

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	adm1	admin2690@gmail.com	<input checked="" type="checkbox"/>	✓	Standard Platform User
<input type="checkbox"/>	Chatter Event	shahn_0000000007evbuvay.c811cdvhllo@chatter.salesforce.com		<input checked="" type="checkbox"/>	✓	Chatter Free User
<input type="checkbox"/>	Department Head	desa	department2690@gmail.com	<input checked="" type="checkbox"/>	✓	Chatter Free User
<input type="checkbox"/>	EPIC_OrgFarm	GEPI	gepi.56d94544c28d@orgfarm.salesforce.com	<input checked="" type="checkbox"/>	✓	System Administrator
<input type="checkbox"/>	HR Manager	hr.m	hr2690@gmail.com	<input checked="" type="checkbox"/>	✓	Standard Platform User
<input type="checkbox"/>	Recruiter	rscr	recruiter2690@gmail.com	<input checked="" type="checkbox"/>	✓	Standard Platform User
<input type="checkbox"/>	User_Intervention	inteo	intervention@009v000007evbuvay.com	<input checked="" type="checkbox"/>	✓	Analytics Cloud Intervention User
<input type="checkbox"/>	User_Jessith	ies	intervention@009v000007evbuvay.com	<input checked="" type="checkbox"/>	✓	Analytics Cloud Security User
<input type="checkbox"/>	Venkatesh_Tammireddy	tha	thammirekka@5743@sapientforce.com	<input checked="" type="checkbox"/>	✓	System Administrator

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 the 'prof' profile. The main content area displays the 'HR Manager' profile details:

- Profile Detail:** Name: HR Manager, User License: Salesforce, Description: None, Created By: TammieSelli.Venkatalarao, Modified By: TammieSelli.Venkatalarao.
- Page Layouts:** Standard Object Layouts table showing assignments for various objects like Global, Email Application, Home Page Layout, Account, and Alternative Payment Method.

The screenshot shows the Salesforce Setup interface under the Profiles section. A search bar at the top finds the 'prof' profile. The main content area displays the 'Recruiter Profile' profile details:

- Profile Detail:** Name: Recruiter Profile, User License: Salesforce, Description: None, Created By: TammieSelli.Venkatalarao, Modified By: TammieSelli.Venkatalarao.
- Page Layouts:** Standard Object Layouts table showing assignments for various objects like Global, Email Application, Home Page Layout, Account, and Alternative Payment Method.

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 'Sharing Settings' 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 checkmark column indicates if specific sharing is defined. The table includes rows for Shipment, Shipping Carrier, Shipping Carrier Method, Shipping Configuration Set, Streaming Channel, Tableau Host Mapping, User Presence, User Provisioning Request, Watchlist, Web Cart Document, Work Order, Work Plan, Work Plan Template, Work Step Template, Work Type, Work Type Group, Application, Candidate, Interview, and Job Opening. The 'Candidate' object is listed as 'Controlled by Parent' in both the Default and Specific sharing columns.

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	✓
Watchlist	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 help. Below the navigation is a header for 'Object Manager' with a 'SETUP' button, a search bar ('job'), a 'Schema Builder' button, and a 'Create' button. A message says '1 Items, Sorted by Label'. The main area is a table with the following columns: LABEL, API NAME, TYPE, DESCRIPTION, LAST MODIFIED, and DEPLOYED. One row is shown: 'Job Opening' (Label), 'Job_Opening__c' (API Name), 'Custom Object' (Type), 'Custom Object' (Description), '9/20/2025' (Last Modified), and a dropdown arrow (Deployed).

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 > OBJECT MANAGER.
- Section title: Interview.
- 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
- Fields & Relationships** table (9 items, Sorted by Field Label):

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 for the 'Application' object. On the left, a sidebar lists various setup options under 'Fields & Relationships'. The main panel displays the 'Custom Field Definition Detail' for a field named 'Candidate'. The field is defined as a Master-Detail type, related to the 'Candidate' object. It has a description and a sharing setting. A validation rule is listed at the bottom. The right side of the screen shows the 'Field Information' and 'Master-Detail Options' sections.

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.

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

- 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
- Conditional Field Formatting

Applications ▾

Save ▾ Quick Save Preview As... ▾ Cancel Undo Redo Layout Properties

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

Highlights Panel
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.

Job Opening Detail Standard Buttons

SETUP > OBJECT MANAGER

Job Opening

- 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
- Conditional Field Formatting

Job Opening Compact Layout

Job Title

◀ Back to Job Opening

Compact Layout Detail

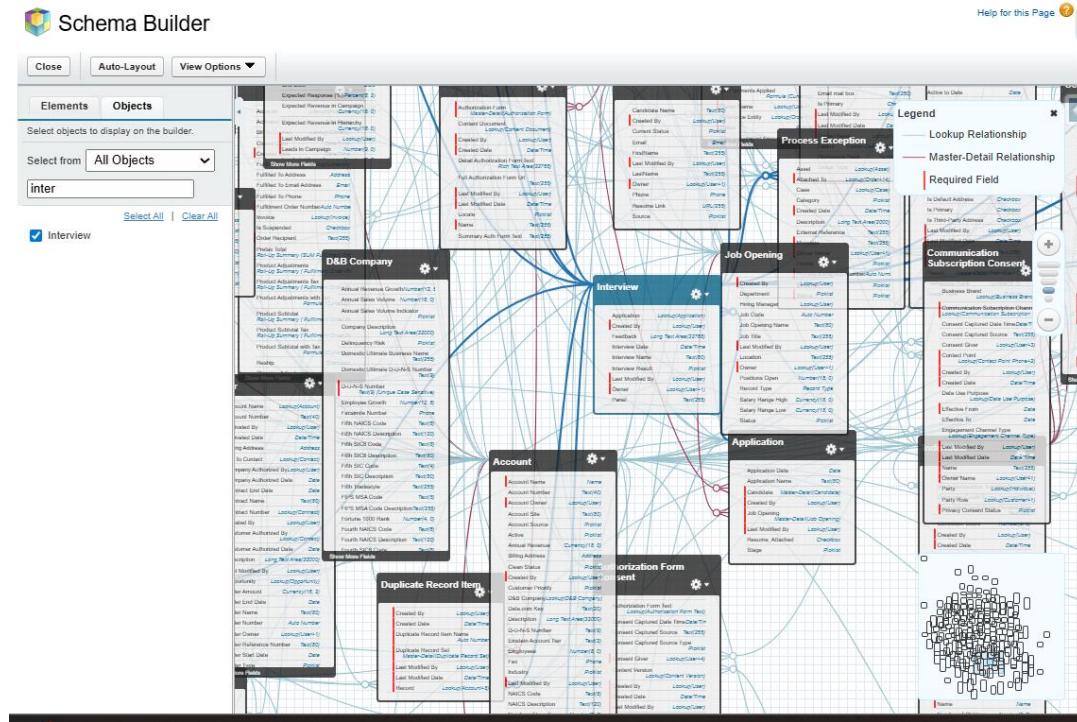
Label	Job Title	Object Name
API Name	Job_Title	Job_Opening
Included Fields	Job Title Status Hiring Manager	
Created By	Tammiseti Venkatarao, 9/20/2025, 8:53 AM	Modified By Tammiseti Venkatarao, 9/20/2025, 8:53 AM

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.

Custom Tabs

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

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

Action	Label	Tab Style	Description
Edit Del	Applications	Airplane	
Edit Del	Candidates	Airplane	
Edit Del	Interviews	Bank	
Edit Del	Job Openings	Airplane	This tab is Already Created. I won't Create Now.

Web Tabs

No Web Tabs have been defined

Visualforce Tabs

No Visualforce Tabs have been defined

Lightning Component Tabs

No Lightning component tabs have been defined

Lightning Page Tabs

No Lightning Page Tabs have been defined

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 for the Candidate object. On the left, there is a sidebar with various configuration options like Details, Fields & Relationships, Page Layouts, etc. The 'Validation Rules' option is selected. The main area displays a table titled 'Validation Rules' with one item listed:

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.	✓	Tammisetti 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 for Duplicate Rules. On the left, the navigation sidebar is open under 'Data' with 'Duplicate Rules' selected. The main content area displays 'Rule 2' for a 'Candidate' object. The 'Duplicate Rule Detail' section includes fields for Rule Name (Rule 2), Description (Object: Candidate, Record-Level Security: Enforce sharing rules), and Action On Create/Edit (Allow). It also shows Operations On Create/Edit (Alert checked, Report checked) and Alert Text (Use one of these records?). The 'Matching Rule' section lists 'Rule 1' and 'Mapped'. The 'Matching Criteria' field contains the formula: (Candidate: Email EXACT MatchBlank = FALSE) AND (Candidate: Phone EXACT MatchBlank = FALSE). The 'Conditions' section shows Created By (Tammiseti Venkatarao, 9/20/2025, 9:22 AM) and Modified By (Tammiseti Venkatarao, 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.

The screenshot shows the Salesforce Setup interface with the 'Bulk Data Load Jobs' page open. The page title is 'Bulk Data Load Jobs' with a sub-section '750gL00000DnuWG'. It displays detailed information about a completed job, including the Job ID (750gL00000DnuWG), Status (Closed), and various performance metrics like Total Processing Time (192 ms) and API Active Processing Time (118 ms). Below this, the 'Batches' section shows a single batch with a View Request link, a View Result link, and other processing details.

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.

Job Opening Custom Field
Salary Range High

[Back to Job Opening](#) [Validation Rules](#)

Custom Field Definition Detail [Edit](#) [Set Field-Level Security](#) [View Field Accessibility](#) [Where is this used?](#)

Field Information	Object Name	Data Type
Field Label: Salary Range High	Object Name: Job_Opening	Data Type: Currency
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: Tammisetti Venkatarao 9/20/2025, 8:09 AM	Modified By: Tammisetti Venkatarao 9/20/2025, 8:09 AM	

General Options
Required
Default Value

Currency Options
Length: 18
Decimal Places: 0

Validation Rules [New](#) [Validation Rules Help](#)

No validation rules defined.

[^ Back To Top](#) Always show me [more](#) records per related list

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 Object Manager selected. The main area displays the 'Fields & Relationships' section for the 'Application' object. The table lists the following fields:

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)

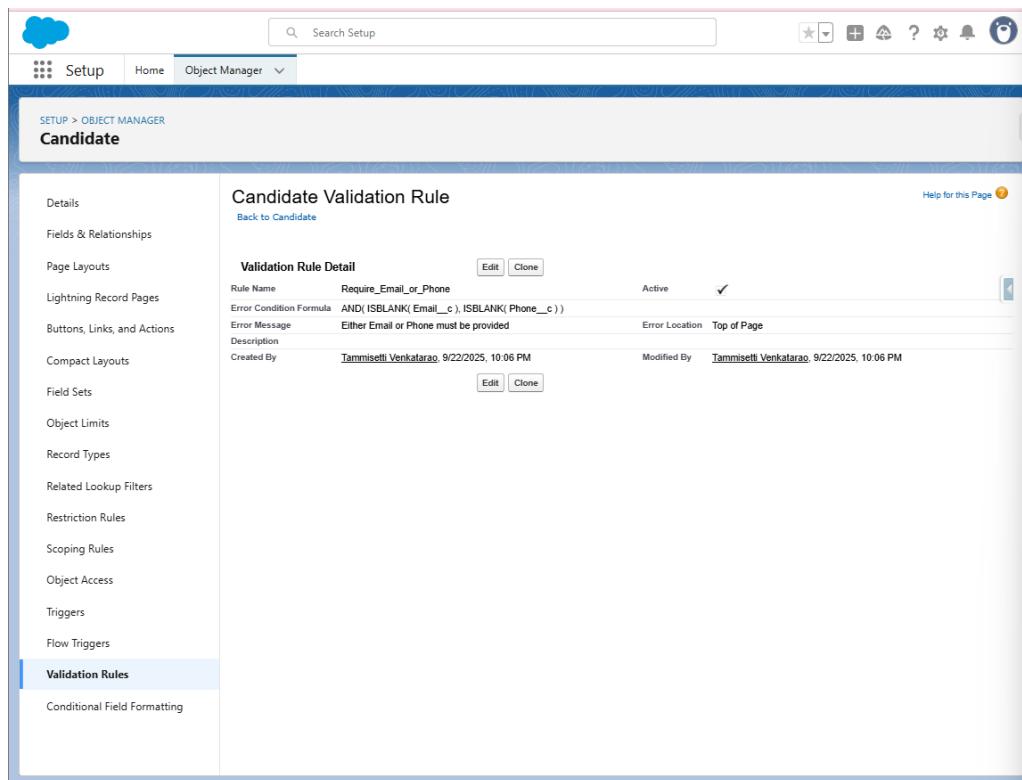
Goal:

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:
`AND(ISBLANK(Email__c), ISBLANK(Phone__c))`
4. Error Message: Either Email or Phone must be provided.
5. Error Location: Top of Page → Save.



The screenshot shows the Salesforce Setup interface for creating a validation rule. The left sidebar lists various object settings like Details, Fields & Relationships, and Validation Rules. The Validation Rules section is currently selected. The main content area displays the 'Candidate Validation Rule' configuration page. The 'Validation Rule Detail' section contains the following information:

Field	Value
Rule Name	Require_Email_or_Phone
Error Condition Formula	AND(ISBLANK(Email__c), ISBLANK(Phone__c))
Error Message	Either Email or Phone must be provided
Description	(empty)
Created By	Tammisetti Venkatarao, 9/22/2025, 10:06 PM
Modified By	Tammisetti Venkatarao, 9/22/2025, 10:06 PM

The 'Active' checkbox is checked. The 'Error Location' is set to 'Top of Page'. The page also includes standard Salesforce navigation and help links.

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):

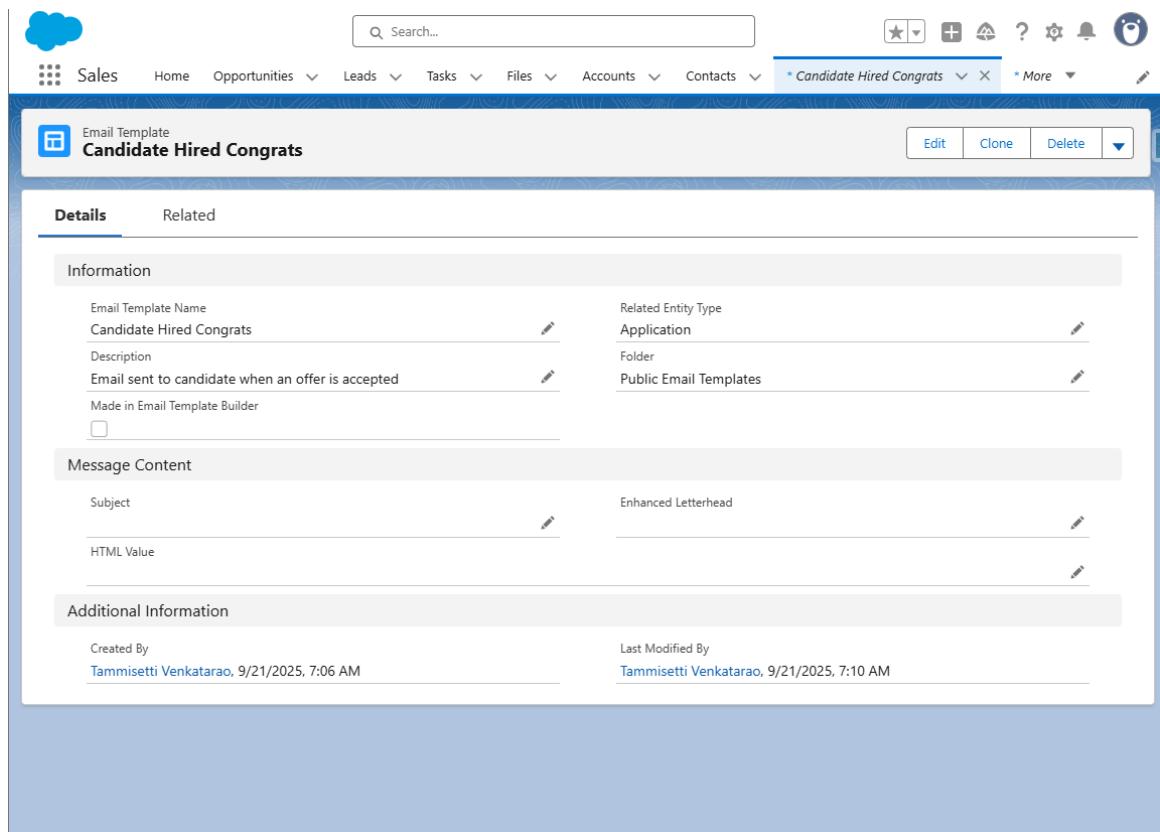
Hi {!Application__c.Candidate__r.FirstName},

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.



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 left sidebar has 'Email al' selected under 'Email Alerts'. The main area displays the 'Email Alerts' page for a specific alert named 'Send Congratulations to Candidate on Hired'. The alert details are as follows:

Description	Send Congratulations to Candidate on Hired	Email Template	Candidate_Hired_Congrats
Unique Name	Send_Congratulation_to_Candidate_on_Hired	Object	Application
From Email Address	Current User's email address		
Recipients			
Additional Emails	tammisetivenkatrao53@gmail.com		
Created By	Tammiseti Venkatarao, 9/21/2025, 7:13 AM	Modified By	Tammiseti Venkatarao, 9/21/2025, 7:13 AM

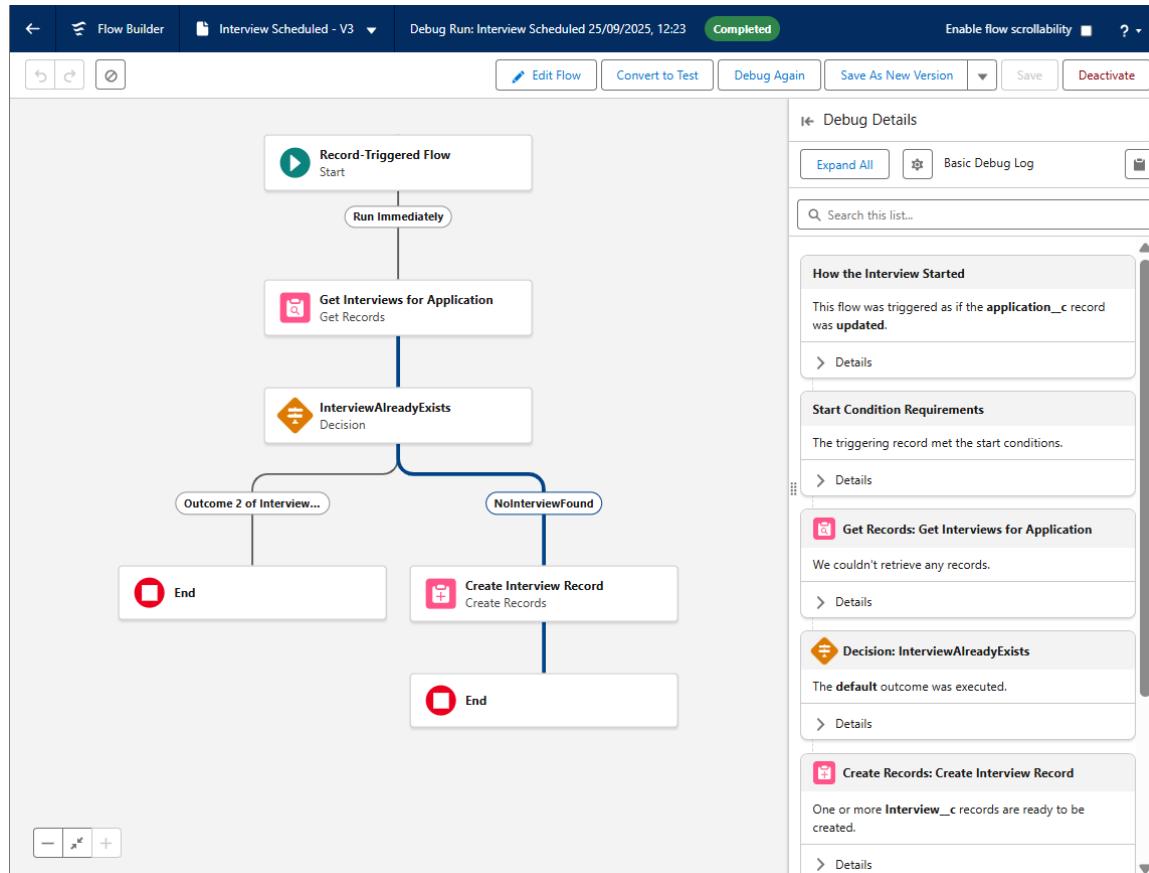
Below the details, there are sections for 'Rules Using This Email Alert', 'Approval Processes Using This Email Alert', 'Entitlement Processes Using This Email Alert', and 'Flows Using This Email Alert'. Each section indicates that no rules, approval processes, or entitlement processes are currently assigned to this alert.

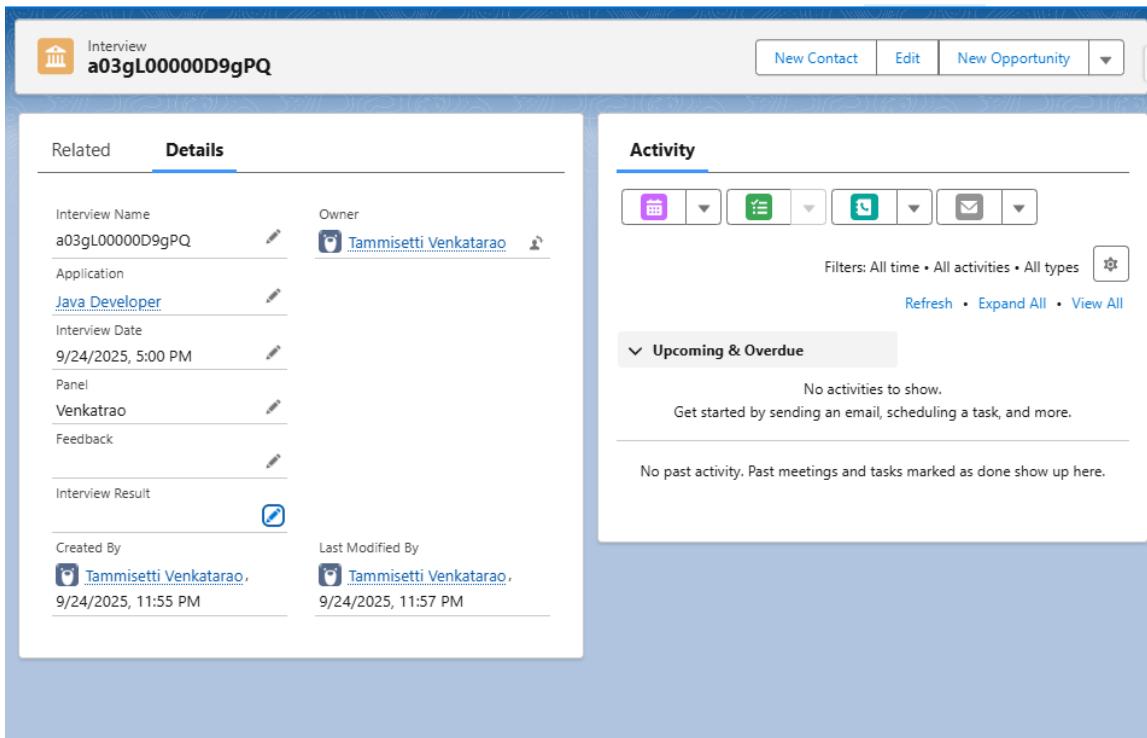
3) Record-Triggered Flow: Auto-create Interview when Stage = Interview Scheduled

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.

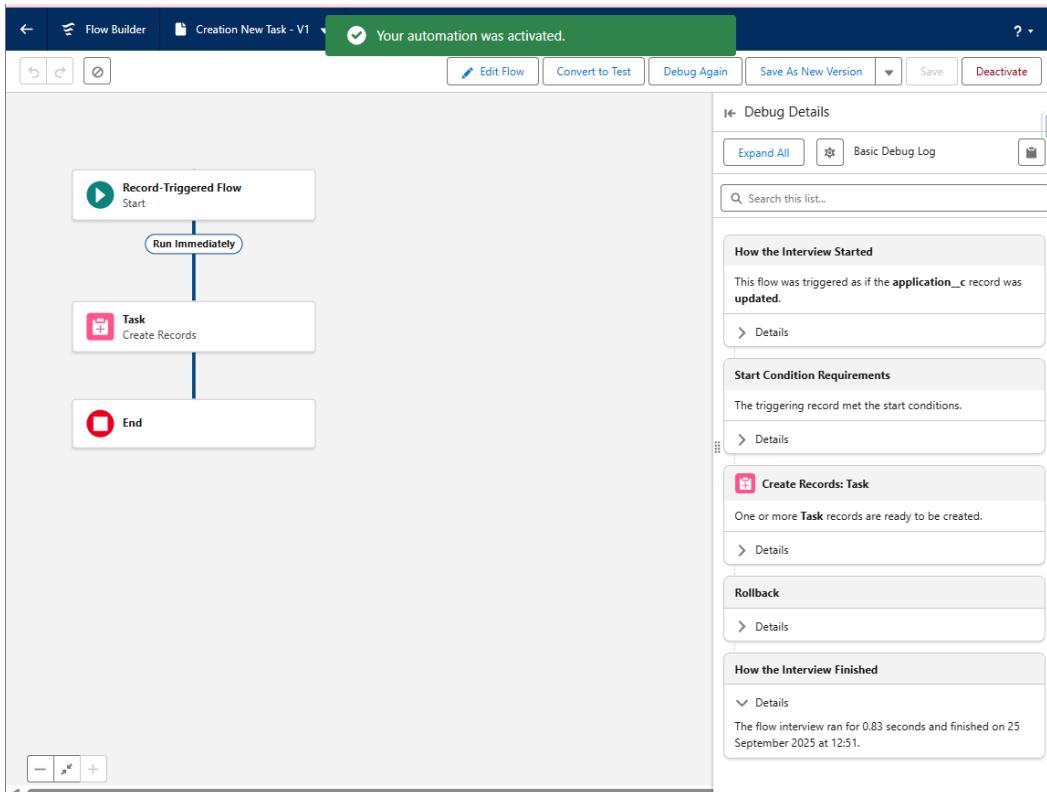




4) Record-Triggered Flow: Auto-create Task for Recruiter on Stage = Screening

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

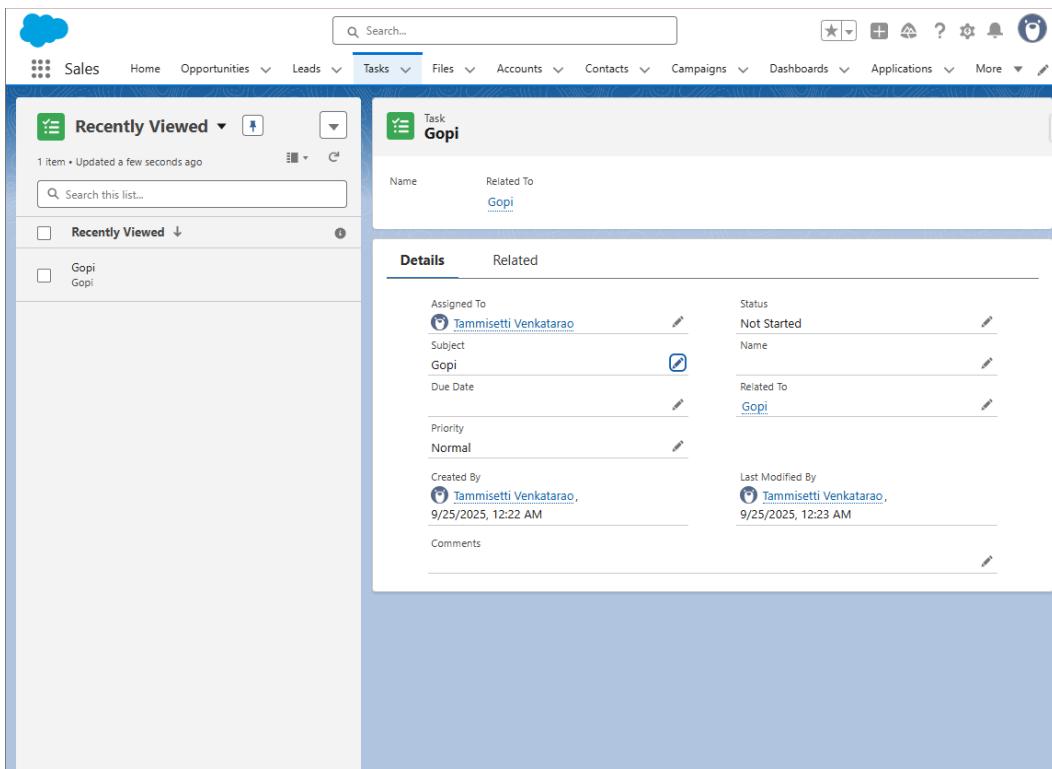
Related Details

Application Name: Salesforce
Candidate: Gopi
Job Opening: SalesForce
Application Date: 9/25/2025
Stage: Screening
Resume_Attached:
Created By: Tammisetti Venkatarao, 9/24/2025, 10:07 AM
Last Modified By: Tammisetti Venkatarao, 9/25/2025, 12:22 AM

Activity

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

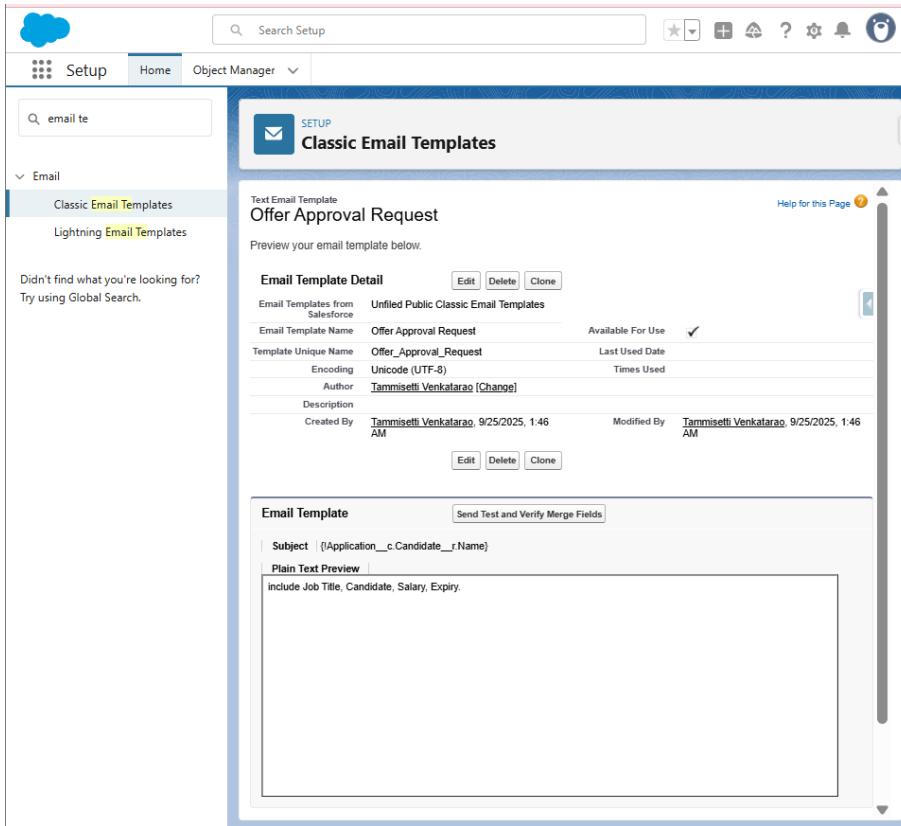
No past activity. Past meetings and tasks marked as done show up here.



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.

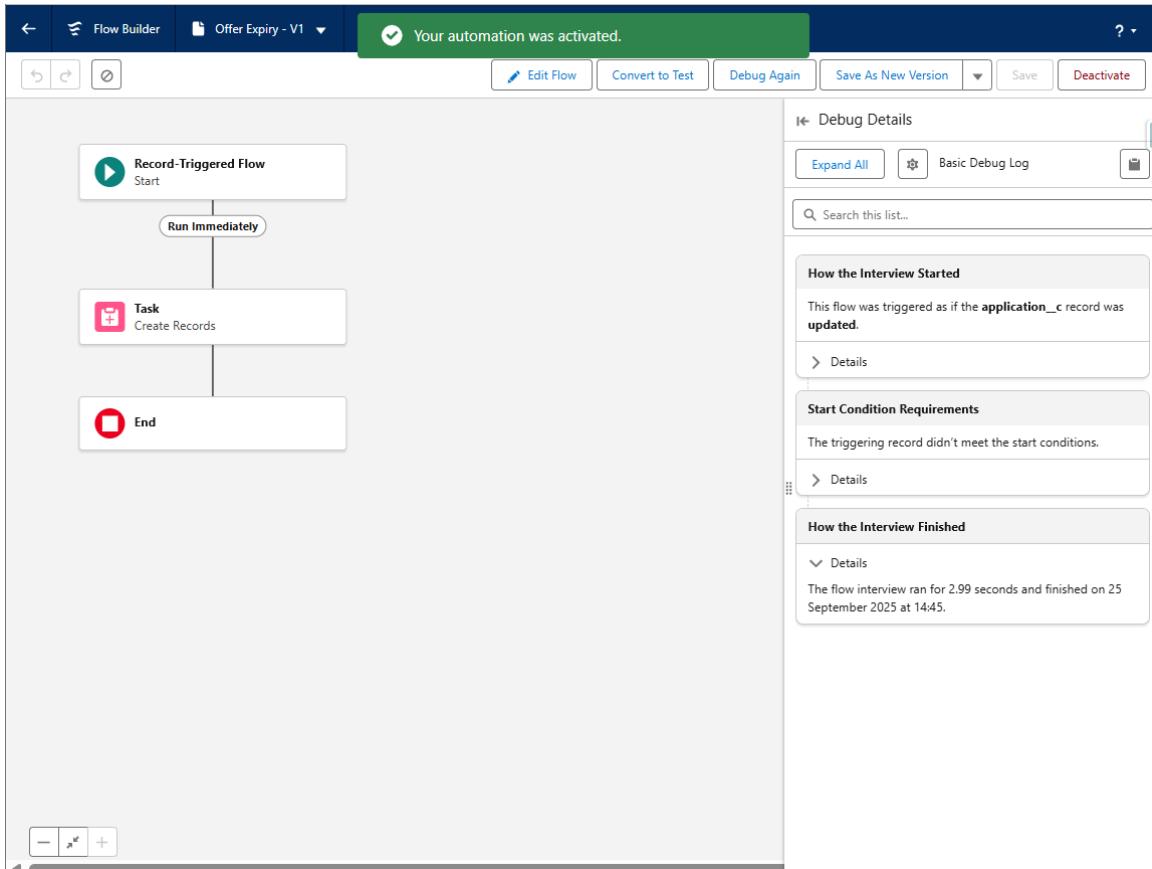
Field	Value
Unique Name	Offer_Approval_Process
Description	
Entry Criteria	ISPICKVAL(Stage__c, "Offer")
Record Editability	Administrator OR Current Approver
Approval Assignment Email Template	Offer Approval Request
Initial Submitters	Candidate Owner
Created By	Tammisetti Venkataraao
Modified By	Tammisetti Venkataraao
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.

Job Opening Custom Field
Number of Hired Applications

[Help for this Page](#)

[Back to Job Opening](#)

Custom Field Definition Detail

Edit	Set Field-Level Security	View Field Accessibility	Where is this used?
----------------------	--	--	-------------------------------------

Field Information

Field Label	Number of Hired Applications	Object Name	Job Opening
Field Name	Number_of_Hired_Applications		
API Name	Number_of_Hired_Applications__c		
Description			
Help Text			
Data Owner			
Field Usage			
Data Sensitivity Level			
Compliance Categorization			
Created By	Tammisetti Venkatarao , 9/25/2025, 2:25 AM	Modified By	Tammisetti Venkatarao , 9/25/2025, 2:25 AM

Roll-Up Summary Options

Data Type	Roll-Up Summary	Summary Type	COUNT
Summarized Object	Application		
Filter Criteria	Stage EQUALS Hired		

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

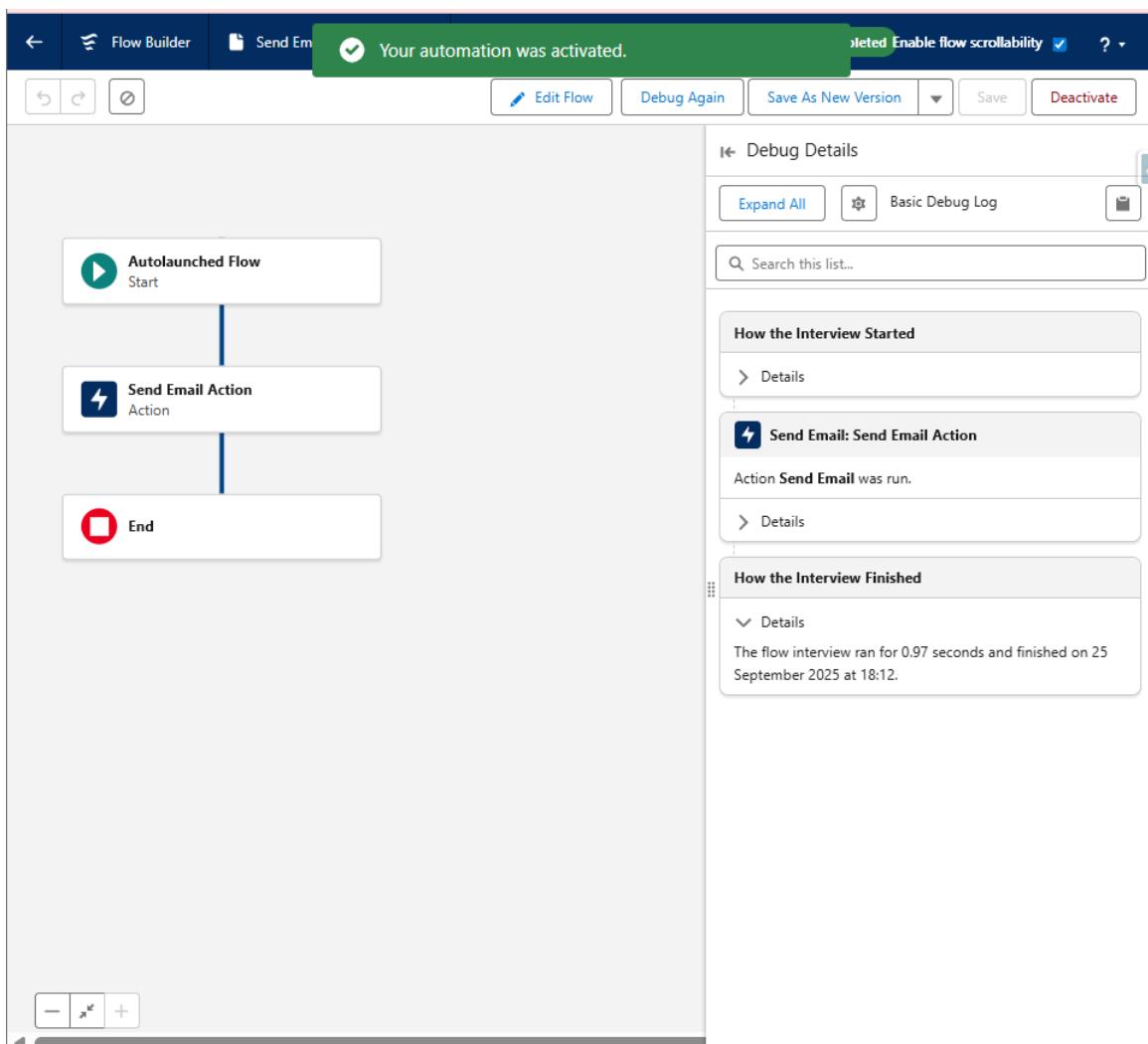
8) Auto-convert Leads to Candidates (incoming from LinkedIn / Site)

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

Apex Trigger ApplicationTrigger

[Help for this Page](#)

[« Back to List](#)

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

```

1trigger 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](#)

- 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

<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()) {

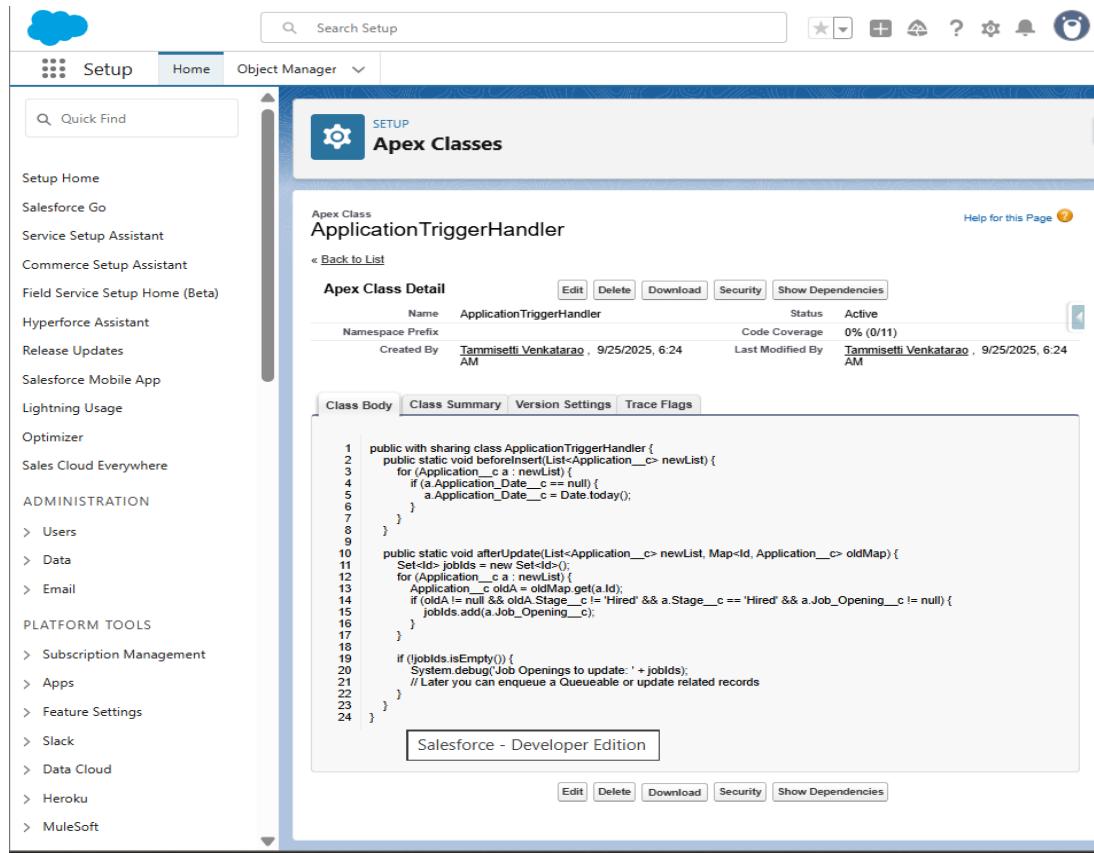
    System.enqueueJob(new UpdateJobOpeningQueueable(jobIds));

}

}

}

```



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());
    }
}
}
}

```

The screenshot shows the Salesforce Setup interface. The left sidebar lists various setup categories like Setup Home, Salesforce Go, Service Setup Assistant, etc. The main content area displays the Apex Class Detail page for 'UpdateJobOpeningQueueable'. The page includes tabs for Apex Class Detail, Class Body, Class Summary, Version Settings, and Trace Flags. The Class Body tab shows the following Apex code:

```

1 public class UpdateJobOpeningQueueable implements Queueable {
2     private List<Id> jobIds;
3
4     public UpdateJobOpeningQueueable(List<Id> jobIds) {
5         this.jobIds = jobIds;
6     }
7
8     public void execute(QueueableContext context) {
9         List<Job_Opening__c> jobs = [
10             SELECT Id, Positions_Open__c
11             FROM Job_Opening__c
12             WHERE Id IN :jobIds
13         ];
14
15         for (Job_Opening__c job : jobs) {
16             if (job.Positions_Open__c > 0) {
17                 job.Positions_Open__c -= 1; // decrement
18             }
19         }
20
21         if (!jobs.isEmpty()) {
22             update jobs;
23         }
24     }
25 }

```

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
    }
}
```

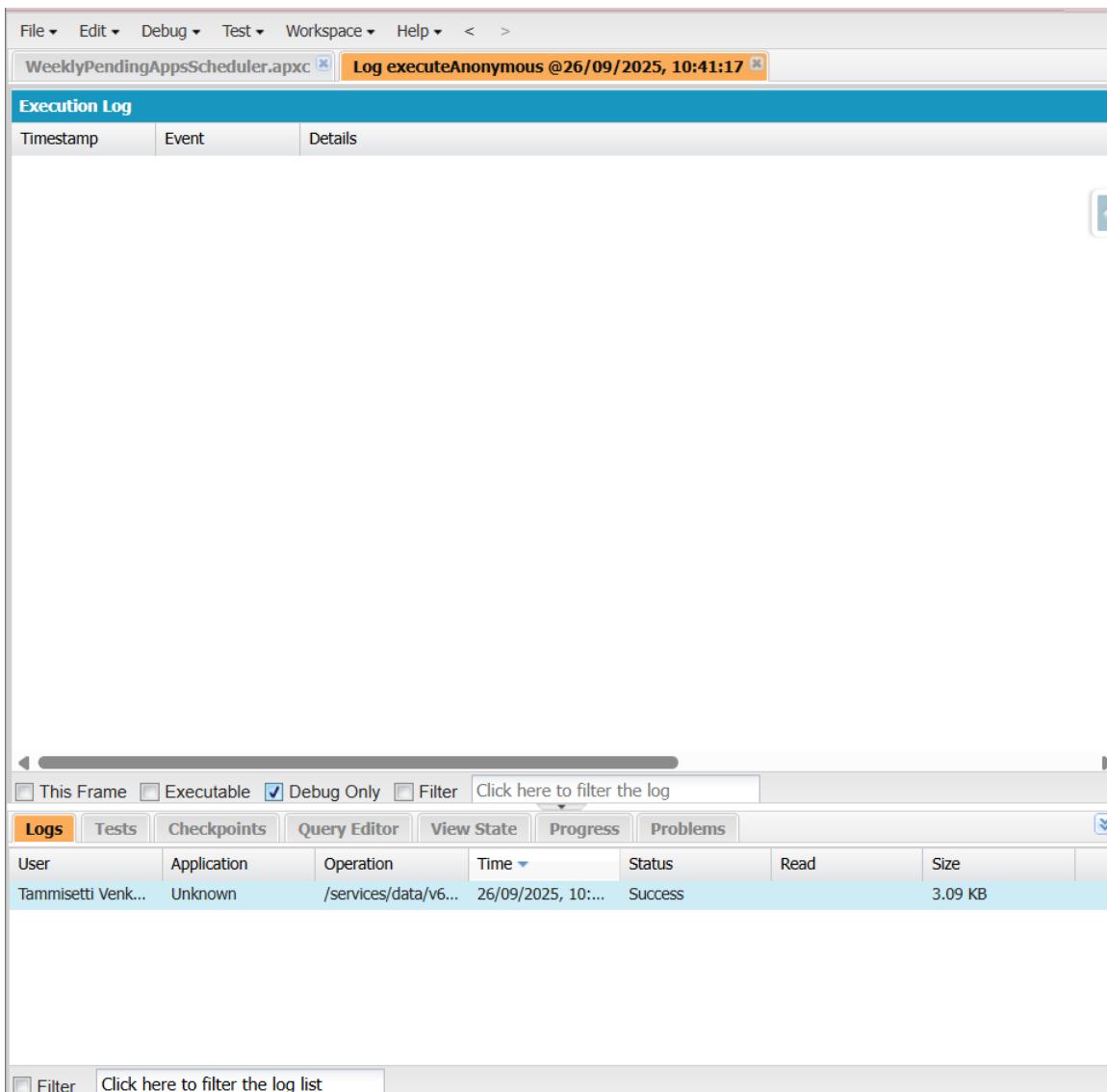
}

The screenshot shows the Salesforce Setup interface. On the left, there's a sidebar with various links like Setup Home, Salesforce Go, Service Setup Assistant, etc. The main area is titled 'Apex Class PendingApplicationsBatch'. It shows the class detail with fields like Name (PendingApplicationsBatch), Status (Active), and Created By (Tammisetti Venkatarao). Below this is the 'Class Body' tab, which contains the Apex code:

```
1 global class PendingApplicationsBatch implements Database.Batchable<sObject> {
2     global Database.QueryLocator start(Database.BatchableContext BC) {
3         Datetime cutoff = System.now().addDays(-7);
4         return Database.getQueryLocator(
5             [SELECT Id, Candidate__c
6              FROM Application__c
7              WHERE Stage__c = 'Screening' AND CreatedDate <= :cutoff]
8         );
9     }
10    global void execute(Database.BatchableContext BC, List<Application__c> scope) {
11        List<Id> candidateIds = new List<Id>();
12        for (Application__c a : scope) {
13            if (a.Candidate__c != null) {
14                candidateIds.add(a.Candidate__c);
15            }
16        }
17        System.debug('PendingApplicationsBatch candidateIds size: ' + candidateIds.size());
18        // TODO: create summary records or take actions on candidates
19    }
20    global void finish(Database.BatchableContext BC) {
21        System.debug('PendingApplicationsBatch finished.');
22    }
23 }
24 }
```

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) {

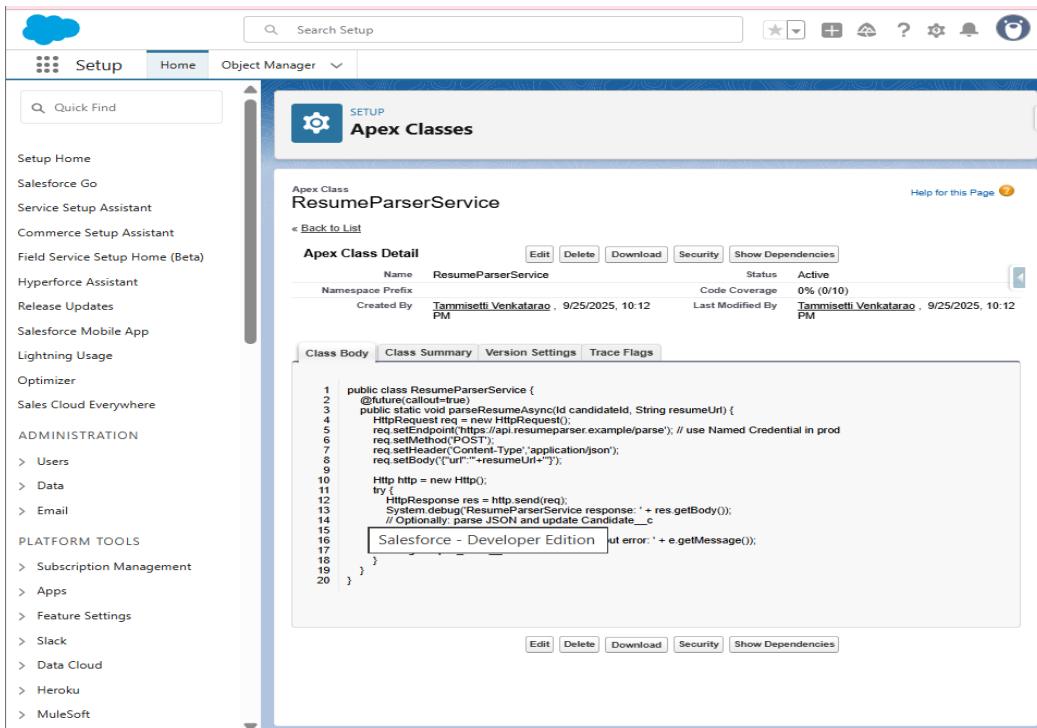
        System.debug('Callout error: ' + e.getMessage());

    }

}

}

```



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

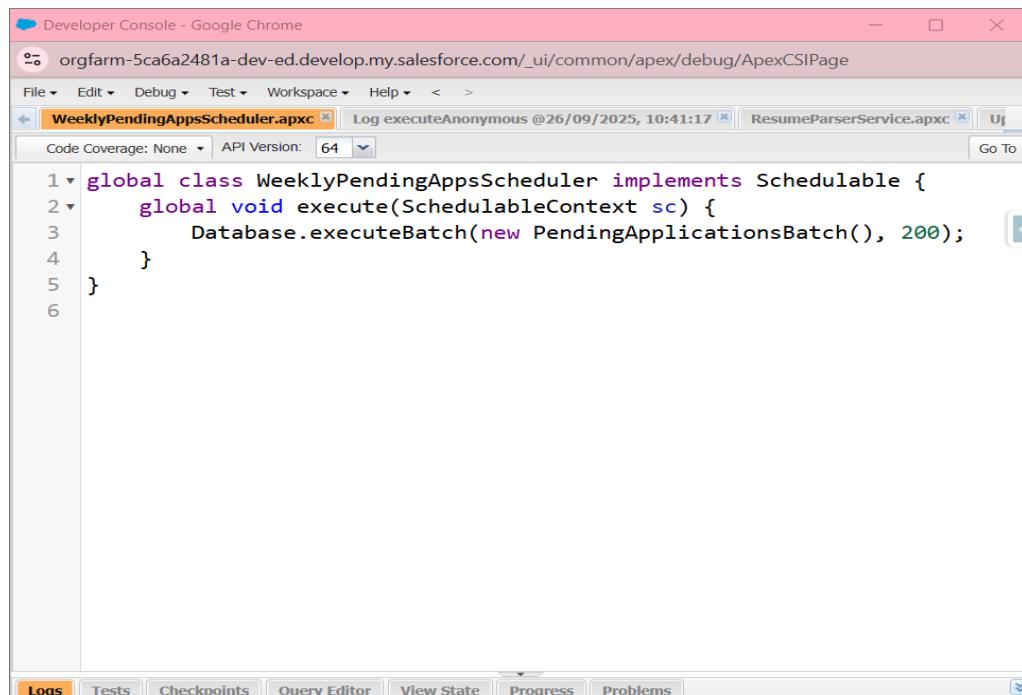
6) Scheduled Apex (recurring jobs)

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 interface. The title bar reads "Developer Console - Google Chrome" and the URL is "orgfarm-5ca6a2481a-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage". The menu bar includes File, Edit, Debug, Test, Workspace, Help, and a Go To button. 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
```

Below the code editor, there are tabs for Logs, Tests, Checkpoints, Query Editor, View State, Progress, and Problems. The "Logs" tab is currently selected.

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];
    }
}

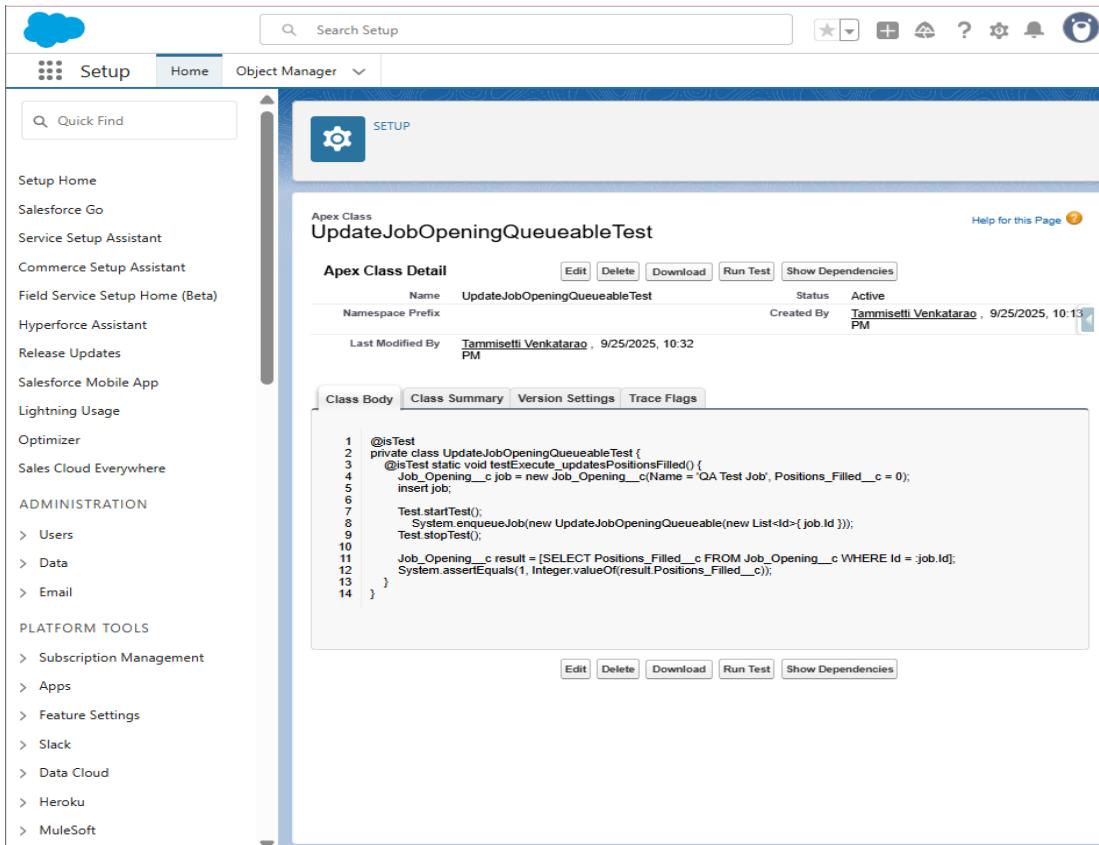
```

```

        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 [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: 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:
 - **Recruiter**
 - Object Permissions: **Read/Write** on Candidate__c and Application__c.
 - **Read Only** on Job_Opening__c.
 - **HR Manager**
 - Full access to all HR-related objects.
 - **Department Head**
 - **Read/Edit** on Job_Opening__c and Interview__c.

The screenshot shows the Salesforce Setup interface with the 'Profiles' page selected. The top navigation bar includes 'Setup', 'Home', and 'Object Manager'. The left sidebar shows 'Users' and 'Profiles' selected. The main content area displays the 'HR Manager' profile details. It includes sections for 'Profile Detail' (Name: HR Manager, User License: Salesforce, Description: , Created By: Tammisetti Venkatara, Modified By: Tammisetti Venkatara), 'Page Layouts' (Standard Object Layouts, Global, Email Application, Home Page Layout, Account, Alternative Payment Method, Appointment Invitation, Asset, Asset Action, Asset Action Source, Asset Relationship, Asset State Period), and 'Object Permissions' (Listed under 'Enabled Record Type Access' and 'Enabled Custom Permissions').

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, the navigation bar includes 'Setup', 'Home', and 'Object Manager'. A search bar at the top says 'Search Setup'. The main area is titled 'Permission Sets' and shows a permission set named 'CandidateSensitiveAccess'. The 'Object Settings' tab is selected, displaying a table of objects and their permissions. The table includes columns for 'Object Name', 'Object API Name', 'Object Permissions', 'Total Fields', and 'Tab Settings'. Many objects have 'No Access' permissions. The 'Activation Data Model Field' row is highlighted, showing 'MktSgtActvDataModFldDef' as the object API name and 'No Access' as the permission level.

Object Name	Object API Name	Object Permissions	Total Fields	Tab Settings
Accounts	Account	No Access	40	--
Activation Attribute	MktSgtActvAudAttrDef	No Access	--	--
Activation Contact Point	MktSgtActvCltcPtDef	No Access	--	--
Activation Contact Point Field	MktSgtActvCltcPtfldDef	No Access	--	--
Activation Contact Point Source	MktSgtActvCltcPtsrcDef	No Access	--	--
Activation Data Model Field	MktSgtActvDataModFldDef	No Access	--	--
Activation Data Source	MktSgtActvDataSourceDef	No Access	--	--
Activation Definition	MarketSegmentActivationDef	No Access	--	--
Activation Platform Activation Attributes	ActivationPlatformActvAttr	No Access	--	--
Activation Platform Audience Identifiers	ActvPlatformAdncIdentifier	No Access	--	--
Activation Platform Data Connector for S3	ActvPfrmDataConnectorS3	No Access	--	--
Activation Platform Fields	ActivationPlatformField	No Access	--	--
Activation Platform Field Value Definitions	ActvTgtPlatFieldValDef	No Access	--	--
Activation Platform Field Values	ActvPlatformFieldValue	No Access	--	--
Activation Platform OAuth Connectors	ActvPlatformOAuthConnector	No Access	--	--
Activation Platforms	ActivationPlatform	No Access	--	--
Activations	MarketSegmentActivation	--	--	--
Activation Target Definition	ActivationTargetDef	No Access	--	--
Activation Target Internal Organization Access	ActivationTrglInOrgAccess	No Access	--	--
Activation Target Platform Definitions	ActivationTgtPlatFormDef	No Access	--	--
Activation Target Platform Field Values	ActvTgtPlatformFieldValue	No Access	--	--
Activation Target Platforms	ActivationTargetPlatform	No Access	--	--
Activation Targets	ActivationTarget	No Access	13	--
Activation Target Secure FTP	ActivationTargetSecureFTP	No Access	--	--
...

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 'Profiles' search term entered. The main window displays the 'Set Field-Level Security' page for the 'Candidate Name' field. The table lists various profiles along with their 'Visible' and 'Read-Only' status. Most profiles have the 'Visible' checkbox checked, while the 'Read-Only' checkbox is unchecked for all.

Field Label	Candidate Name	Visible	Read-Only
Data Type	Text(80)		
Analytics Cloud Integration User	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Analytics Cloud Security User	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Anypoint Integration	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Contract Manager	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Cross Org Data Proxy User	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Custom: Marketing Profile	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Custom: Sales Profile	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Custom: Support Profile	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Department Head	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Einstein Agent User	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Force.com - App Subscription User	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Force.com - Free User	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Gold Partner User	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
HR Manager	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Identity User	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Marketing User	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Minimum Access - API Only Integrations	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Minimum Access - Salesforce	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Partner App Subscription User	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Partner Community Login User	<input checked="" type="checkbox"/>	<input type="checkbox"/>	

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 'Sharing Settings' page in Salesforce. The main title is 'Organization-Wide Sharing Defaults Edit'. It includes a note: 'Edit your organization-wide sharing defaults below. Changing these defaults will cause all sharing rules to be recalculated. This could require significant system resources and time depending on the amount of data in your organization. Setting an object to Private makes records visible to record owners and those above them in the role hierarchy, and access can be extended using sharing rules.' Below this is a table with columns for Object, Default Internal Access, Default External Access, and Grant Access Using Hierarchies.

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

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**.

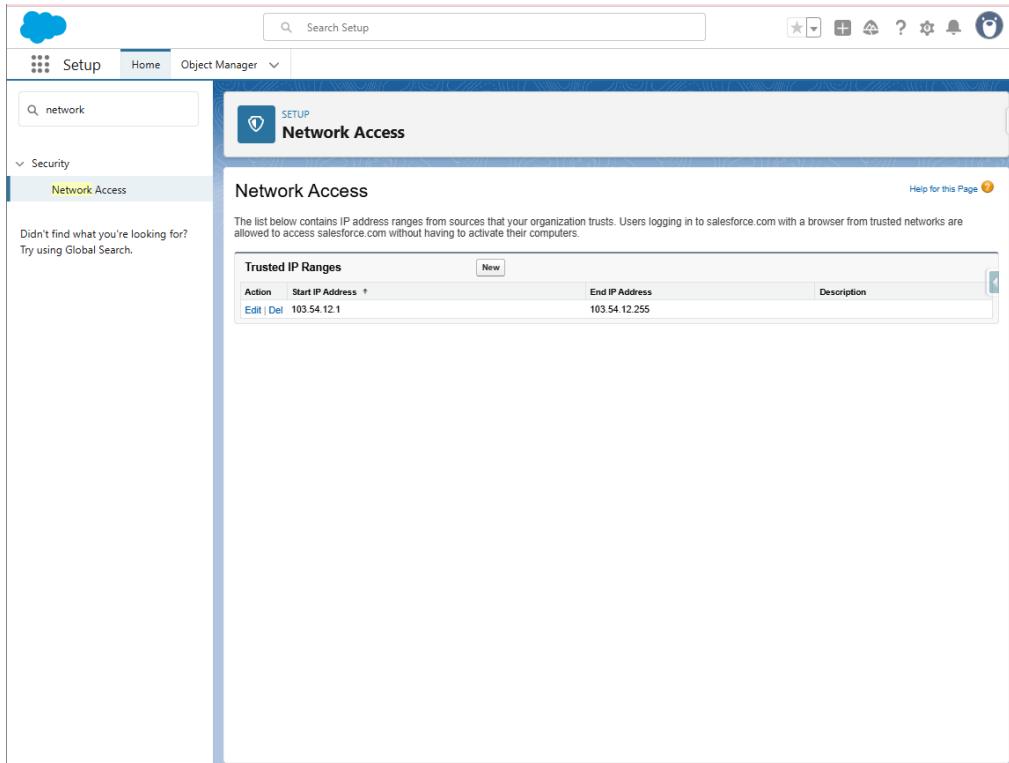
The screenshot shows the Salesforce Sharing Settings page within the Setup interface. The top navigation bar includes a cloud icon, a search bar labeled 'Search Setup', and various global buttons. The main content area is titled 'Sharing Settings' and contains a table for managing sharing rules across different object types:

Sharing Rules	New	Recalculate	Help
Lead Sharing Rules	New	Recalculate	Lead Sharing Rules Help ?
Account Sharing Rules	New	Recalculate	Account Sharing Rules Help ?
Opportunity Sharing Rules	New	Recalculate	Opportunity Sharing Rules Help ?
Case Sharing Rules	New	Recalculate	Case Sharing Rules Help ?
Campaign Sharing Rules	New	Recalculate	Campaign Sharing Rules Help ?
User Sharing Rules	New	Recalculate	User Sharing Rules Help ?
Individual Sharing Rules	New	Recalculate	Individual Sharing Rules Help ?
Voice Call Sharing Rules	New	Recalculate	Voice Call Sharing Rules Help ?
Activation Target Sharing Rules	New	Recalculate	Activation Target Sharing Rules Help ?

Each row indicates 'No sharing rules specified.' and includes a note that no sharing rules have been created.

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**:
 - o Confirm they cannot see salary fields.
 - o Verify they can create and update Candidates and Applications.
2. Log in as **HR Manager**:
 - o Confirm full access, including salary and offer details.
3. Log in as **Department Head**:
 - o Confirm read-only access to Job Openings.

Deliverables

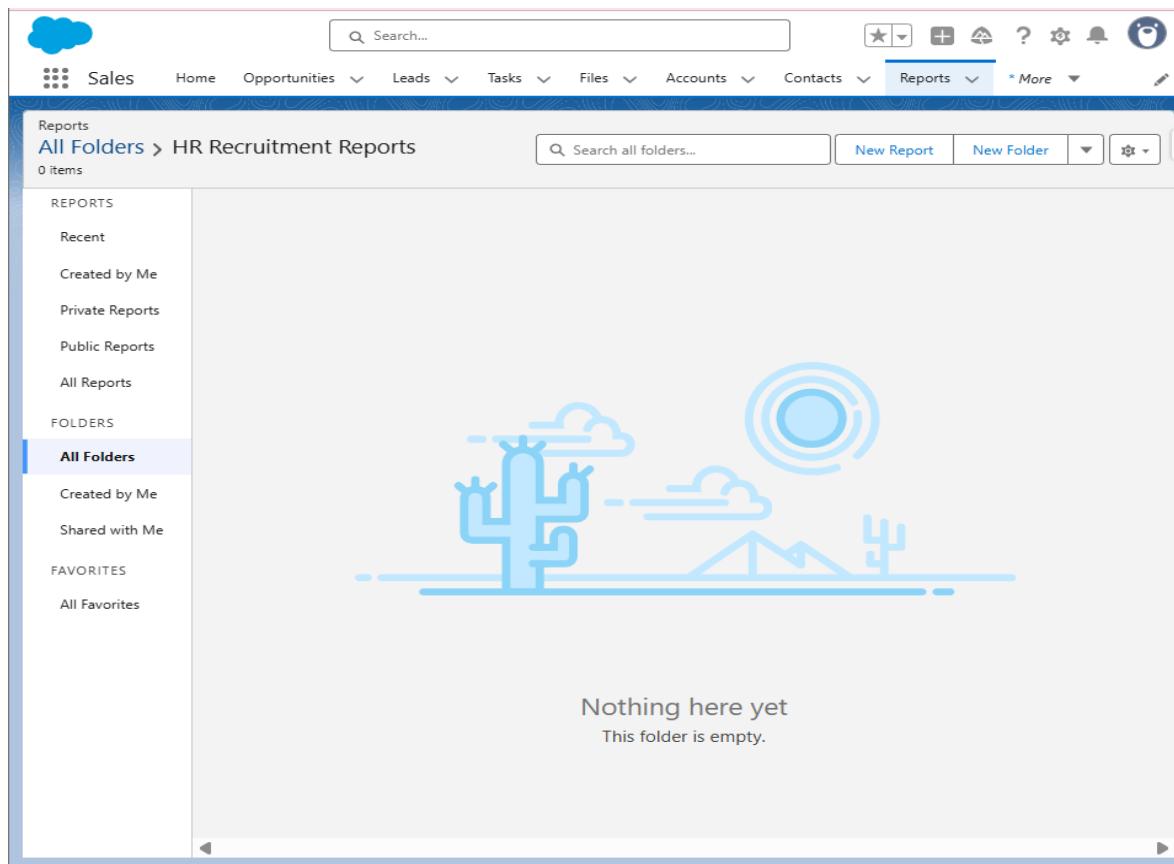
- Screenshots of Profiles, Permission Sets, and OWD configurations.
- Table summarizing Sharing Rules and FLS settings.
- Evidence of testing different user roles.

Phase 7: Reports & Dashboards.

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

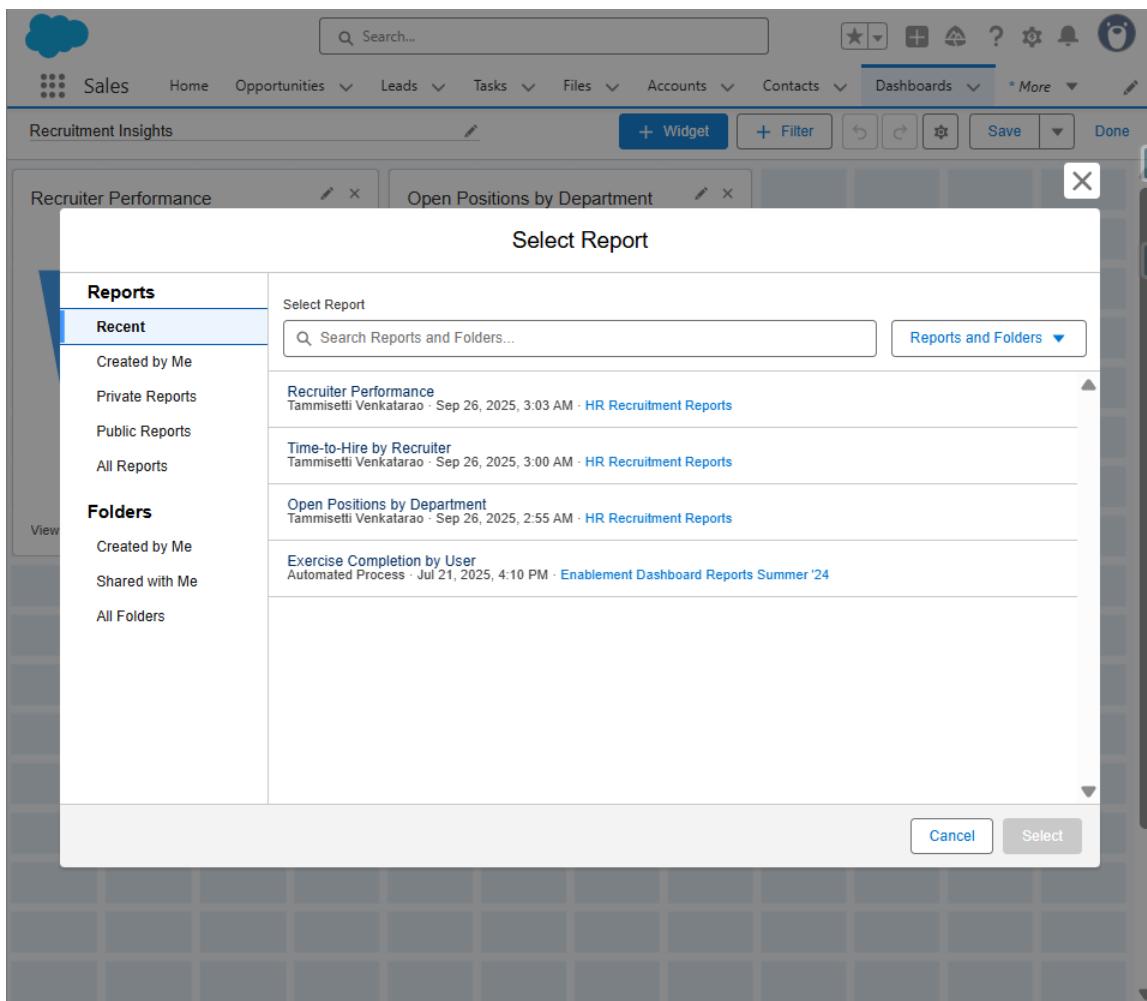
1. Ensure you have a formula field on Application: **Time_to_Hire__c** = **Hired_Date__c** - **Application_Date__c**.
2. Create **New Report** → Applications.
3. Add **Time_to_Hire__c** and **Recruiter__c**.
4. Summarize **Average Time_to_Hire__c**.
5. Save as Time-to-Hire by Recruiter.

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.



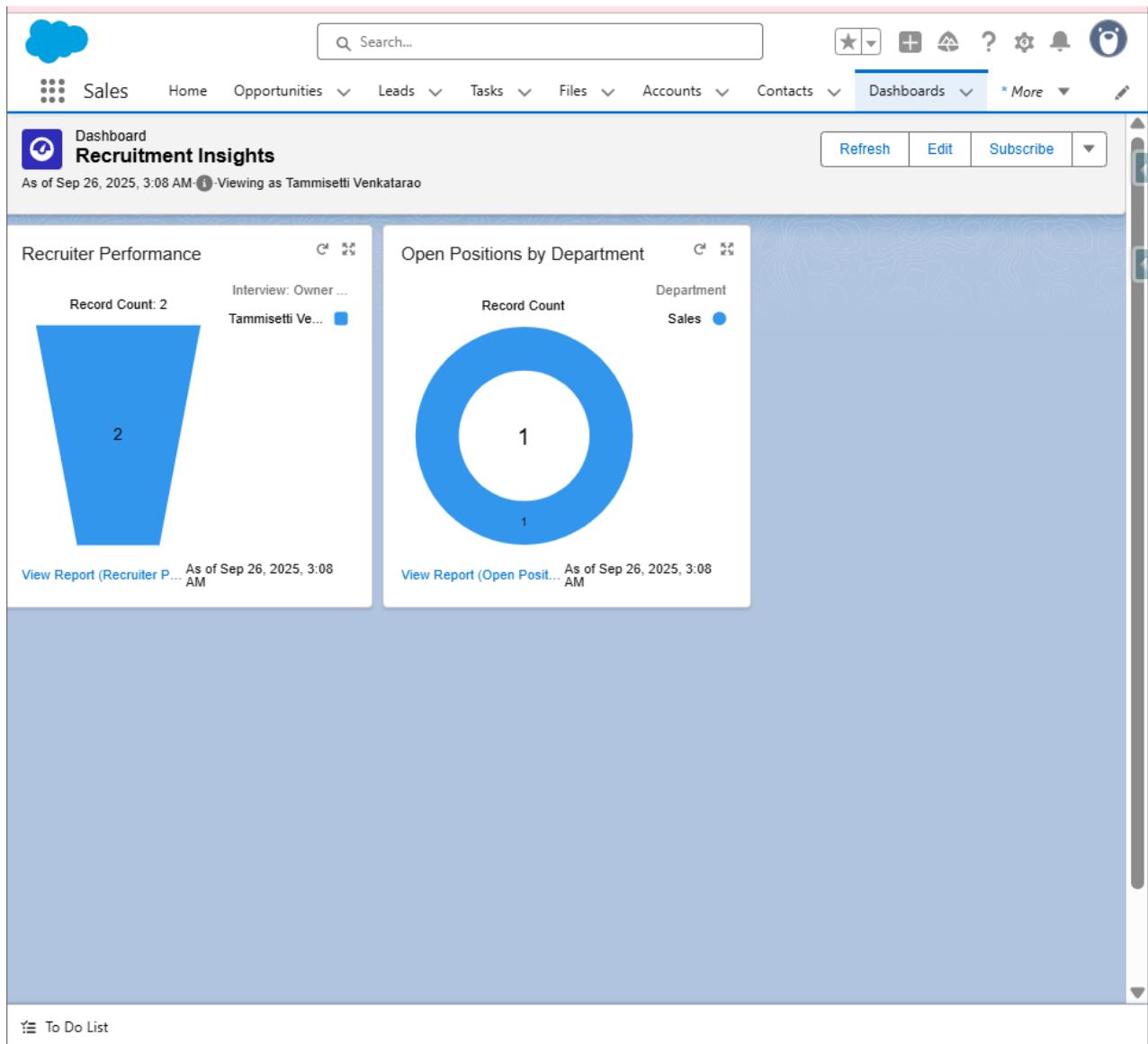
4. Build Dashboards

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.



5. Schedule Report Subscriptions

1. In each report, click ▼ → **Subscribe**.
2. Set: Frequency → Weekly (Monday 9 AM).
3. Recipients: HR Manager, Department Heads.
4. Click **Save**.

Deliverables

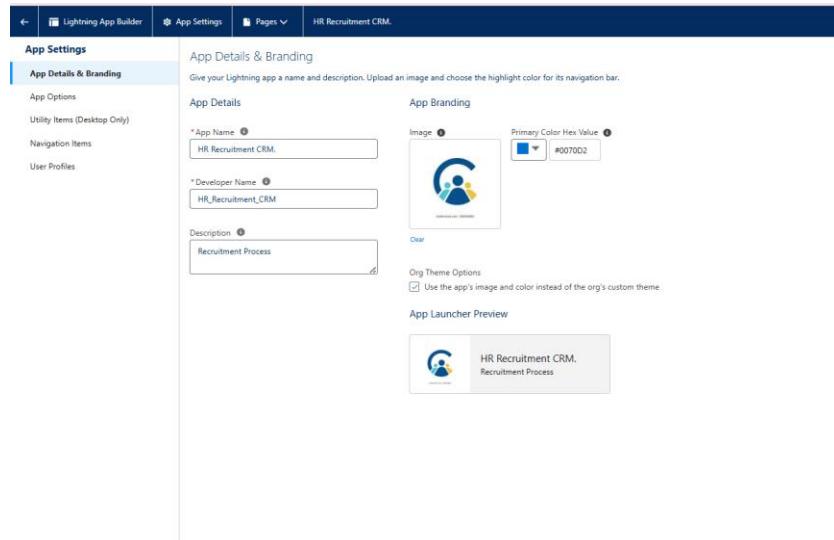
- Screenshots of four reports.
- Screenshot of Recruitment Insights dashboard.
- Subscription confirmation email list.

Phase 8: App Customization & Branding

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

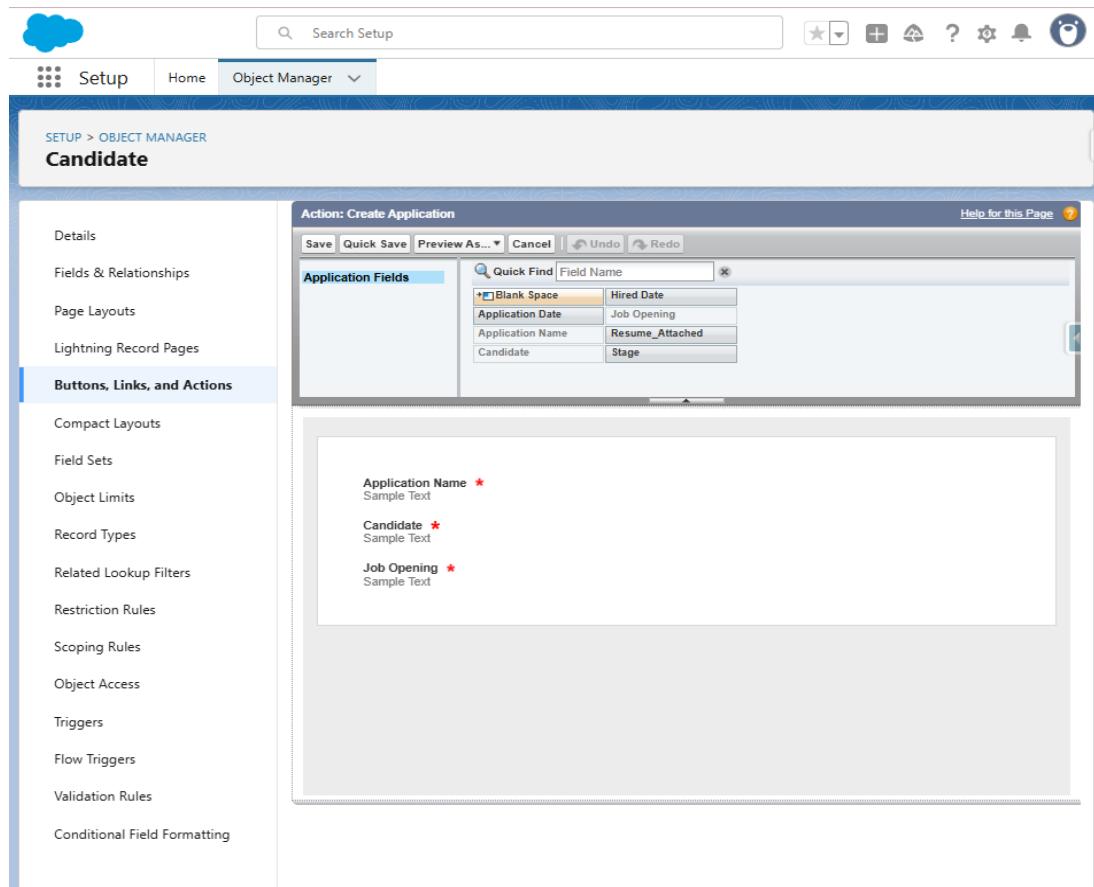
1. Create a Custom Lightning App

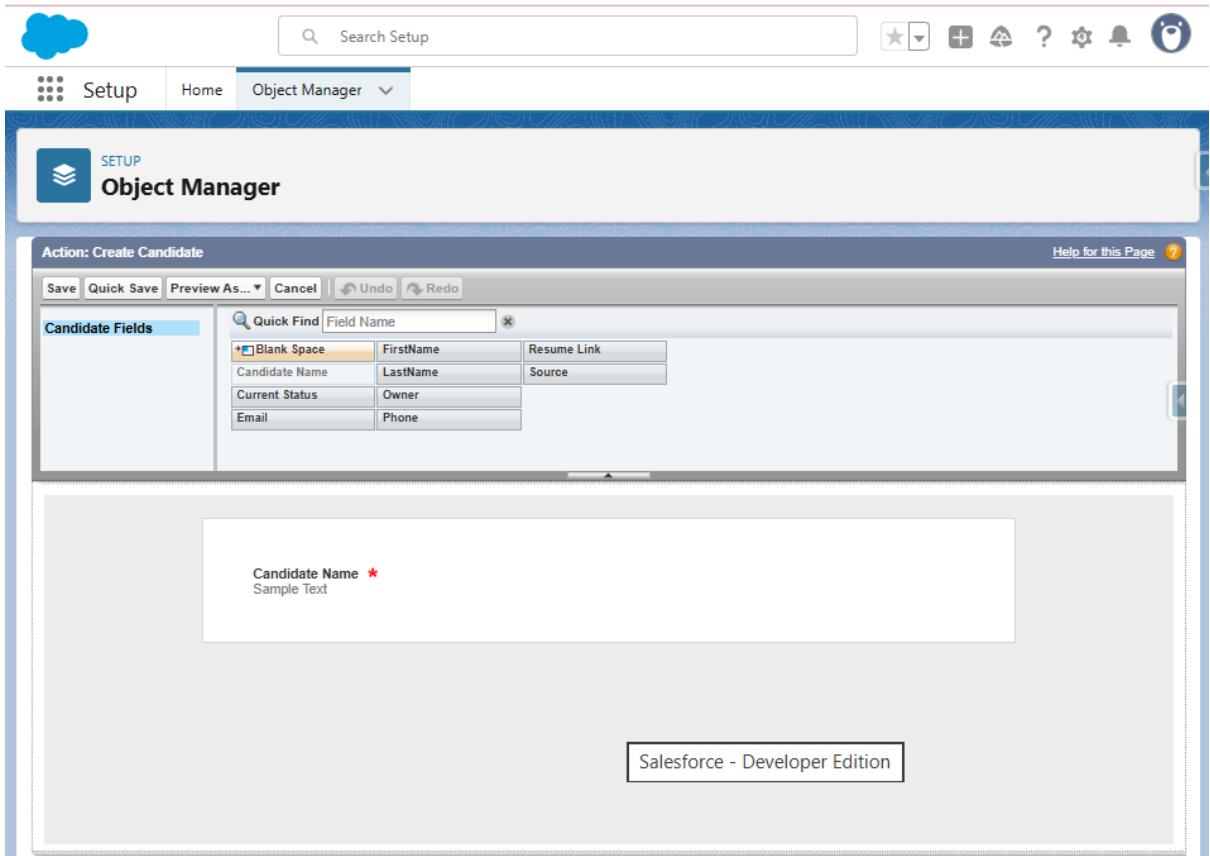
1. Click **Setup** (gear icon) → **App Manager**.
2. Click **New Lightning App**.
3. Enter:
 - **App Name:** HR Recruitment CRM.
 - **Developer Name:** HR_Recruitment_CRM.
 - Upload **Company Logo** (PNG/JPG, 128×128 px).
4. Configure:
 - Navigation Style: **Standard Navigation**.
 - App Options: Enable **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 'Utility Items (Desktop Only)' configuration page in the Lightning App Builder. The left sidebar has 'App Settings' selected. The main area displays a list of utility items: 'Recent Items', 'Chatter Feed', and 'Notes'. The 'Recent Items' item is currently selected. The configuration panel includes fields for 'Label' (set to 'Recent Items'), 'Icon' (set to 'fallback X'), 'Panel Width' (set to 340), 'Panel Height' (set to 480), and a checkbox for 'Start automatically' which is unchecked. Below these are sections for 'Component Properties' with dropdowns for 'Label' (set to 'Custom') and 'Custom Label' (set to 'Standard.RecentItems'), and a section for 'Objects' with a dropdown set to 'API Anomaly Event Store' and a 'Select...' button. At the bottom is a field for 'Number of Records to Display' with the value '3'.

✓ Deliverables

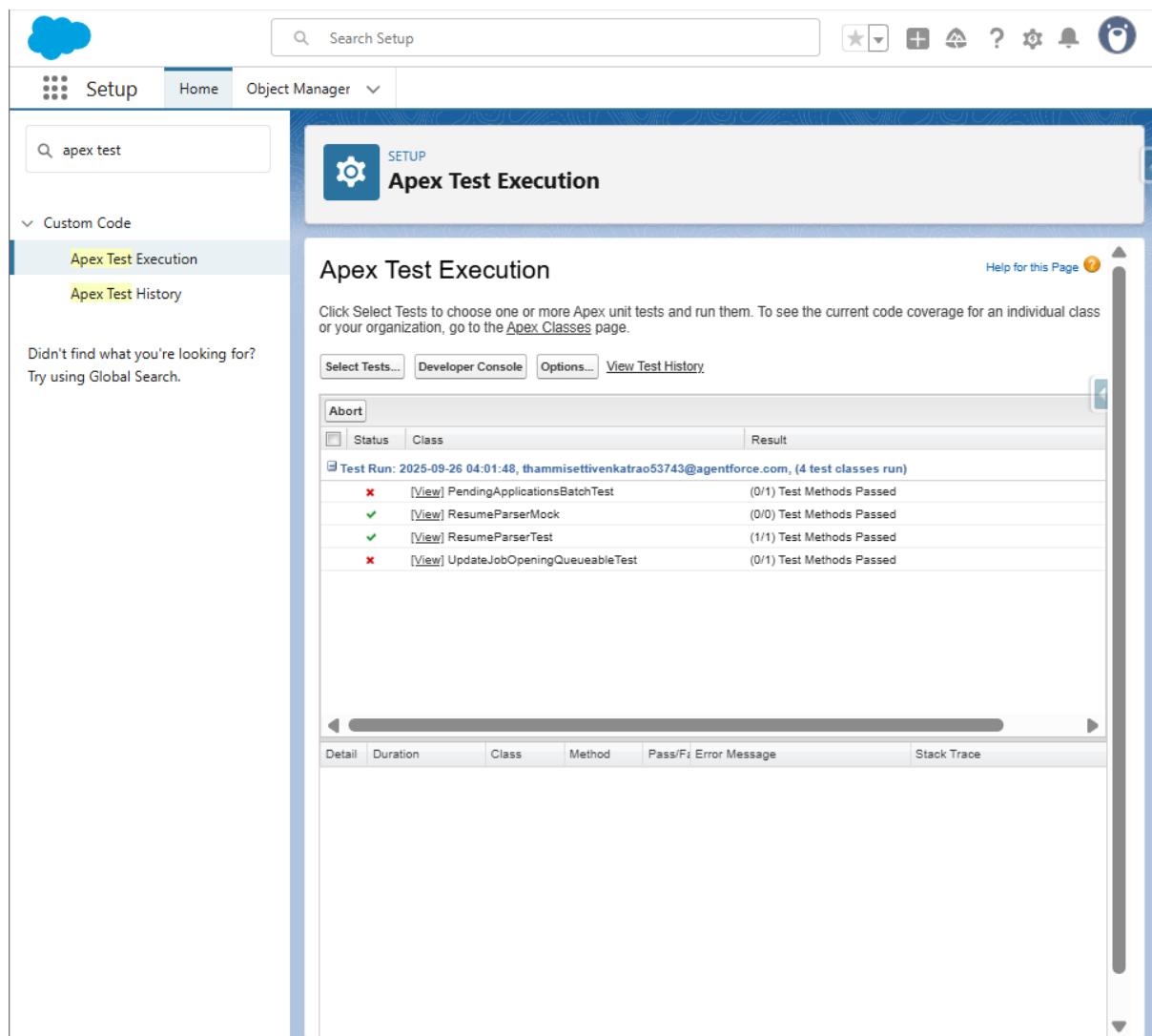
- Screenshot of the HR Recruitment CRM app tile with logo.
- Candidate and Job Opening record page screenshots showing custom components.
- Theme and Branding screenshot.
- Screenshot of Utility Bar configuration.

Phase 9: Testing & Deployment

Goal: Validate all functionality in a Sandbox environment, perform User Acceptance Testing (UAT), and safely deploy the HR Recruitment CRM to Production.

1. Run Unit Tests

1. Click **Setup** → Quick Find → **Apex Test Execution**.
2. Click **Select Tests** → Choose all test classes created in Phase 5.
3. Click **Run**.
4. Verify **Code Coverage ≥ 75%**.



2. Perform Functional Testing

1. Recruiter Role Test:

- Create Candidate.
- Submit Application.
- Schedule Interview.
- Confirm Recruiter cannot see salary fields.

2. HR Manager Role Test:

- Approve Job Opening.
- View full Candidate details (including sensitive fields).

3. Department Head Role Test:

- Access Job Openings (Read Only).

3. Create a Rollback Plan

1. Retain a backup of old configurations or flows.
2. Maintain Apex/Flow versions in Source Control (e.g., GitHub).
3. If deployment fails, deactivate new components or redeploy a previous version.

✓ Deliverables

- Sandbox UAT Sign-off document.
- Deployment Change Set summary.
- Screenshot of successful deployment in Production.
- Smoke Test checklist and results.

Rollback plan stored in documentation folder.