** For robust lead conversion (account/contact/opportunity creation) consider Apex Database.LeadConvert or an invocable Apex class.

## 9) Subflows (Reusable) & Best Practices

- Create reusable subflows for repeated actions (e.g., Notify\_Recruiter\_Subflow to send email & custom notification).
- Name flows clearly and add descriptions (purpose, owner, inputs/outputs).
- Add **Fault Paths** on actions to log errors to a custom object (e.g., Flow\_Error\_\_c) for admin review.
- Avoid heavy synchronous processing; for bulk operations use scheduled flows or Batch Apex.
- Test flows with bulk updates to ensure governor limits are respected.



## **10) Send Custom Notifications (In-app)**

1. Setup → **Notification Builder** → **Notification Types** → New.
  - Label: Recruitment\_Update → Save.
2. In Flow add **Action** → **Send Custom Notification**, select Recruitment\_Update, set title/body and recipient (e.g., recruiter or panel member).
3. Test: Notification shows in Salesforce bell icon and mobile app.

## **11) Test & Debug Flows**

- Use **Debug** in Flow Builder to step through flow execution with sample input.
- Monitor **Paused and Failed Flow Interviews** (Setup → Paused & Failed Flow Interviews).
- Use **Debug Logs** (Setup → Debug Logs) especially when flows invoke Apex or platform events.
- Run bulk tests (update multiple records) to validate performance and bulk-safety.

## **12) Deployment of Flows & Automation**

- In Sandbox: Create an **Outbound Change Set** → add Flows, Email Templates, Email Alerts, Approval Processes, Notification Types.
- Upload Change Set to Production. In Production, **Validate** and **Deploy**.
- Alternatively: use **SFDX / CI-CD** for version-controlled deployments.
- Post-deployment: run smoke tests (create sample records and exercise each automation path).

## **13) Monitoring & Maintenance**

- Build an **Admin Dashboard** with Flow error counts, paused flows, and pending approvals.
- Document flow owners, inputs/outputs, and purpose in project documentation.
- Schedule periodic reviews to ensure automations reflect business needs.

- Keep a change log for all flow/config updates for audit/tracking.

#### **14) Sample Deliverables for Phase 4**

- List of Validation Rules (names, formulas, purpose).
- Flow names, diagrams (Visio/Miro) and descriptions.
- Email Templates and Email Alerts list with bodies.
- Approval Process specification and email templates.
- Custom Notification types and usage doc.
- Test cases and sandbox testing logs.
- Deployment Change Set or SFDX manifest.

#### **15) Quick Checklist (perform in Sandbox first)**

- Create and test Validation Rules.
- Create Email Templates and Email Alerts.
- Build and activate Record-Triggered Flows (Interview creation, Task creation, Submit for Approval).
- Build Scheduled Paths for reminders.
- Create Approval Process for Offers and test approval/rejection.
- Implement Roll-up/Flow to update Job Opening counts.
- Create Notification Types and test in-app/mobile notifications.
- Create Change Set and validate deployment.

## Phase 5: Apex Programming (Developer)

### Goal:

Implement server-side logic to support business rules and automations that are not possible (or practical) with declarative tools. Deliver bulk-safe triggers, Apex classes, asynchronous processing (Batch, Queueable, Scheduled, @future), robust test coverage, and deployable packages.

### 1) Create Apex Classes (basic steps)

1. Click **Setup** → Quick Find → **Apex Classes** → **New**.
2. Enter Apex code in the editor (example classes below).
3. Click **Save**.
4. For larger projects, use **VS Code + Salesforce Extensions** and SFDX for source control.

### 2) Trigger Design Pattern (one trigger per sObject)

**Why:** keeps logic maintainable and testable. All triggers delegate to a handler class.

#### Steps to create a trigger and handler:

1. Setup → Quick Find → **Apex Triggers** → **New** (or use Developer Console).
2. Create a simple trigger that delegates:

```
trigger ApplicationTrigger on Application__c (before insert, after update) {  
    if (Trigger.isBefore && Trigger.isInsert) {  
        ApplicationTriggerHandler.beforeInsert(Trigger.new);  
    }  
    if (Trigger.isAfter && Trigger.isUpdate) {  
        ApplicationTriggerHandler.afterUpdate(Trigger.new, Trigger.oldMap);  
    }  
}
```

SETUP > OBJECT MANAGER

## Application

- [Details](#)
- [Fields & Relationships](#)
- [Page Layouts](#)
- [Lightning Record Pages](#)
- [Buttons, Links, and Actions](#)
- [Compact Layouts](#)
- [Field Sets](#)
- [Object Limits](#)
- [Record Types](#)
- [Related Lookup Filters](#)
- [Search Layouts](#)
- [List View Button Layout](#)
- [Restriction Rules](#)
- [Scoping Rules](#)
- [Object Access](#)
- [Triggers](#)
- [Flow Triggers](#)
- [Validation Rules](#)

### Apex Trigger ApplicationTrigger

[« Back to List](#) [Help for this Page](#)

Name	ApplicationTrigger	sObject Type	Application
Code Coverage	0% (0/4)	Status	Active
Created By	Tammisetti Venkatarao, 9/25/2025, 10:01 AM	Last Modified By	Tammisetti Venkatarao, 9/25/2025, 10:01 AM
Namespace Prefix			

Apex Trigger Version Settings Trace Flags

```
1 trigger ApplicationTrigger on Application__c (before insert, after update) {
2   if (Trigger.isBefore && Trigger.isInsert) {
3     ApplicationTriggerHandler.beforeInsert(Trigger.new);
4   }
5   if (Trigger.isAfter && Trigger.isUpdate) {
6     ApplicationTriggerHandler.afterUpdate(Trigger.new, Trigger.oldMap);
7   }
8}
```

[Edit](#) [Delete](#) [Download](#) [Show Dependencies](#)

<https://orgfarm-5ca6a2481a-dev-ed.develop.lightning.force.com/lightning/setup/ObjectManager/01lgL000002lxRt/ApexTriggers/page?...>

3. Create the handler class (Apex Class → New):

```
public with sharing class ApplicationTriggerHandler {
    public static void beforeInsert(List<Application__c> newList) {
        for (Application__c a : newList) {
            if (a.Application_Date__c == null) a.Application_Date__c = Date.today();
        }
    }
}
```

```
public static void afterUpdate(List<Application__c> newList, Map<Id, Application__c> oldMap) {
```

```

List<Id> jobIds = new List<Id>();

for (Application__c a : newList) {

    Application__c oldA = oldMap.get(a.Id);

    if (oldA.Stage__c != 'Hired' && a.Stage__c == 'Hired' && a.Job_Opening__c != null) {

        jobIds.add(a.Job_Opening__c);

    }

}

if (!jobIds.isEmpty()) {

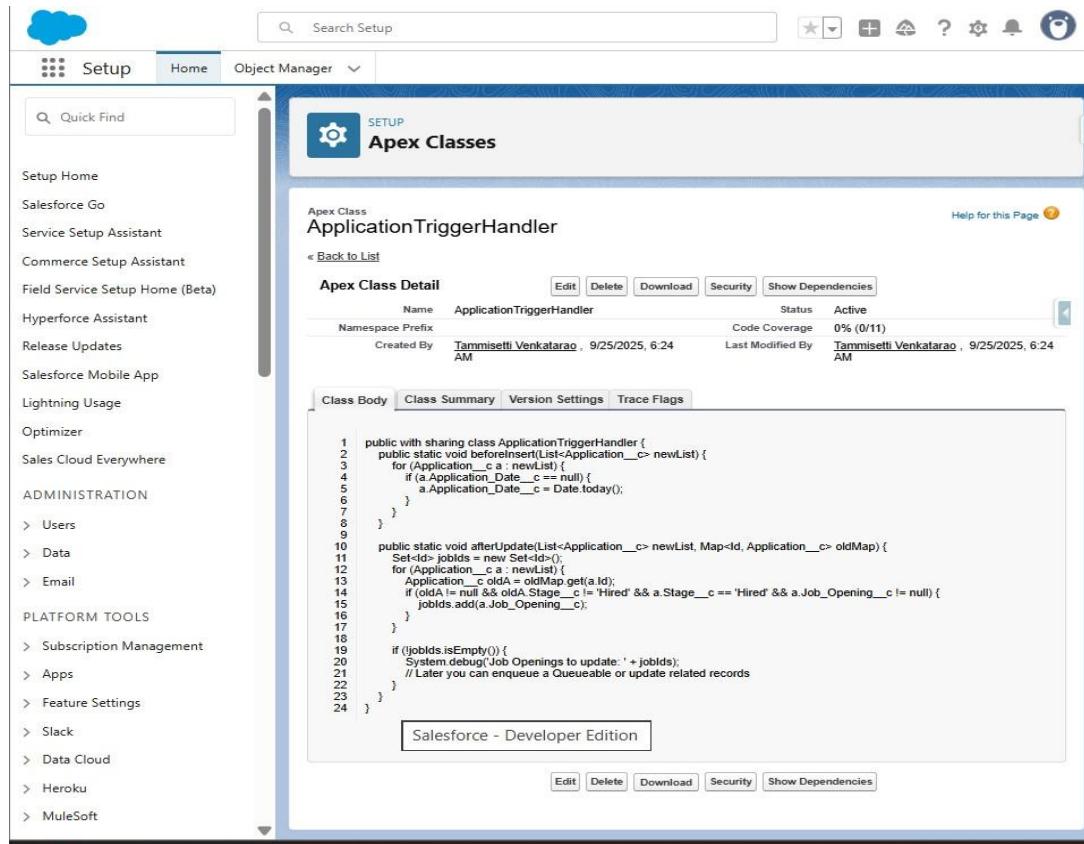
    System.enqueueJob(new UpdateJobOpeningQueueable(jobIds));

}

}

}

```



## **Notes / Best practices:**

- Only one trigger per object.
- Trigger body just delegates to handler(s).
- Use oldMap to detect field changes.
- Avoid SOQL/DML inside loops — use collections (Set/Map/List).

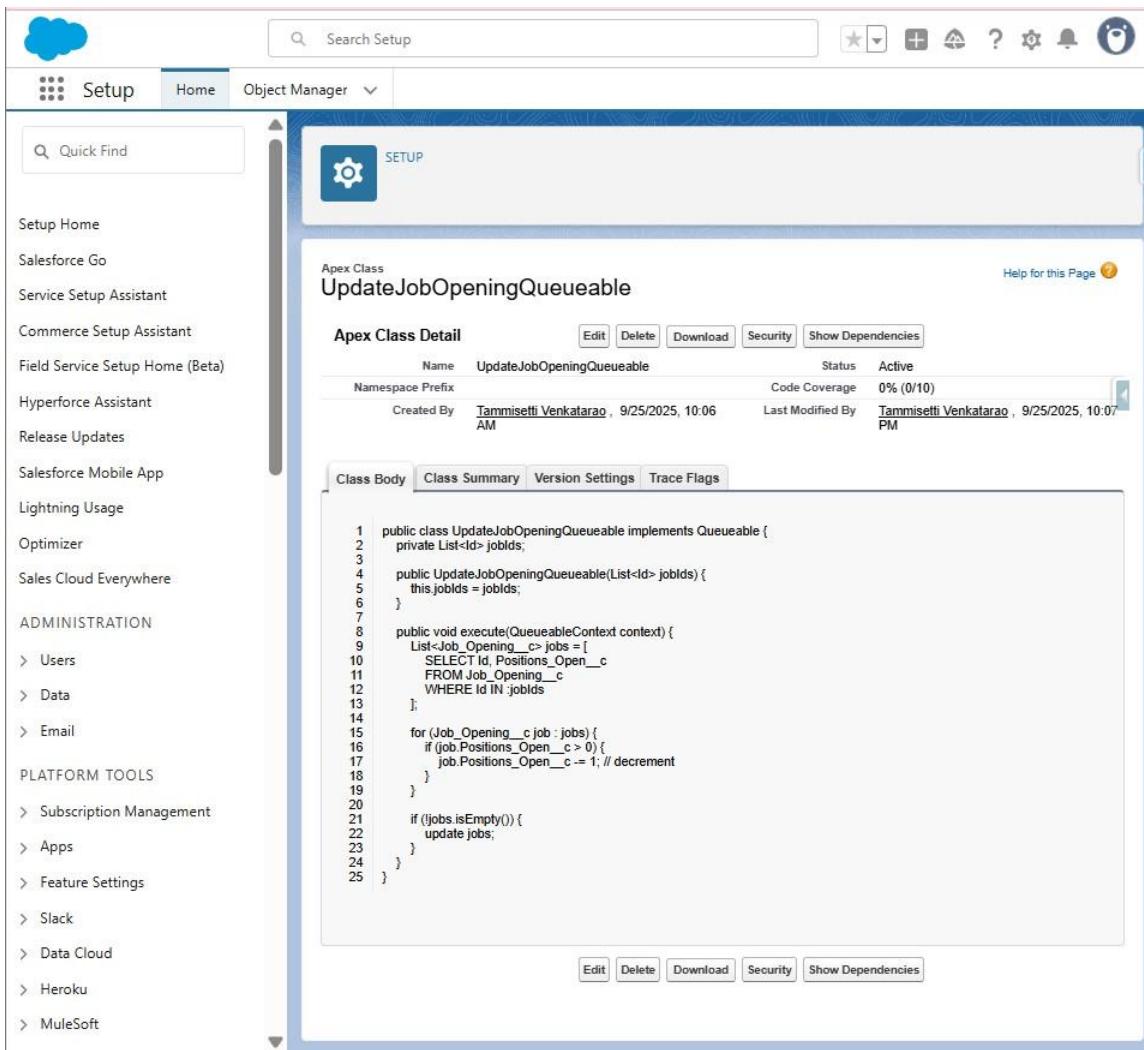
## **3) Queueable Apex (for small async jobs)**

**Use case:** Update Job Opening counters after hires or call external resume-parsing API.

**Create Queueable class (Apex Class → New):**

```
public class UpdateJobOpeningQueueable implements Queueable {  
    private List<Id> jobIds;  
  
    public UpdateJobOpeningQueueable(List<Id> jobIds) {  
        this.jobIds = jobIds;  
    }  
  
    public void execute(QueueableContext ctx) {  
        List<Job_Opening__c> jobs = [SELECT Id, Positions_Filled__c FROM  
        Job_Opening__c WHERE Id IN :jobIds];  
  
        for (Job_Opening__c j : jobs) {  
            Integer filled = (j.Positions_Filled__c == null) ? 0 :  
            Integer.valueOf(j.Positions_Filled__c);  
  
            j.Positions_Filled__c = filled + 1;  
        }  
  
        try {  
            update jobs;  
        } catch (DmlException e) {  
            // Log errors to custom object or use Platform Events  
        }  
    }  
}
```

```
        System.debug('Error updating jobs: ' + e.getMessage());  
    }  
}  
}
```



**Test / run:** In trigger handler we used System.enqueueJob(new UpdateJobOpeningQueueable(jobIds)). In tests, wrap enqueue in Test.startTest()/Test.stopTest() to execute queueable.

#### 4) Batch Apex (for large data sets)

**Use case:** Weekly job to summarize pending applications and email recruiters; operate on thousands of records.

**Create Batch class:**

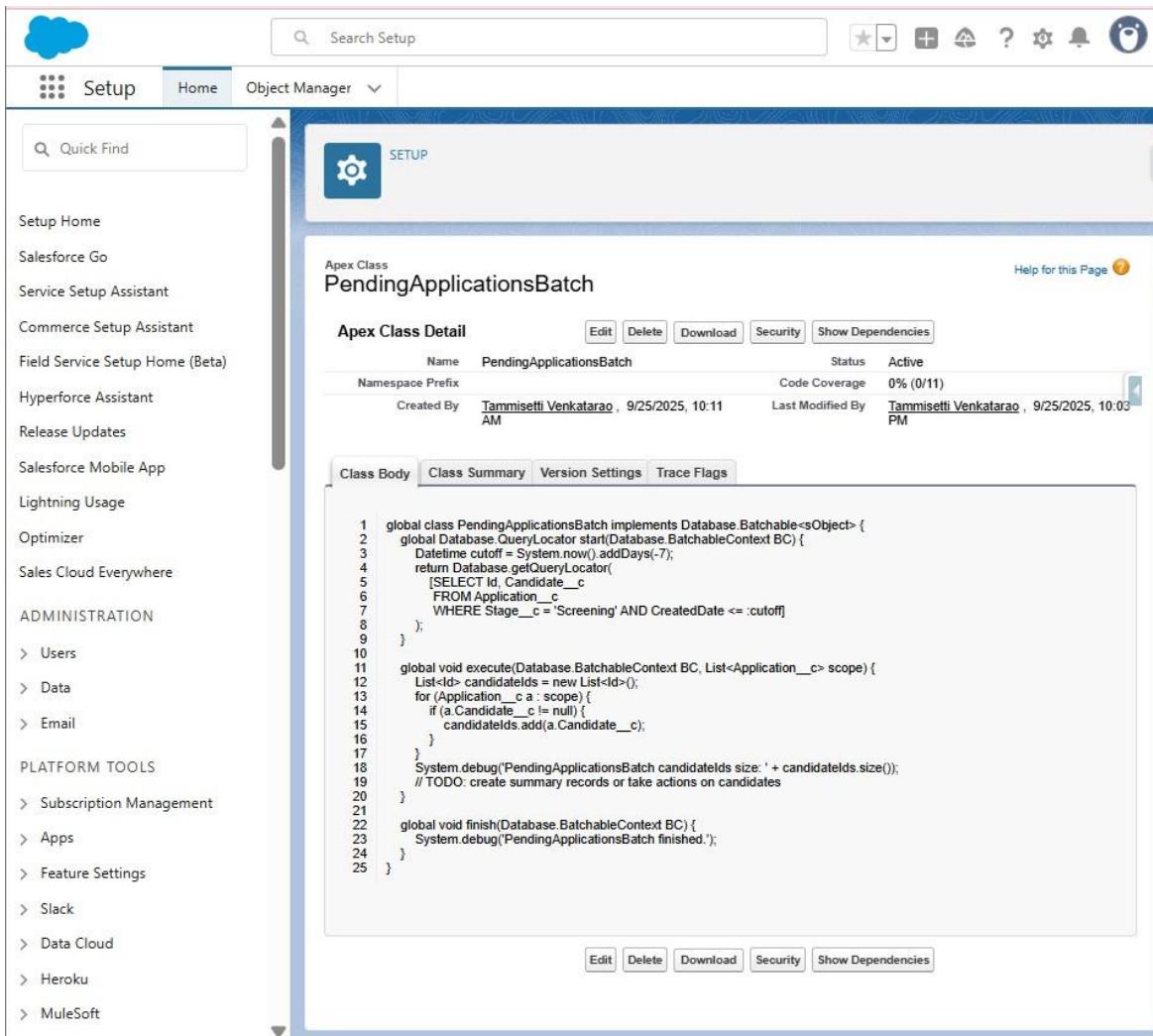
```
global class PendingApplicationsBatch implements Database.Batchable<sObject> {

    global Database.QueryLocator start(Database.BatchableContext BC) {
        return Database.getQueryLocator(
            [SELECT Id, OwnerId, Candidate__c FROM Application__c
             WHERE Stage__c = 'Screening' AND CreatedDate <= :System.now().addDays(-7)]
        );
    }

    global void execute(Database.BatchableContext BC, List<Application__c> scope) {
        // build map of owner -> list of candidate Ids
        Map<Id, List<Id>> ownerMap = new Map<Id, List<Id>>();
        for (Application__c a : scope) {
            if (!ownerMap.containsKey(a.OwnerId)) ownerMap.put(a.OwnerId, new List<Id>());
            ownerMap.get(a.OwnerId).add(a.Candidate__c);
        }
        // Example: create queued emails or notifications per owner
        // (Implementation detail: assemble email body and call Messaging)
    }

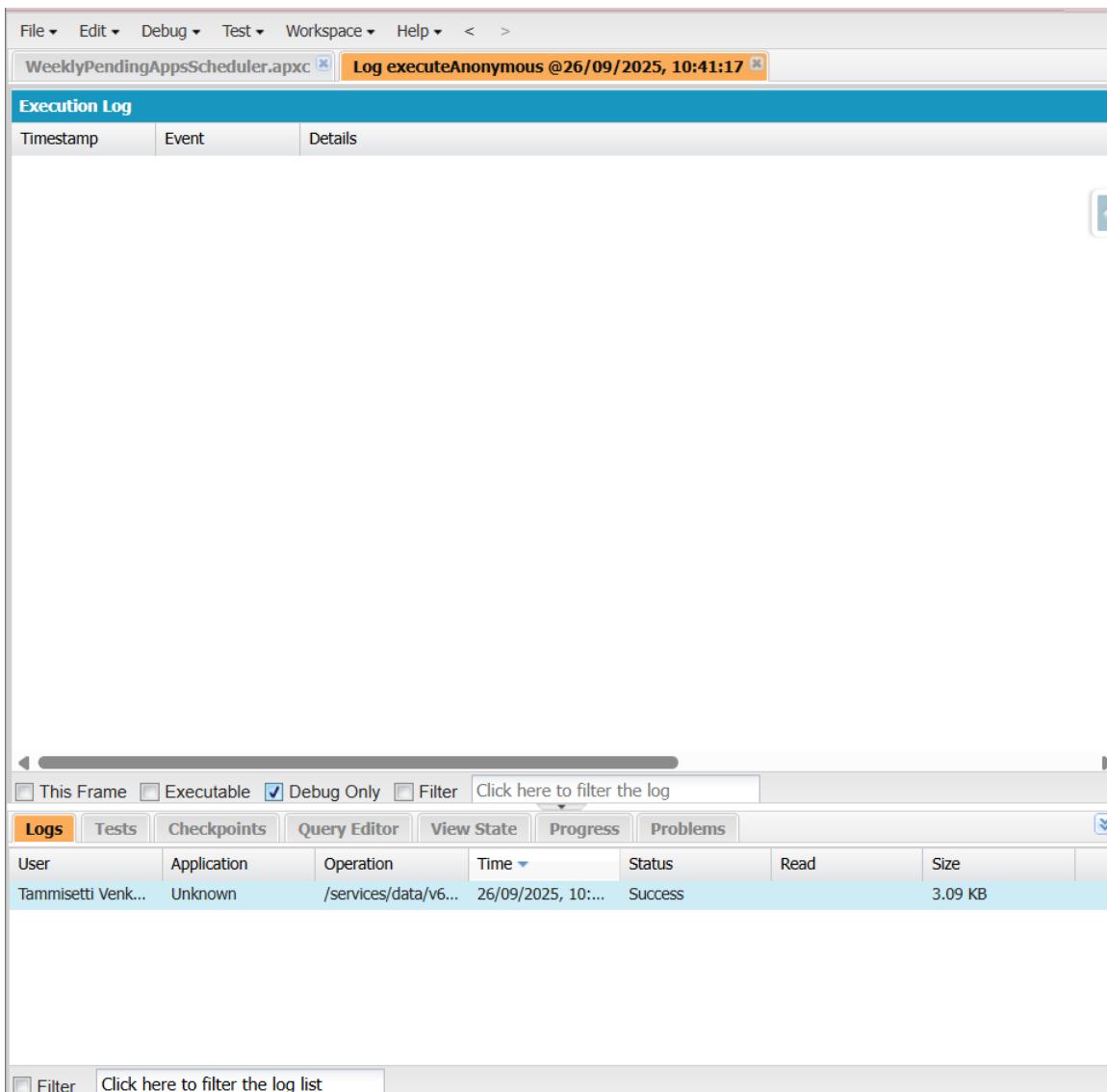
    global void finish(Database.BatchableContext BC) {
        // optional: send a summary email to admins or schedule next batch
    }
}
```

}



**Schedule batch:** Wrap it into a Schedulable class:

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



### Schedule via UI:

- Setup → Apex Classes → **Schedule Apex** → select WeeklyPendingAppsScheduler → set frequency/time.

### 5) @future and Callouts

**Use case:** Make callouts to resume-parsing APIs asynchronously.

#### Example @future callout:

```
public class ResumeParserService {
```

```
    @future(callout=true)
```

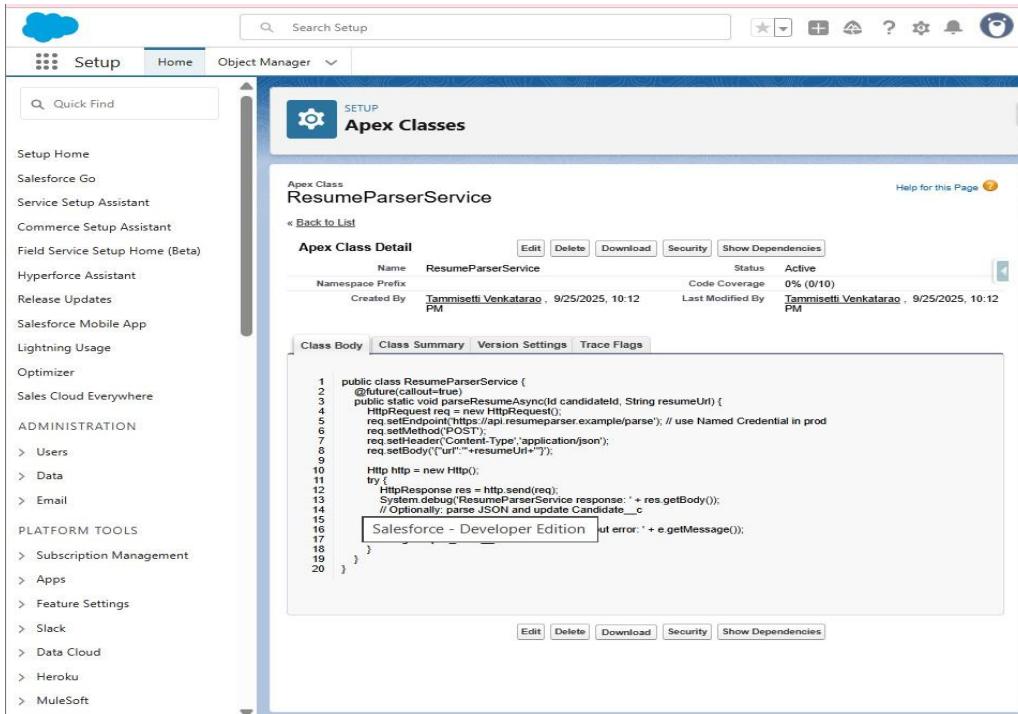
```

public static void parseResumeAsync(Id candidateId, String resumeUrl) {

    HttpRequest req = new HttpRequest();
    req.setEndpoint('https://api.resumeparser.example/parse');
    req.setMethod('POST');
    req.setBody('{"url":"' + resumeUrl + '"}');

    Http http = new Http();
    try {
        HttpResponse res = http.send(req);
        // process response and update Candidate__c fields using direct DML
    } catch (Exception e) {
        System.debug('Callout error: ' + e.getMessage());
    }
}
}

```



**Note:** For more advanced control use Queueable with Database.AllowsCallouts.

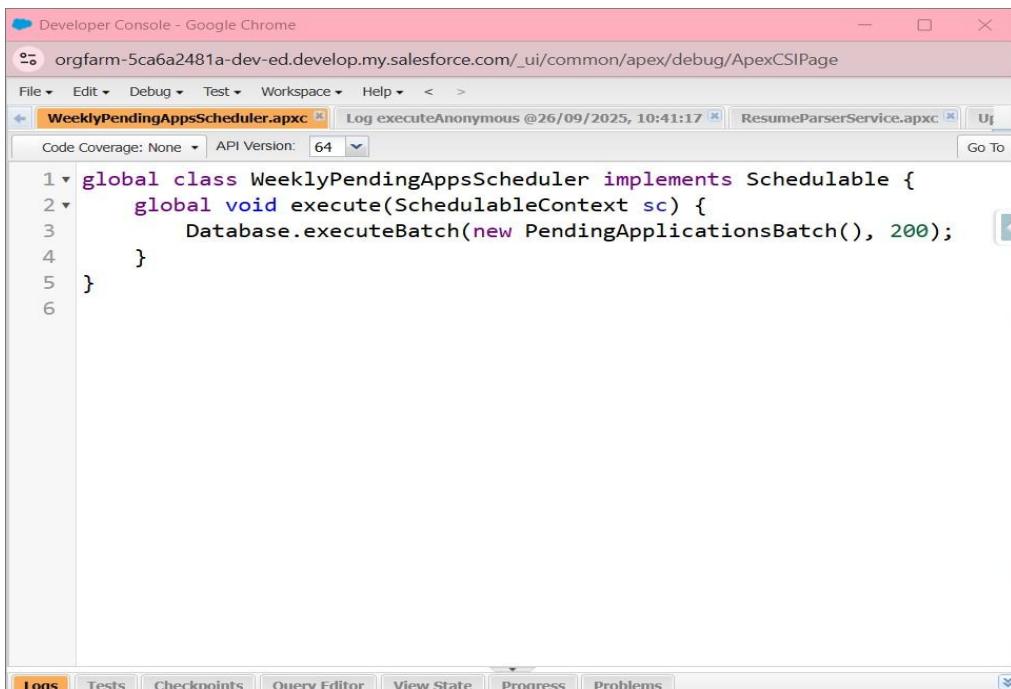
## 6) Scheduled Apex (recurring jobs)

**Steps to create:**

1. Apex Class → New → write a class that implements Schedulable.
2. Save.
3. Setup → **Apex Classes** → **Schedule Apex** → select class → set schedule.

**Example:**

```
global class WeeklyReportsScheduler implements Schedulable {  
    global void execute(SchedulableContext sc) {  
        // create and send weekly hiring report  
        Database.executeBatch(new PendingApplicationsBatch(), 200);  
    }  
}
```



The screenshot shows the Salesforce Developer Console in Google Chrome. The URL is orgfarm-5ca6a2481a-d-dev-ed.develop.my.salesforce.com/\_ui/common/apex/debug/ApexCSIPage. The window title is "Developer Console - Google Chrome". The tabs at the top include "File", "Edit", "Debug", "Test", "Workspace", "Help", and "Logs" (which is currently selected). Below the tabs, there are two tabs: "WeeklyPendingAppsScheduler.apxc" and "ResumeParserService.apxc". The main area displays the Apex code for "WeeklyPendingAppsScheduler.apxc". The code is as follows:

```
1 global class WeeklyPendingAppsScheduler implements Schedulable {  
2     global void execute(SchedulableContext sc) {  
3         Database.executeBatch(new PendingApplicationsBatch(), 200);  
4     }  
5 }  
6
```

The code coverage dropdown shows "None" and the API version is set to "64". At the bottom of the developer console, there are tabs for "Logs", "Tests", "Checkpoints", "Query Editor", "View State", "Progress", and "Problems".

## 7) SOQL & SOSL — Performance & Safety

### Best practices:

- Never put SOQL inside a loop.
- Use **selective filters** and indexed fields (Id, external id, lookup).
- Use **aggregate queries** or roll-up summaries when possible.
- Prefer **LIMIT** where appropriate for admin queries.
- Use **SOSL** for text search across multiple objects/fields: FIND 'John\*' IN ALL FIELDS RETURNING Candidate\_c(Id, Name, Email\_c).

## 8) Bulkification patterns (must-have)

- Use Maps to group related records by Id.
- Collect all Ids to query once, then map results.
- Minimize DML statements — group into a single update/insert.
- Use Database.insert(list, false) for partial success handling if needed.

### Example pattern in handler:

```
Map<Id, Job_Opening_c> jobMap = new Map<Id, Job_Opening_c>(  
    [SELECT Id, Positions_Filled_c FROM Job_Opening_c WHERE Id IN :jobIds]  
);  
  
List<Job_Opening_c> jobsToUpdate = new List<Job_Opening_c>();  
for (Id jId : jobIds) {  
    Job_Opening_c j = jobMap.get(jId);  
    j.Positions_Filled_c = (j.Positions_Filled_c == null ? 0 : j.Positions_Filled_c) + 1;  
    jobsToUpdate.add(j);  
}  
Database.update(jobsToUpdate);
```

```
update jobsToUpdate;
```

## 9) Exception Handling & Logging

- Wrap DML operations in try/catch. Use Database.SaveResult or Database.Update for partial processing.
- Create a custom object (e.g., Apex\_Error\_\_c) to log exceptions from async jobs.

### Example:

```
try {  
    update jobsToUpdate;  
}  
} catch (DmlException dmx) {  
    for (Integer i=0; i<jobsToUpdate.size(); i++) {  
        Apex_Error__c err = new Apex_Error__c(  
            Name = 'Job update error',  
            Message__c = dmx.getMessage(),  
            Record_Id__c = String.valueOf(jobsToUpdate[i].Id)  
        );  
        insert err;  
    }  
}
```

## 10) Test Classes — mandatory & examples

### Rules:

- Create @isTest classes and methods.
- Use Test.startTest() and Test.stopTest() when testing async jobs.

- For callouts use HttpCalloutMock.
- Achieve > 75% coverage for Apex you deploy.

**Sample test for the queueable trigger flow:**

```

@isTest

private class ApplicationTriggerTest {
    @isTest static void testHireUpdatesJob() {
        // Prepare test data

        Job_Opening_c job = new Job_Opening_c(Name='Test Job',
Positions_Filled_c=0);

        insert job;

        Candidate_c cand = new Candidate_c(FirstName_c='John', LastName_c='Doe',
Email_c='j@d.com');

        insert cand;

        Application_c app = new Application_c(Job_Opening_c=job.Id,
Candidate_c=cand.Id, Stage_c='Interview');

        insert app;

        // Move to Hired and assert job update via queueable

        app.Stage_c = 'Hired';

        Test.startTest();

        update app; // trigger will enqueue queueable

        Test.stopTest();

        Job_Opening_c j = [SELECT Id, Positions_Filled_c FROM Job_Opening_c
WHERE Id = :job.Id];
    }
}

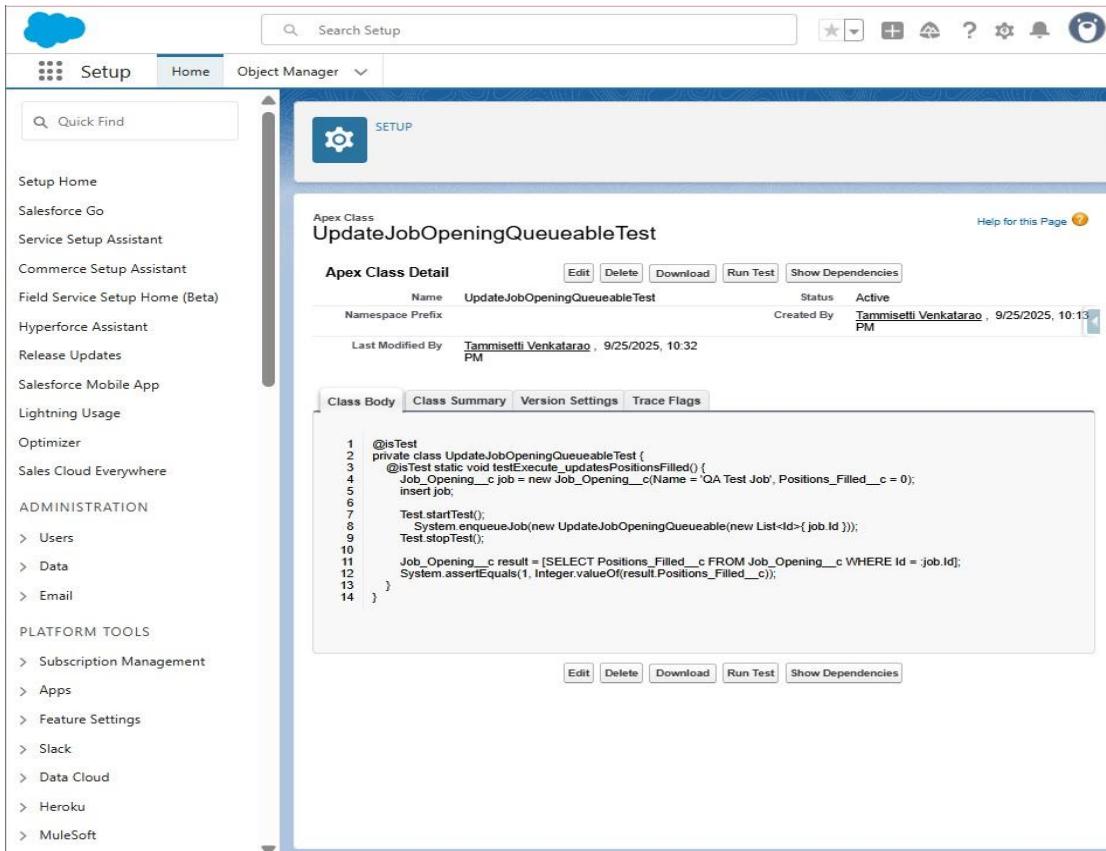
```

```

        System.assertEquals(1, Integer.valueOf(j.Positions_Filled__c));
    }

}

```



## Test batch:

```

@isTest

private class PendingApplicationsBatchTest {

    @isTest static void testBatchExecution() {

        // create data older than 8 days

        List<Application__c> apps = new List<Application__c>();

        for(Integer i=0;i<5;i++){

```

```

        Application__c a = new Application__c(Stage__c='Screening',
Candidate__c=null);

        apps.add(a);

    }

insert apps;

Test.startTest();

PendingApplicationsBatch b = new PendingApplicationsBatch();

Database.executeBatch(b, 50);

Test.stopTest();

// Add asserts relevant to your batch (e.g., created summary records or emails
queued)

}

}

```

```

File ▾ Edit ▾ Debug ▾ Test ▾ Workspace ▾ Help ▾ < >
10:41:17 PendingApplicationsBatchTest.apxc
File ▾ Edit ▾ Debug ▾ Test ▾ Workspace ▾ Help ▾ < >
File ▾ Edit ▾ Debug ▾ Test ▾ Workspace ▾ Help ▾ < >
ResumeParserService.apxc UpdateJobOpeningQueueableTest.apxc PendingApplicationsBatchTest.apxc
Code Coverage: None API Version: 64 Run Test Go To
1 @isTest
2 private class PendingApplicationsBatchTest {
3     @isTest static void testBatchExecution() {
4         // Create sample applications older than 8 days
5         List<Application__c> apps = new List<Application__c>();
6         for (Integer i=0; i<5; i++) {
7             Application__c a = new Application__c(Stage__c='Screening'
8             apps.add(a);
9         }
10        insert apps;
11
12        Test.startTest();
13        PendingApplicationsBatch b = new PendingApplicationsBatch(
14            Database.executeBatch(b, 50);
15        Test.stopTest();
16
17        // Add assertions as appropriate for your implementation
18        System.assert(true); // placeholder - replace with real assert
19    }

```

## **11) Local Dev & Deployment (SFDX preferred)**

### **Local development:**

- Use **Visual Studio Code** + Salesforce Extensions Pack.
- Author Apex classes locally and push to scratch orgs or deploy to your dev org.

### **Deployment steps (Change Set):**

1. In Sandbox: Setup → **Outbound Change Sets** → New → add Apex Classes, Triggers, Tests, Email Templates → Upload to Production.
2. In Production: validate and deploy. Run all tests or run specific test classes if allowed.

### **Deployment steps (SFDX):**

- `sfdx force:source:deploy -p force-app/main/default/classes -u <orgAlias>`
- Use CI (GitHub Actions, Jenkins, CircleCI) to run tests and deploy.

## **12) Debugging & Logs**

1. Setup → **Debug Logs** → New to add yourself or user.
2. Reproduce the issue and view logs (Developer Console → Logs).
3. Use `System.debug()` with descriptive messages but remove excessive debug before production.

## **13) Deliverables for Phase 5**

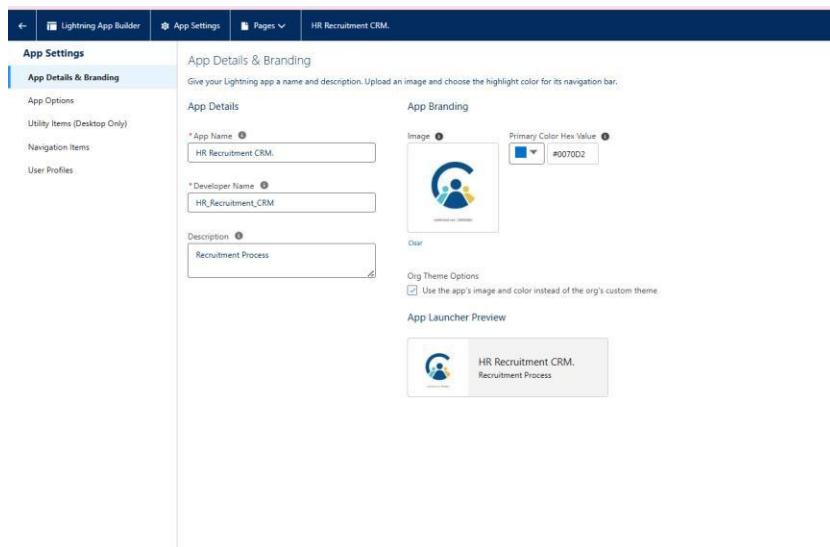
- Apex Trigger files and Handler classes.
- Queueable, Batch, Scheduled, and `@future` classes.
- Test classes with coverage reports and assertions.
- Error logging mechanism (custom object or platform event).
- Deployment artifacts (Change Set or SFDX manifest).

## Phase 6: User Interface Development

**Goal:** Enhance usability, create a professional look for the HR Recruitment CRM, and align the interface with company branding.

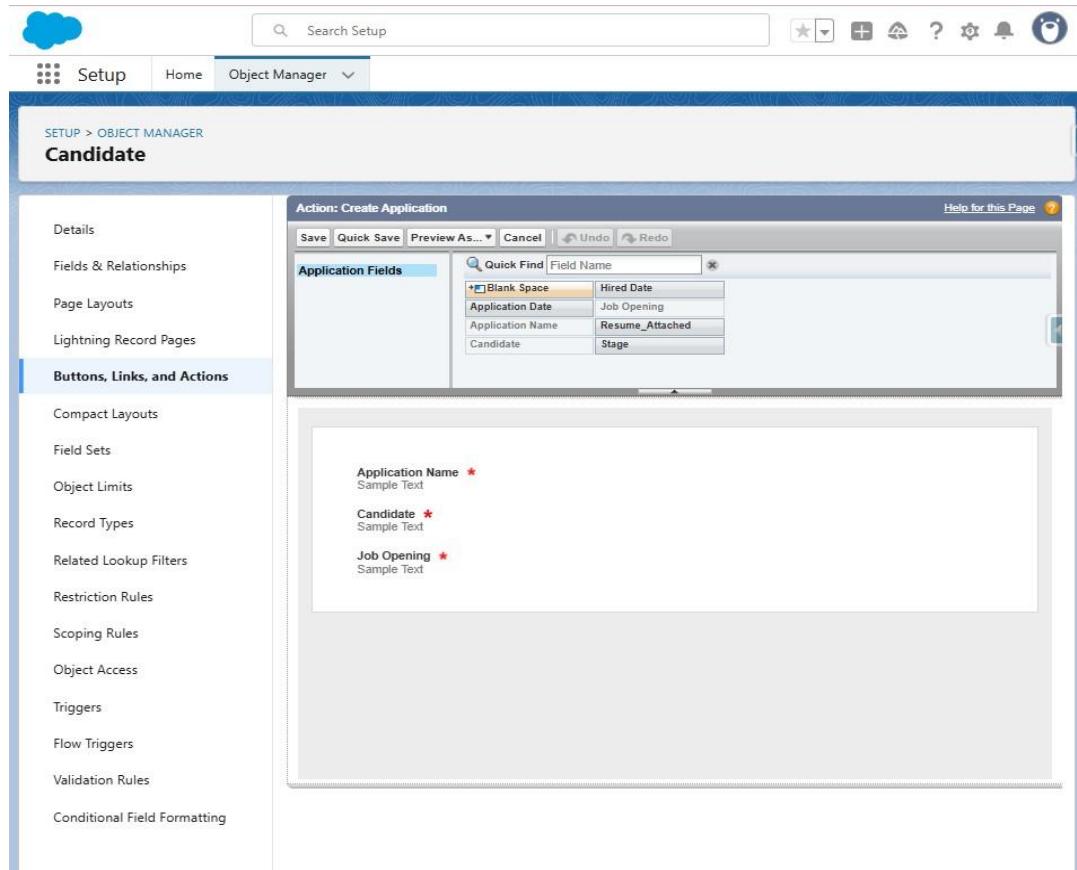
### 1. Create a Custom Lightning App

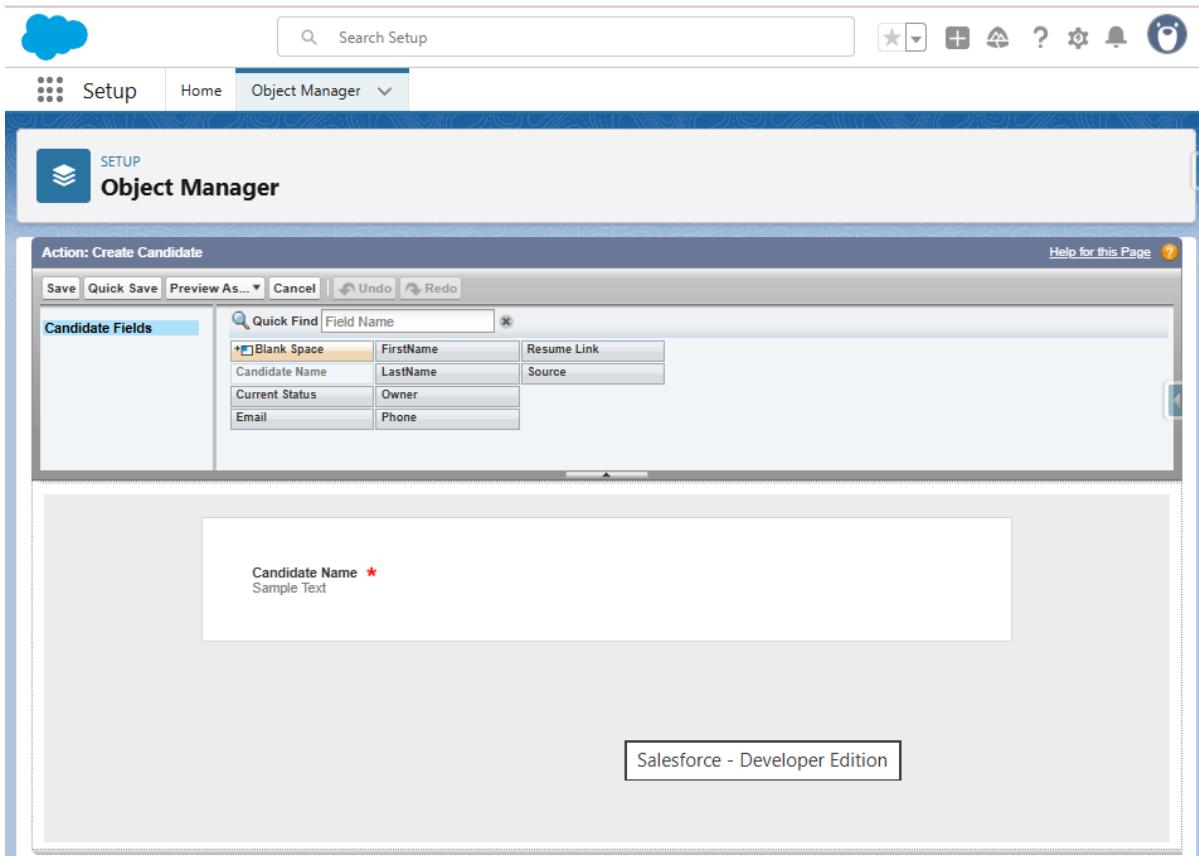
1. Click **Setup** (gear icon) → **App Manager**.
2. Click **New Lightning App**.
3. Enter:
  - **App Name:** HR Recruitment CRM.
  - **Developer Name:** HR\_Recruitment\_CRM.
  - Upload **Company Logo** (PNG/JPG, 128×128 px).
4. Configure:
  - Navigation Style: **Standard Navigation**.
  - App Options: Enable **App Personalization**, **Pinned Tabs**, **Setup Gear**.
5. Add Navigation Items: **Job Openings**, **Candidates**, **Applications**, **Interviews**, **Reports**, **Dashboards**, **Chatter**.
6. Assign the app to profiles: HR Manager, Recruiter, Department Head.
7. Click **Save & Finish**.



## 2. Create Global & Object-Specific Quick Actions

1. Setup → Object Manager → **Candidate** → Buttons, Links, and Actions → **New Action**.
2. Choose:
  - **Action Type:** Create Record.
  - **Target Object:** Application\_\_c.
  - **Label:** Create Application.
3. Add the new action to **Candidate Page Layout** under Salesforce Mobile and Lightning Experience Actions.
4. Global Quick Action:
  - Setup → Global Actions → **New Action** → Log a Call or New Task for recruiters.





### 3. Configure the Utility Bar

1. Setup → App Manager → Edit **HR Recruitment CRM App**.
2. Under **Utility Items**, click **Add Utility Item**.
3. Add:
  - **Notes** – For quick candidate notes.
  - **Chatter Feed** – For collaboration.
  - **Recent Items** – Quick navigation.
4. Save changes and refresh the app.

The screenshot shows the Lightning App Builder interface with the following details:

- Header:** Includes back, refresh, and search icons, followed by "Lightning App Builder", "App Settings", "Pages", "HR Recruitment CRM.", and a help icon.
- Left Sidebar (App Settings):** Contains sections for "App Details & Branding", "App Options", "Utility Items (Desktop Only)" (which is selected and highlighted in blue), "Navigation Items", and "User Profiles".
- Main Content Area:**
  - Section Title:** "Utility Items (Desktop Only)".
  - Description:** "Give your users quick access to productivity tools and add background utility items to your app."
  - Utility Bar Alignment:** Set to "Default".
  - Add Utility Item:** A button to add new items.
  - Recent Items:** A list item under "Recent Items" with an edit icon.
  - Chatter Feed:** A list item under "Chatter Feed" with an edit icon.
  - Notes:** A list item under "Notes" with an edit icon.
  - Properties:** A section for "Recent Items" with up/down arrows and a "Remove" button.
  - Utility Item Properties:** A collapsed section.
  - Label:** "Recent Items".
  - Icon:** "fallback X".
  - Panel Width:** "340".
  - Panel Height:** "480".
  - Start automatically:** An unchecked checkbox.
  - Component Properties:** A collapsed section.
  - Label:** "Custom".
  - Custom Label:** "Standard.RecentItems".
  - Objects:** "API Anomaly Event Store".
  - Select...:** A button to select objects.
  - Number of Records to Display:** "3".

## ✓ Deliverables

- Screenshot of the HR Recruitment CRM app tile with logo.
- Candidate and Job Opening record page screenshots showing custom components.
- Theme and Branding screenshot.
- Screenshot of Utility Bar configuration.

## Phase 7: Integration & External Access

**Goal:** Enable the HR Recruitment CRM to securely connect with external systems (resume parsing APIs, job boards, email services) and expose controlled access for integrations.

### 1. Remote Site Settings (Required for Callouts)

1. Setup → **Remote Site Settings** → New.
2. Enter:
  - Label: ResumeParserSite
  - URL: <https://api.resumeparser.com>
  - Active: Checked
3. Save → Allows Salesforce to call the external service.

### 2. Web Service Callouts (REST API for Resume Parsing)

**Use Case:** Parse candidate resumes uploaded in the system.

```
public class ResumeParserService {  
    @future(callout=true)  
    public static void parseResume(Id candidateId, String resumeUrl) {  
        HttpRequest req = new HttpRequest();  
        req.setEndpoint('callout:ResumeParserAPI/parse');  
        req.setMethod('POST');  
        req.setHeader('Content-Type', 'application/json');  
        req.setBody('{{ "resumeUrl": "'+resumeUrl+''}}');  
        Http http = new Http();  
        HttpResponse res = http.send(req);  
        if (res.getStatusCode() == 200) {  
            Map<String, Object> result =  
                (Map<String, Object>) JSON.deserializeUntyped(res.getBody());  
        }  
    }  
}
```

```

Candidate__c cand = [SELECT Id FROM Candidate__c WHERE Id =
:candidateId LIMIT 1];

    cand.Experience__c = (String)result.get('experience');

    cand.Skills__c = (String)result.get('skills');

    update cand;

}

}

}

```

### **3. External Services (Optional — for HR API catalog)**

1. Setup → **External Services**.
2. Register schema from job portal API (Swagger/OpenAPI).
3. Auto-generate Apex Actions for integration (like posting job openings).

### **4. Platform Events (For Notifications)**

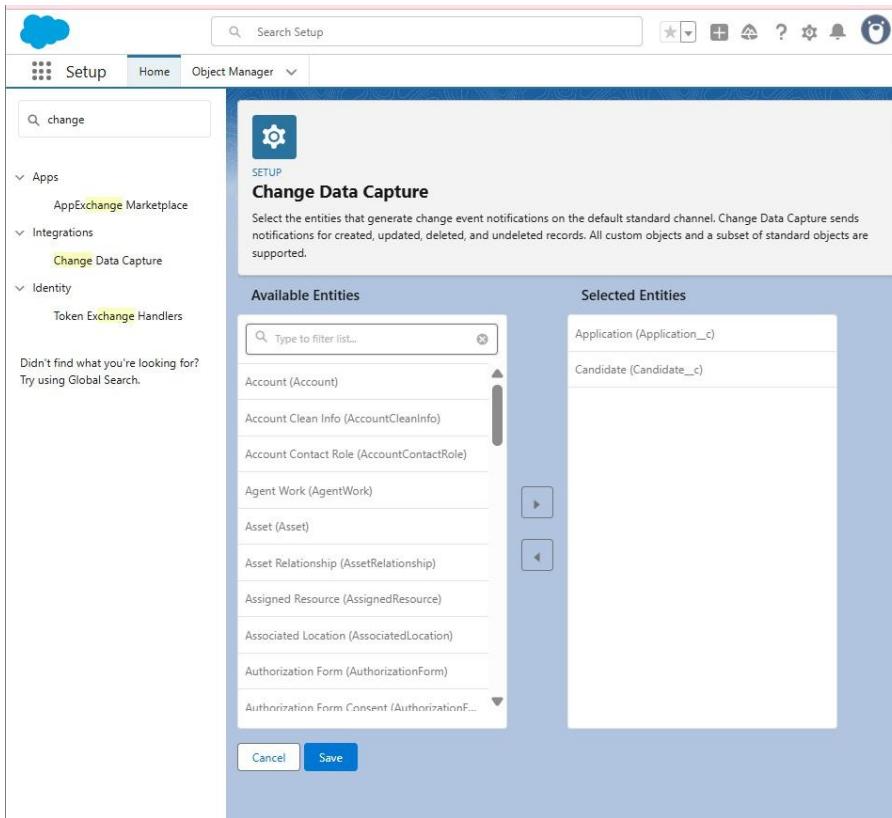
**Use Case:** Notify recruiters instantly when a candidate is hired.

1. Setup → **Platform Events** → **New**.
  - Label: Candidate\_Hired\_\_e.
  - Fields: CandidateId, JobOpeningId.
2. Publish event in Apex when Application stage = Hired.
3. Subscribe via Flow or external system.

### **5. Change Data Capture (CDC)**

**Use Case:** Sync candidate updates with external HRMS system.

1. Setup → **Change Data Capture** → Enable for Candidate\_\_c and Application\_\_c.
2. External system subscribes to CDC events via CometD or Pub/Sub API.

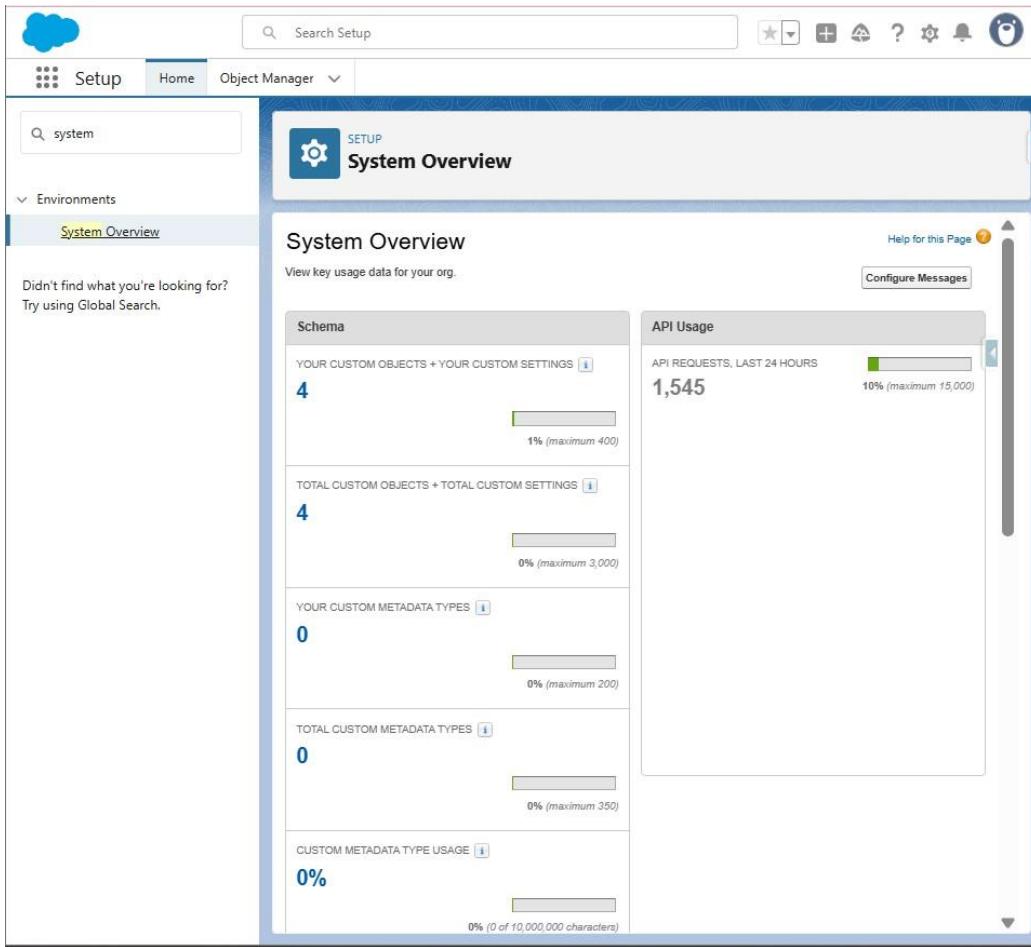


## 7. OAuth & Authentication (Required for Secure Access)

1. Setup → App Manager → New Connected App.
2. Enter:
  - Name: HR Recruitment API Access
  - Enable **OAuth Settings** → Callback URL: external app URL.
  - Selected OAuth Scopes: Full access, Perform requests at any time.
3. Save → Use Consumer Key/Secret in integrations.

## 8. API Limits

1. Setup → System Overview → Monitor API Requests.
2. Ensure resume parsing and integrations do not exceed daily API limits.
3. Use **Batching** and **Queueable Apex** for large requests.



## 9. Salesforce Connect (Optional for External HRMS/ERP)

- Use when external HR database must be accessed in real-time without importing data.
- Setup → **Salesforce Connect** → **External Data Source** → Choose OData 2.0/4.0.

### ✓ Final Deliverables

- Working REST API callout for parsing resumes.
- Enabled Change Data Capture for Candidate & Application.
- Connected App with OAuth settings.
- Documented API limit monitoring plan.

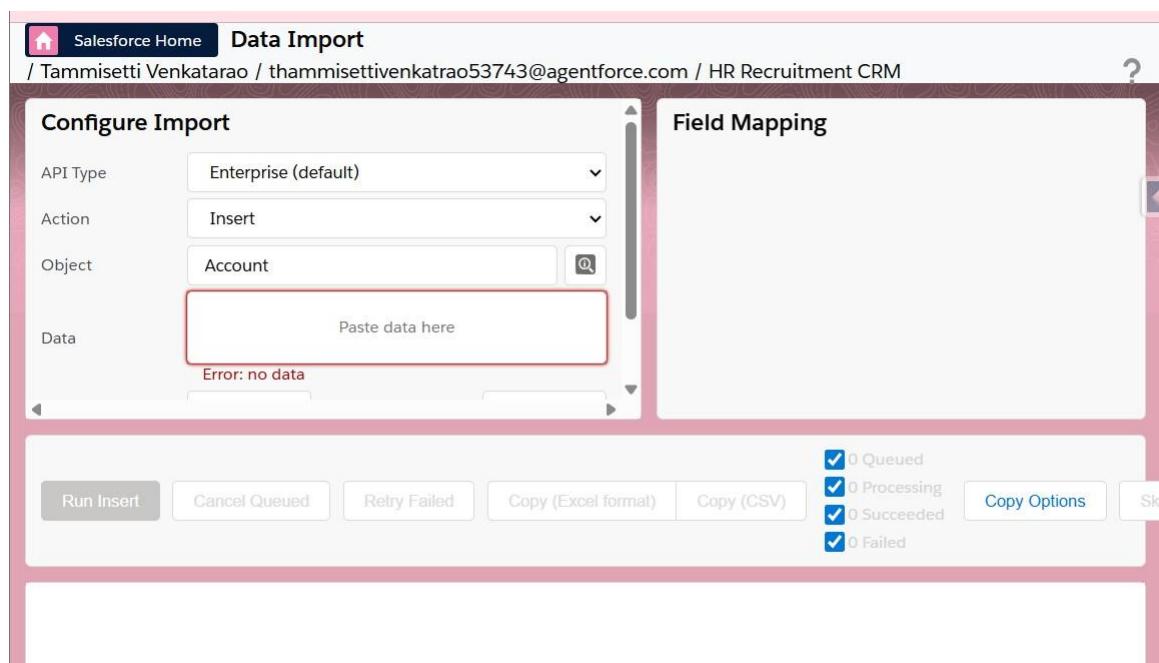
## Phase 8: Data Management & Deployment

**Goal:** Manage recruitment data efficiently, prevent duplicates, ensure regular backups, and deploy configurations/code reliably from Sandbox to Production.

### 1. Import Recruitment Data

#### a) Using Data Import Wizard (Small Data Sets)

1. App Launcher → **Data Import Wizard**.
2. Select **Candidate\_\_c** object.
3. Upload CSV file (fields: Name, Email, Phone, Resume\_Link\_\_c).
4. Map fields → Click **Start Import**.
5. Monitor progress in Bulk Data Load Jobs.



#### b) Using Data Loader (Large Data Sets)

1. Install **Data Loader** (Windows/Mac).

2. Login with Salesforce credentials + security token.
3. Choose **Insert** → Select object (e.g., Application\_\_c).
4. Upload CSV (Candidate Id, Job Opening Id, Stage).
5. Click **Next** → Map fields → **Finish**.
6. Verify inserted records in Salesforce.

The screenshot shows the Salesforce Data Export interface. At the top, it displays the user's name (Tammisetti Venkataraao) and email (thammisettivenkatrao53743@agentforce.com). The main area is titled "Export Query" and contains the following details:

- Query:** SELECT Id FROM Interview\_\_c WHERE Application\_\_c = 'a02gL000007SrgUQAS'
- Options:** Templates, Query History, Clear, Saved Queries, Query Label, Save Query, Deleted/Archived Records?, Tooling API?
- Buttons:** Run Export, Export Query, Query Plan, Help

Below the query editor is the "Export Result" section, which shows the following information:

- Export Result:** Copy (Excel), Copy (CSV), Copy (JSON), Download, Delete Records, Filter Result, Exported 1 record in 410.6ms.
- Table:** A table with one row showing the result of the query. The columns are labeled "Id" and "Interview\_\_c". The data row is: a03gL00000D9v7dQAB.

## 2. Prevent Duplicate Records

1. Setup → **Duplicate Rules** → New Rule.
2. Select Object: **Candidate\_\_c**.
3. Matching Rule: Email\_\_c must be unique.
4. Action: **Block** or **Allow but Alert**.
5. Activate Rule.

6. Test by entering a duplicate Candidate email.

The screenshot shows the Salesforce Setup interface. The left sidebar has a search bar with 'duplic' typed in. Under 'Data', the 'Duplicate Management' section is expanded, showing 'Duplicate Error Logs', 'Duplicate Rules' (which is selected and highlighted in blue), and 'Matching Rules'. A message at the bottom of the sidebar says 'Didn't find what you're looking for? Try using Global Search.' The main content area is titled 'Duplicate Rules' and shows 'Candidate Duplicate Rule Email'. It displays the 'Duplicate Rule Detail' for an 'Email' rule. The rule details include:

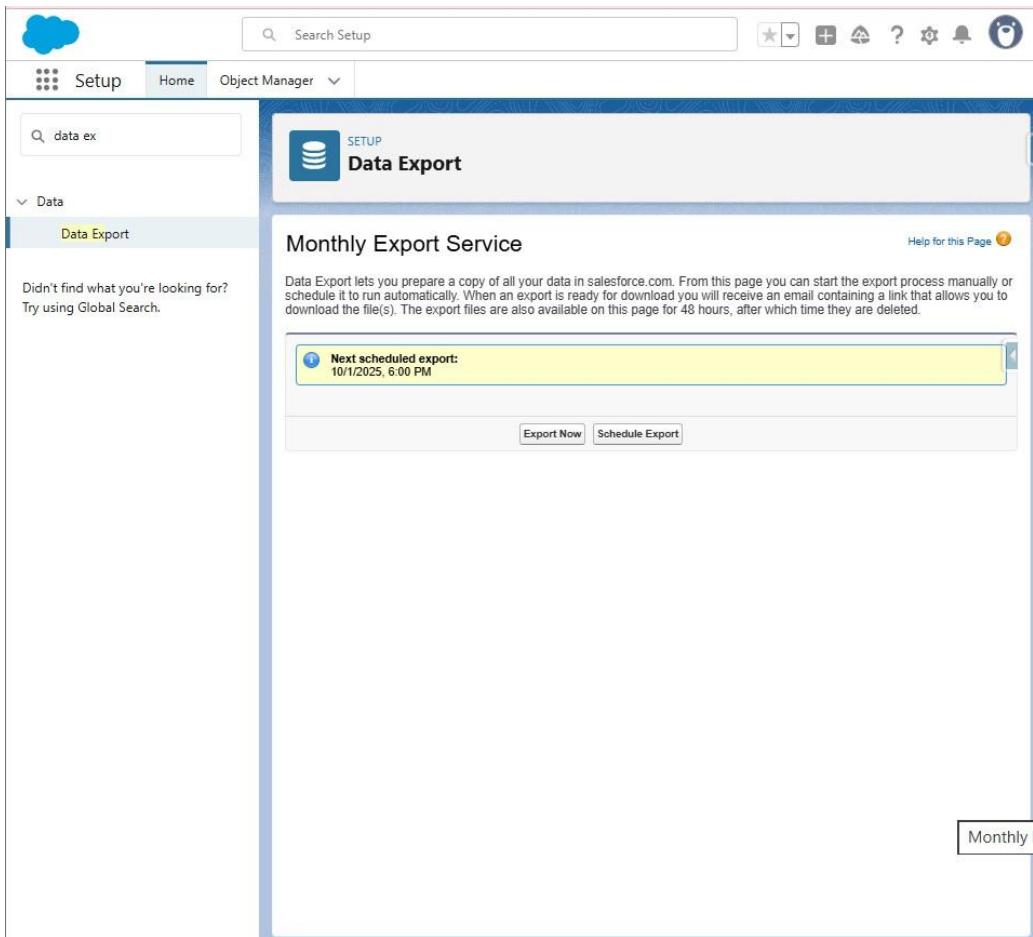
Rule Name	Email	Order
Description		2 of 2 [ Reorder ]
Object	Candidate	
Record-Level Security	Bypass sharing rules	
Action On Create	Allow	Operations On Create ✓ Alert ✓ Report
Action On Edit	Allow	Operations On Edit ✓ Alert ✓ Report
Alert Text	Use one of these records?	
Active	✓	
Matching Rule	Rule 1 Mapped	Matching Criteria (Candidate: Email EXACT MatchBlank = FALSE) AND (Candidate: Phone EXACT MatchBlank = FALSE)
Conditions		
Created By	Tammiseti Venkatarao, 9/26/2025, 5:51 AM	Modified By Tammiseti Venkatarao, 9/26/2025, 5:53 AM

At the bottom of the detail view are 'Edit', 'Delete', 'Clone', and 'Deactivate' buttons.

### 3. Data Export & Backup

1. Setup → Quick Find → **Data Export**.
2. Click **Schedule Export**.
3. Choose **Weekly Export** → Select all objects (Candidates, Applications, Job Openings).
4. Include attachments/files.
5. Salesforce generates a downloadable zip file with CSV backups.

6. Store securely in company's backup drive/cloud.



## 4. Deployment Options

### a) Change Sets (Admin-Friendly)

1. In Sandbox → Setup → **Outbound Change Sets** → New.
2. Add: Objects, Fields, Flows, Apex Classes, Reports.
3. Upload to Production.
4. In Production → Inbound Change Sets → **Validate** → **Deploy**.

### b) Unmanaged vs Managed Packages

- **Unmanaged:** Use for internal HR CRM deployments (editable, good for dev/test).
- **Managed:** Use if you plan to distribute HR Recruitment CRM externally.

- Setup → **Packages** → New → Add components → Upload.

### c) ANT Migration Tool (Advanced Admins)

1. Install Apache ANT.
2. Create build.xml file with login details.
3. Use retrieve to fetch Sandbox metadata.
4. Use deploy to push into Production.
5. Run:
6. ant deployCode

### **Deliverables**

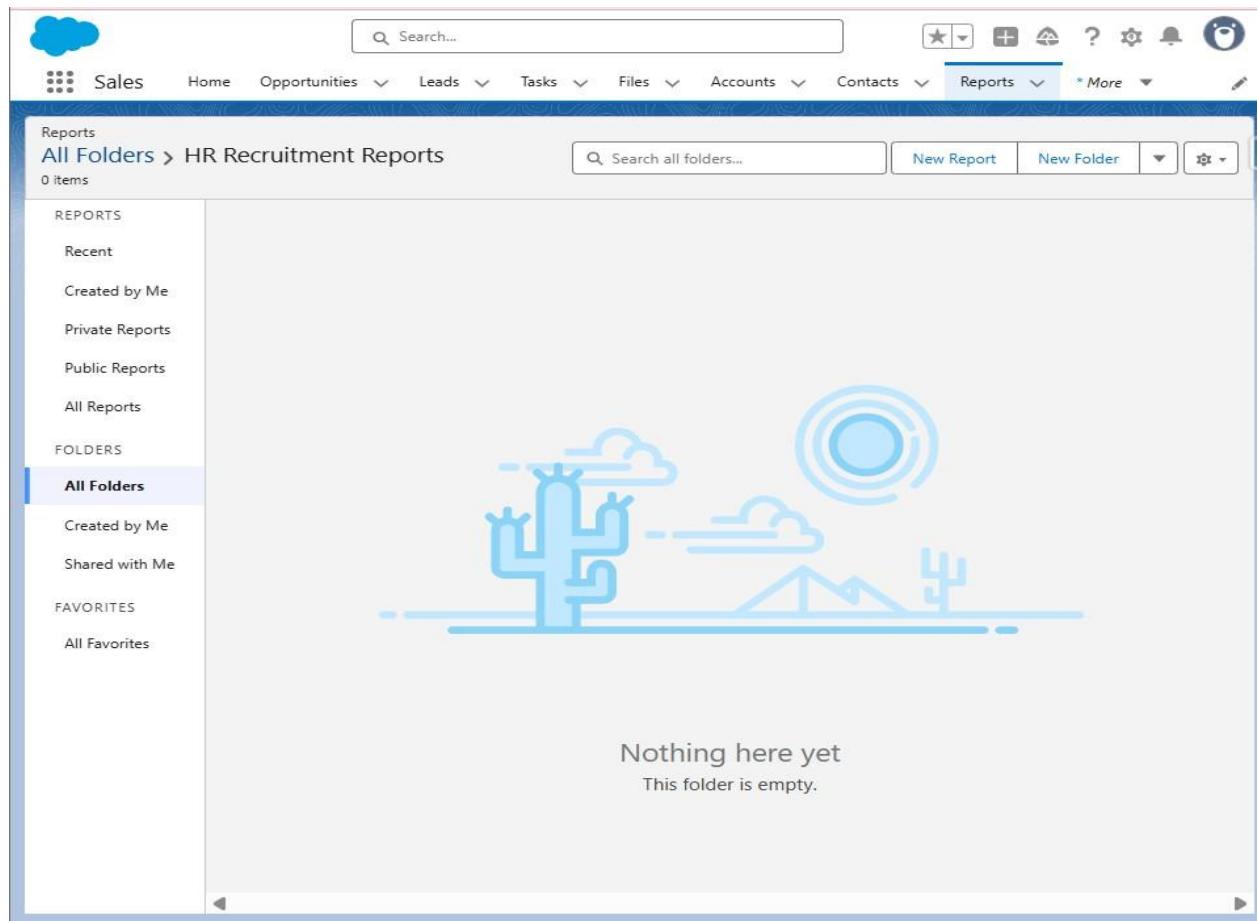
- CSV import templates for Candidates, Applications, Job Openings.
- Active Duplicate Rule for Candidate emails.
- Backup schedule screenshot.
- Change Set deployment record.
- VS Code project repo for Apex/Flows.

## Phase 9: Reports & Dashboards & Security Review

**Goal:** Provide actionable insights on the recruitment pipeline, recruiter productivity, and time-to-hire metrics.

### 1. Create Report Folders

1. Click App Launcher → Reports.
2. Click New Folder.
3. Enter **Folder Name:** HR Recruitment Reports.
4. Click **Share** → Select **HR Managers** and **Recruiters** → Grant **View and Edit** access.



## **2. Build Key Reports**

### **a) Open Positions by Department**

1. Click **New Report** → Select **Job Openings** report type.
2. Add columns: **Job Title**, **Department\_c**, **Status\_c**.
3. Group by **Department\_c**.
4. Add filter: **Status\_c** = Open.
5. Click **Save & Run** → Name: Open Positions by Department.

### **b) Candidate Pipeline by Stage**

1. Click **New Report** → Select **Applications** report type.
2. Group rows by **Stage\_c**.
3. Add filter: **Application\_Status\_c** = Active.
4. Click **Add Chart** → Chart Type: Donut → Show Values: Count of Applications.
5. Save as Candidate Pipeline by Stage.

### **c) Time-to-Hire Report**

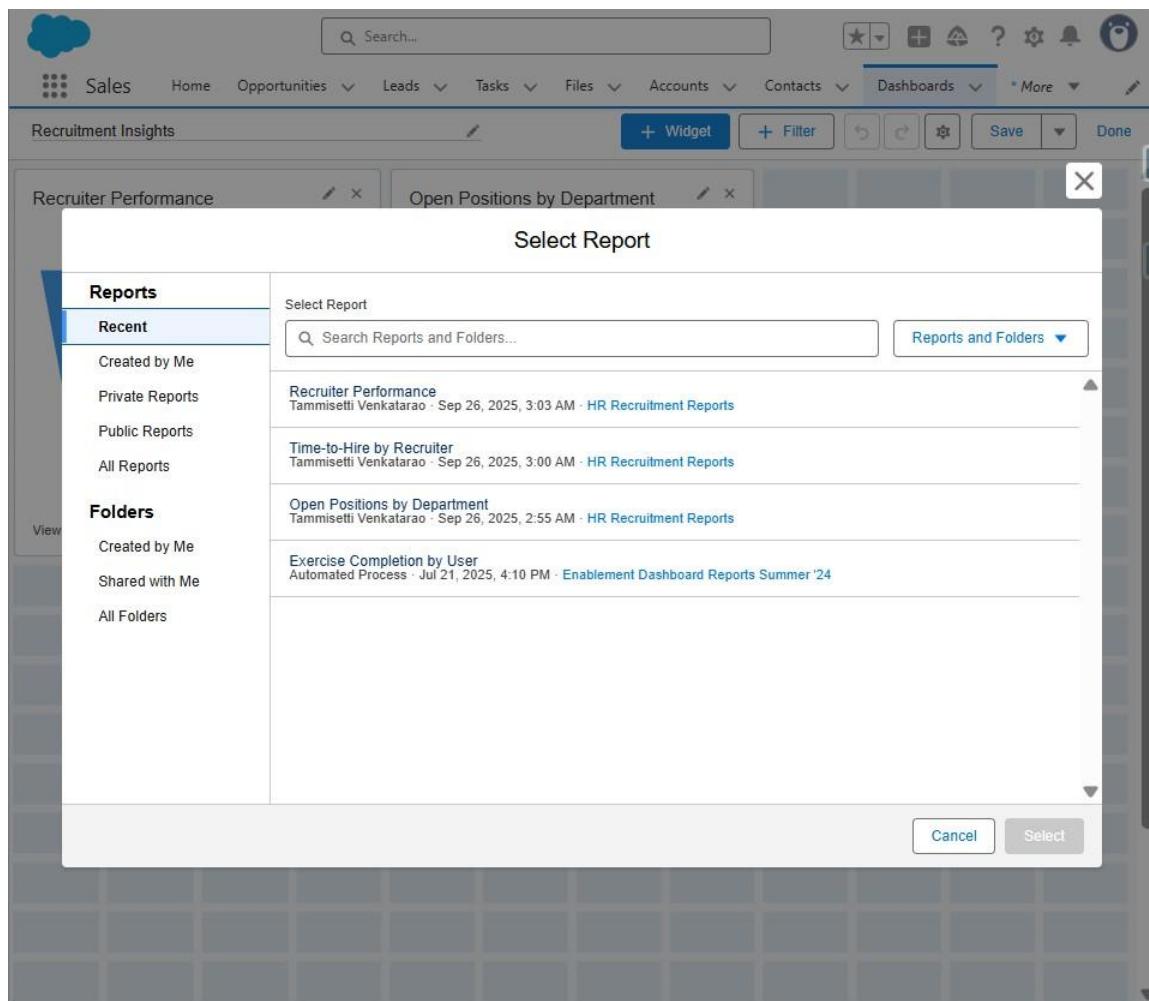
1. Ensure you have a formula field on Application: **Time\_to\_Hire\_c** = **Hired\_Date\_c** - **Application\_Date\_c**.
2. Create **New Report** → Applications.
3. Add **Time\_to\_Hire\_c** and **Recruiter\_c**.
4. Summarize **Average Time\_to\_Hire\_c**.
5. Save as Time-to-Hire by Recruiter.

### **d) Recruiter Performance**

1. Create **New Report** → Applications.
2. Group rows by **Owner**.
3. Add filters: Close Date = Current Fiscal Quarter.
4. Add Chart: Bar Chart → X-Axis: Recruiter → Y-Axis: Applications Closed.
5. Save as Recruiter Performance.

### 3. Create Custom Report Types

1. Setup → **Report Types** → New.
2. Primary Object: Application.
3. Related Object: Candidate.
4. Include Candidate fields: Email\_\_c, Source\_\_c.
5. Save and deploy.



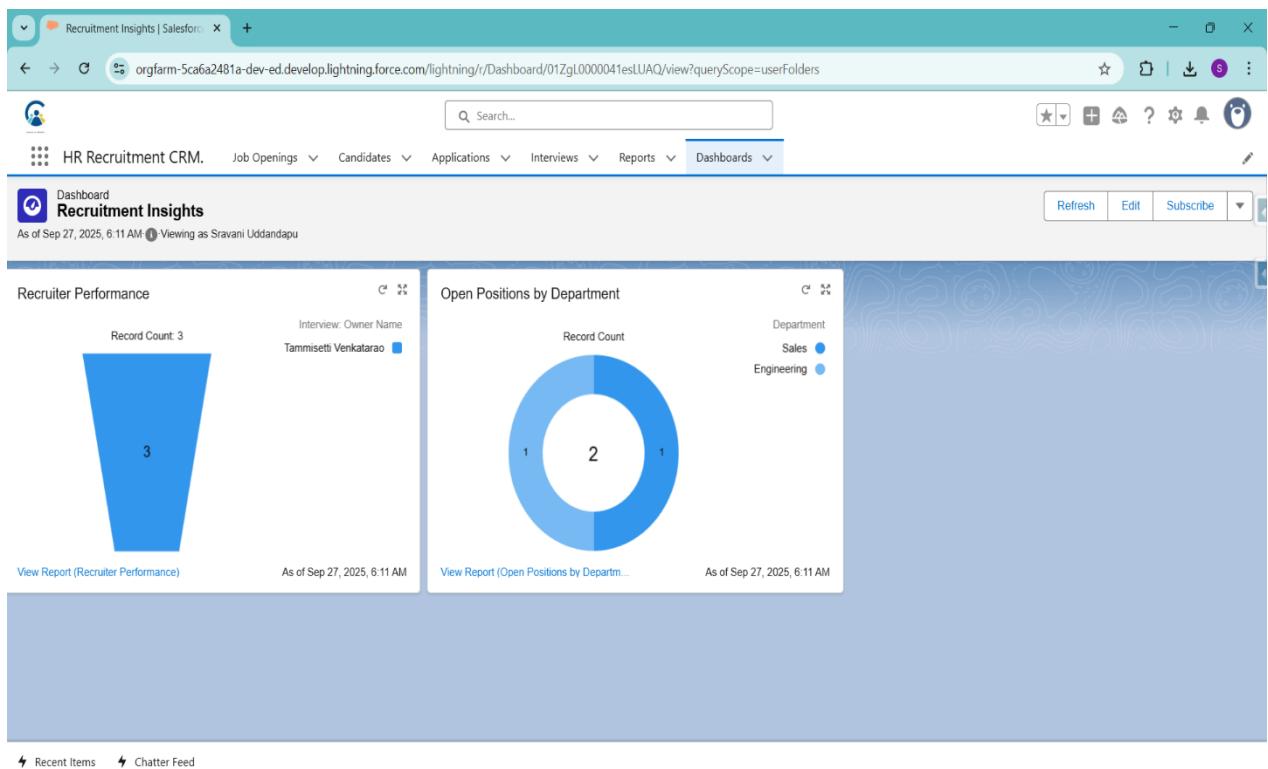
### 4. Build Dashboards

1. App Launcher → **Dashboards** → New Dashboard.
2. Name: Recruitment Insights.

3. Add components:

- o **Donut Chart** → Candidate Pipeline by Stage.
- o **Bar Chart** → Open Positions by Department.
- o **Gauge** → Average Time-to-Hire.
- o **Table** → Recruiter Performance.

4. Save and **Activate** the dashboard.



# Security & Access Control

**Goal:** Secure candidate and hiring data, define user access according to roles, and enforce compliance with company security policies.

## 1. Review & Refine Profiles

1. Click **Setup** (gear icon) → **Setup**.
2. In Quick Find, type **Profiles** → Click **Profiles**.
3. Edit each profile:
  - o **Recruiter**
    - Object Permissions: **Read/Write** on Candidate\_c and Application\_c.
    - **Read Only** on Job\_Opening\_c.
  - o **HR Manager**
    - Full access to all HR-related objects.
  - o **Department Head**
    - **Read/Edit** on Job\_Opening\_c and Interview\_c.

The screenshot shows the Salesforce Setup interface with the 'Profiles' tab selected. The main content area displays the 'HR Manager' profile details. The profile name is 'HR Manager', user license is 'Salesforce', and it is a 'Custom Profile'. The 'Profile Detail' section shows the creation date as '9/22/2025, 9:25 PM' and the last modified by 'Tammisetti Venkatarao' at '9/25/2025, 11:58 PM'. The 'Page Layouts' section lists various standard object layouts assigned to the profile, such as 'Email Application', 'Home Page Layout', 'Account', 'Alternative Payment Method', 'Appointment Invitation', 'Asset', 'Asset Action', 'Asset Action Source', 'Asset Relationship', and 'Asset State Period'. Each layout is associated with specific global or location group layouts and assignment types like 'Macro', 'Object Milestone', 'Operating Hours', 'Opportunity Product', 'Order', 'Order Product', 'Payment', and 'Payment Authorization'.

## 2. Create Permission Sets for Sensitive Fields

1. Setup → **Permission Sets** → New.
2. Label: CandidateSensitiveAccess.
3. Object Settings → **Candidate\_c** → Enable visibility for **Salary\_Expectation\_c** and **Offer\_Details\_c**.
4. Click **Manage Assignments** → Assign only to **HR Manager** users.

The screenshot shows the Salesforce Setup interface. On the left, there's a sidebar with a search bar and sections for 'Users', 'Permission Set Groups', and 'Custom Code'. The main area is titled 'Permission Sets' and shows a list of existing permission sets. A new permission set named 'CandidateSensitiveAccess' is being created. The 'Object Settings' tab is selected, displaying a table of objects and their permissions. One row in the table is for 'Activation Platform Activation Attributes', which has 'No Access' listed under 'Object Permissions'.

## 3. Configure Field-Level Security (FLS)

1. Setup → **Object Manager** → Select **Candidate\_c**.
2. Click **Fields & Relationships** → Choose **Salary\_Expectation\_c**.
3. Click **Set Field-Level Security**.
4. Uncheck **Recruiter** visibility → Save.

The screenshot shows the Salesforce Setup interface with the search bar set to "profiles". A modal window titled "SETUP" is open, showing the "Field-Level Security" configuration for the "Candidate Name" field. The table lists various profiles along with their "Visible" and "Read-Only" status for the specified field.

Field Label	Candidate Name	Visible	Read-Only
Data Type	Text(50)		
Analytics Cloud Integration User	✓	□	
Analytics Cloud Security User	✓	□	
Anypoint Integration	✓	□	
Contract Manager	✓	□	
Cross Org Data Proxy User	✓	□	
Custom: Marketing Profile	✓	□	
Custom: Sales Profile	✓	□	
Custom: Support Profile	✓	□	
Department Head	✓	□	
Einstein Agent User	✓	□	
Force.com - App Subscription User	✓	□	
Force.com - Free User	✓	□	
Gold Partner User	✓	□	
HR Manager	✓	□	
Identity User	✓	□	
Marketing User	✓	□	
Minimum Access - API Only Integrations	✓	□	
Minimum Access - Salesforce	✓	□	
Partner App Subscription User	✓	□	
Partner Community Login User	✓	□	

## 4. Set Organization-Wide Defaults (OWD)

1. Setup → Sharing Settings → Click Edit.
2. Configure:
  - Job Openings → Private
  - Candidates → Private
  - Interviews → Controlled by Parent
3. Click Save.

The screenshot shows the Salesforce Setup interface with the search bar set to "sharing". A modal window titled "SETUP" is open, showing the "Sharing Settings" configuration for "Organization-Wide Sharing Defaults Edit". The table lists various objects and their sharing settings for internal access, external access, and grant access using hierarchies.

Object	Default Internal Access	Default External Access	Grant Access Using Hierarchies
Lead	Public Read/Write/Transfer	Private	Private
Account and Contract	Public Read/Write	Public	Public
Order	Controlled by Parent	Controlled by Parent	Controlled by Parent
Contact	Controlled by Parent	Controlled by Parent	Controlled by Parent
Asset	Controlled by Parent	Controlled by Parent	Controlled by Parent
Opportunity	Public Read/Write	Private	Private
Case	Public Read/Write/Transfer	Private	Private
Campaign	Public Full Access	Private	Private
Campaign Member	Controlled by Campaign	Controlled by Campaign	Controlled by Campaign
User	Public Read Only	Private	Private
Individual	Public Read/Write	Private	Private
Voice Call	Private	Private	Private
Activity	Private	Private	Private
Calendar	Hide Details and Add Events	Use	Use
Price Book	Public	Public Read/Write	Public Read/Write
Product	Public Read/Write	Private	Private
Agent Work	Public Read Only	Private	Private
Alternative Payment Method	Private	Private	Private
Analytics User Attribute Function	Public Read Only	Private	Private
Tab	Private	Private	Private
Appointment Invitation	Private	Private	Private
Approval Submission	Private	Private	Private
Authorization Form	Private	Private	Private

## 5. Create Sharing Rules

1. Setup → Sharing Settings → Scroll to **Sharing Rules** → Click **New**.

### 2. Rule 1: Recruiter\_to\_HRManager

- Object: Candidate
- Criteria: All records owned by Recruiters
- Access: Read/Write for HR Managers

### 3. Rule 2: JobOpenings\_to\_DepartmentHead

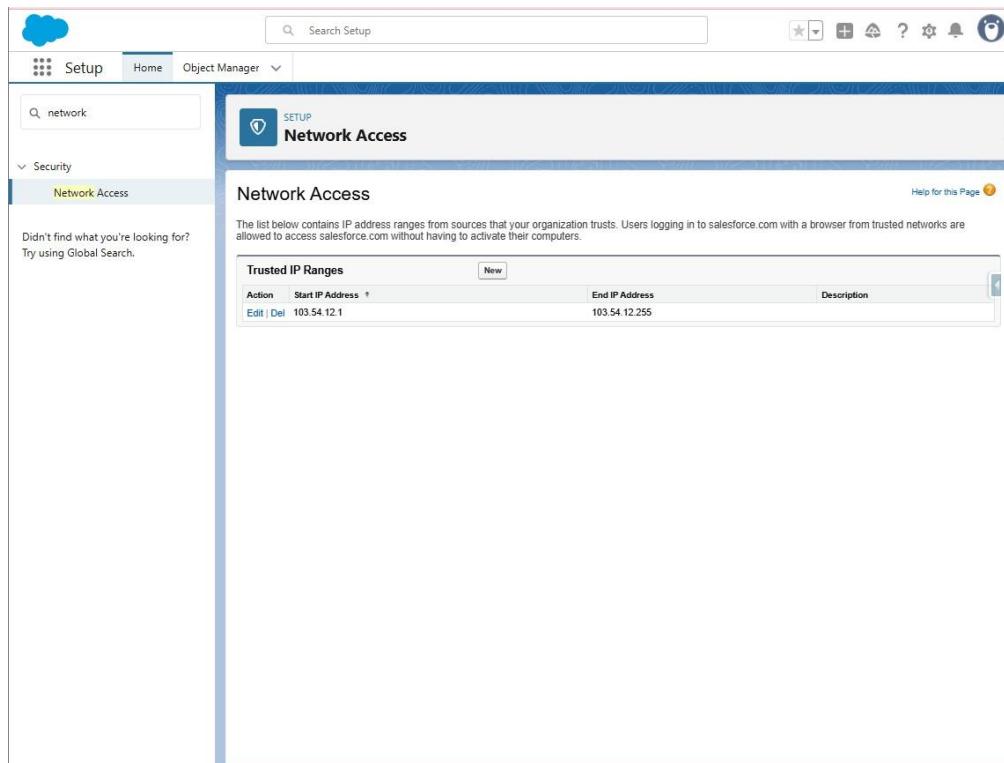
- Object: Job Opening
- Criteria: All Job Openings
- Access: Read Only for Department Heads

4. Click **Save**.

The screenshot shows the Salesforce Sharing Settings page. At the top, there's a search bar with 'sharing sett' and a 'Search Setup' button. Below the search bar, the 'Setup' tab is selected in the navigation bar, along with 'Home' and 'Object Manager'. On the left, a sidebar under 'Security' has 'Sharing Settings' selected. A message says 'Didn't find what you're looking for? Try using Global Search.' The main content area is titled 'Sharing Rules' and contains sections for various objects: Lead Sharing Rules, Account Sharing Rules, Opportunity Sharing Rules, Case Sharing Rules, Campaign Sharing Rules, User Sharing Rules, Individual Sharing Rules, Voice Call Sharing Rules, and Activation Target Sharing Rules. Each section includes a 'New' button and a 'Recalculate' button. Help links are provided for each section.

## 6. Configure Login IP Ranges & Session Settings

1. Setup → Network Access → Click New → Add your company's IP ranges (e.g., Start IP: 103.54.12.1, End IP: 103.54.12.255).
2. Setup → Session Settings:
  - Enable Multi-Factor Authentication (MFA).
  - Set Session Timeout (e.g., 2 hours).
  - Save changes.



## 7. Test Access

1. Log in as **Recruiter**:
  - Confirm they cannot see salary fields.
  - Verify they can create and update Candidates and Applications.
2. Log in as **HR Manager**:
  - Confirm full access, including salary and offer details.

3. Log in as **Department Head**:

- o Confirm read-only access to Job Openings.

**Deliverables**

- Screenshots of Profiles, Permission Sets, and OWD configurations.
- Table summarizing Sharing Rules and FLS settings.

Evidence of testing different user roles.

## PHASE 10:

### Testing Results:

Test Case ID	Test Description	Input/Setup	Expected Result	Actual Result	Status
TC_01	Objects and Fields creation	Objects and fields created and visible under their respective objects	Objects and fields created and visible under their respective objects	Sucessfully created All fields	Passed
TC_02	Filter the Applications Who are reached the Interviewing stage	Records with eligible Applications	Only eligible Candidates fetched	Correct list returned	Passed
TC_03	Setting FLS	FLS must be applied for Required fields	FLS applied Correctly	FLS applied Successfully	Passed
TC_04	Tick Checkbox if Resume attached	Checkbox have tick If resume attached	Tick mark	Check box have tick Mark if resume attached	Passed
TC_05	Number of Hired Applications Automatically filled	Automatically filled if Number of applications Reach number of openings	Automatically filled if Number of applications Reach number of openings	Succesfully returned the value	Passed

## **Test Environment**

- **Salesforce Org Type:** Developer Edition
- **Data Used:** 5 sample records
- **Test Tools:** Flow Debug, Object Record View

## **Project Demo**

We can access the demo video here

(<https://drive.google.com/file/d/1BgrcfV6KiFLBSGFToWsGSLame53RvKTJ/view?usp=drivesdk>)