HireMe - Job Recruitment Training CRM

Phase 1: Problem Understanding & Industry Analysis

👉 **Goal:** Understand what we're building and why.

---

1. Requirement Gathering

Talk to stakeholders (HR Manager, Training Coordinator, Hiring Manager, Candidate).
Example requirements:

- Track job recruitment training programs for new hires.
- Allow candidates/employees to enroll in training sessions.
- Provide automated reminders, attendance tracking, and completion certificates.
- Link training results with job performance and recruitment success rate.
- Generate reports on training effectiveness, candidate progress, and ROI.

---

2. Stakeholder Analysis

- **Admin (you):** Manage CRM setup, configure objects, automation, and reports.
- **HR Manager:** Defines training curriculum, ensures compliance, and measures effectiveness.
- **Training Coordinator:** Schedules sessions, tracks attendance, uploads materials.
- **Hiring Manager:** Monitors candidate training progress and evaluates readiness for roles.
- **Candidate/Employee:** End-user, enrolls in training, completes assessments, and gets certified.

---

3. Business Process Mapping

Draw a flow:
Job Application → Candidate Selected → Training Assigned → Candidate Enrolls → Session Conducted → Attendance + Assessments → Training Completed → Feedback + Certificate → HR Manager Reviews Performance.

---

4. Industry-Specific Use Case Analysis

- In the recruitment industry, **training is crucial** to bridge the gap between candidate selection and job readiness.
- Many industries (IT, Banking, Retail, BPO, Healthcare) require **structured onboarding and skill-based training**.
- Compliance-related training (data security, workplace ethics, safety) is often **mandatory before employment confirmation**.
- Real-time progress tracking and **personalized learning paths** are essential to reduce dropouts.

---

5. AppExchange Exploration

- Search for "Learning Management System (LMS)" or "Recruitment Training" apps on AppExchange.
- Examples: **Cornerstone OnDemand, Docebo, Salesforce LMS apps.**
- But for learning purposes, we'll build a **custom Job Recruitment Training CRM** with Salesforce objects, automation, and dashboards.

📊 Industry Analysis

---

1. Current Problems in Recruitment Training

- Training programs are managed manually in spreadsheets/emails.
- No centralized view of candidate enrollment, progress, or completion.
- Difficult to measure ROI of training on hiring success.
- Lack of personalized training schedules for different roles.
- No automated reminders → high dropout rates.

---

2. Current Market Solutions

- **Learning Management Systems (LMS):** Platforms like Moodle, SAP SuccessFactors, Cornerstone.

- **HR Tech Tools:** Workday, Oracle HCM, Zoho People – offer onboarding + training features.
- **Recruitment CRMs:** Focus on hiring but lack deep training integration.

---

3. Market Trends

- **Skill-Based Hiring:** Recruiters emphasize training for role-specific skills.
- **Blended Learning:** Mix of online (self-paced) + classroom training.
- **Gamification:** Quizzes, leaderboards, and badges increase candidate engagement.
- **AI-driven Personalization:** Adaptive training paths for different candidates.
- **Compliance Training:** Mandatory certifications in regulated industries (finance, healthcare, IT).

---

4. Gaps in Current Systems

- Existing tools are either **too generic (LMS)** or **too recruitment-focused (ATS)**.
- Lack of **integration** between job recruitment and training modules.
- Limited **analytics on training effectiveness vs job performance**.
- High **cost** of enterprise LMS tools for small-medium businesses.

---

5. Opportunity for Job Recruitment Training CRM

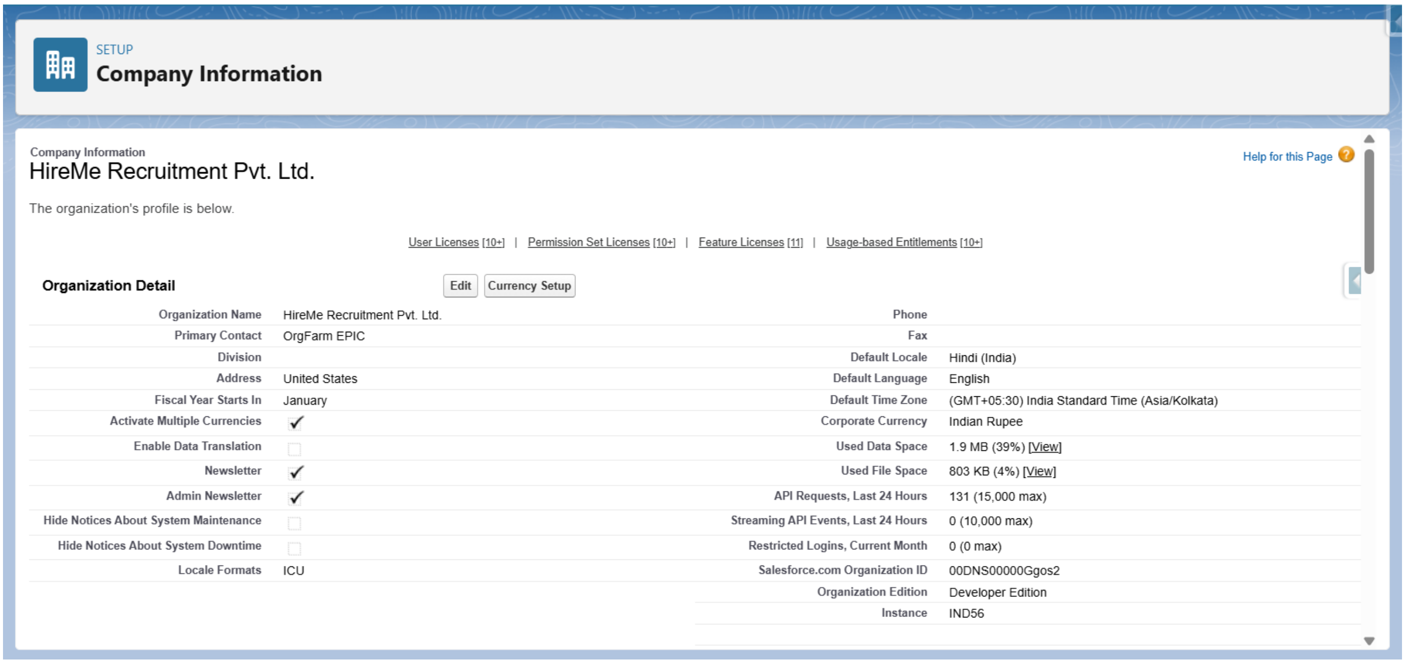
The custom CRM can solve these gaps by:

1. **Centralized Platform:** Integrates recruitment and training in one system.
2. **Automated Enrollment:** Assign training sessions to candidates automatically after hiring stage.
3. **Session Management:** Schedule classes, upload materials, track attendance.
4. **Assessment & Certification:** Online tests + auto-generated certificates.
5. **Progress Tracking:** Dashboards for candidates, HR, and hiring managers.
6. **Performance Reports:** Measure training impact on candidate success in job roles.

📌 Phase 2: Org Setup & Configuration

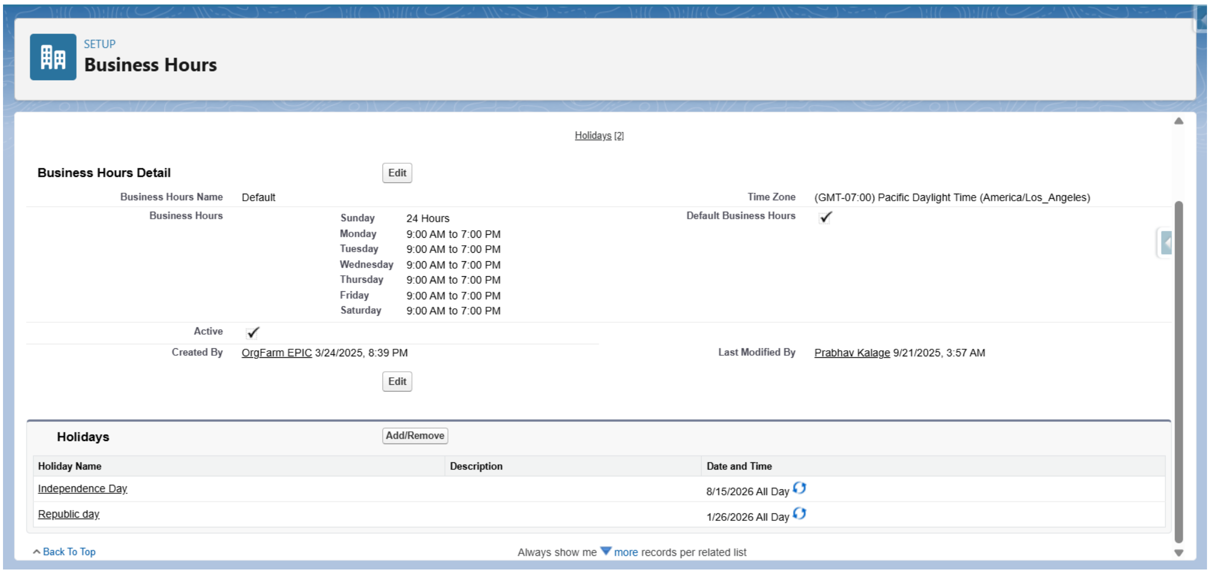
🔹 1. Salesforce Edition & Org Setup

- Selected **Salesforce Developer Edition Org** for implementation.
- Connected Org with **VS Code (SFDX Project Setup)** for version control & deployment.
- Verified basic org information:
  - Company Profile: **HireMe Recruitment Pvt. Ltd.**
  - Default Locale: **English (India)**
  - Default Currency: **INR (₹)**



🔹 2. Company Profile Setup

- Updated Company Information → Company Name = **HireMe CRM**.
- Fiscal Year = **Standard (Jan–Dec)**.
- Configured Business Hours:
  - Default = **Mon–Sat, 09:00 AM – 7:00 PM IST**
  - Holidays = Added **National & Regional holidays** (placeholder list).



◆ 3. User Setup & Licenses

Created initial project users with **Standard Salesforce Licenses**:

- CEO (System Administrator)
- HR Head
- Recruitment Manager
- Trainer
- Recruiter

◆ 4. Role Hierarchy

Configured Role Hierarchy for controlled visibility:

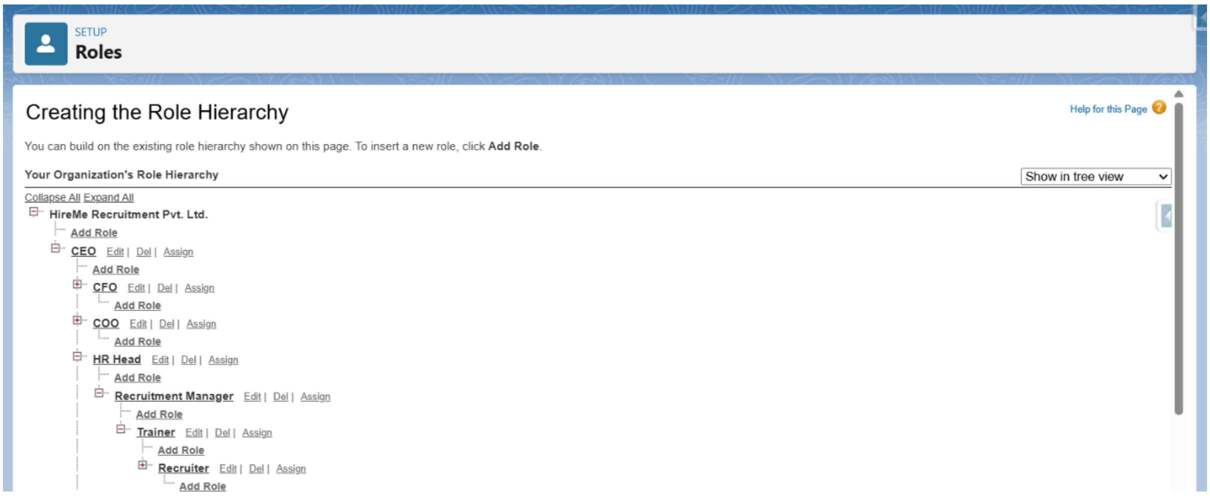
graph TD

A[CEO] --> B[HR Head]

B --> C[Recruitment Manager]

C --> D[Trainer]

D --> E[Recruiter]



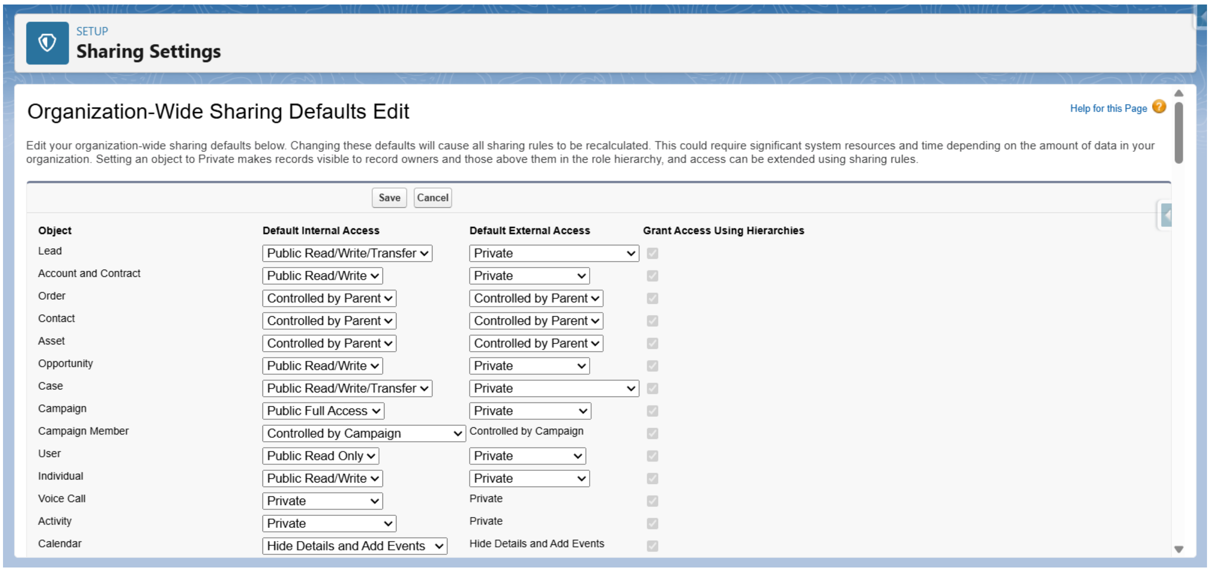
- **CEO → Full visibility**
- **HR Head → Access to all Recruitment Managers' records**
- **Recruitment Manager → Access to Trainers & Recruiters' records**
- **Trainer → Access to Recruiters' assigned candidates**
- **Recruiter → Access only to their own candidate/job records**

---

5. Organization-Wide Defaults (OWD)

- **Position__c → Private (roles/visibility controlled)**
- **Job_Posting__c → Private (linked to Position, synced externally)**
- **Employment_Website__c → Controlled by Parent (linked with Job Posting)**
- **Job_Application__c → Private (central workflow node)**
- **Candidate__c → Private (submitted via Job Application)**
- **Review__c → Controlled by Parent (evaluations tied to Job Applications)**

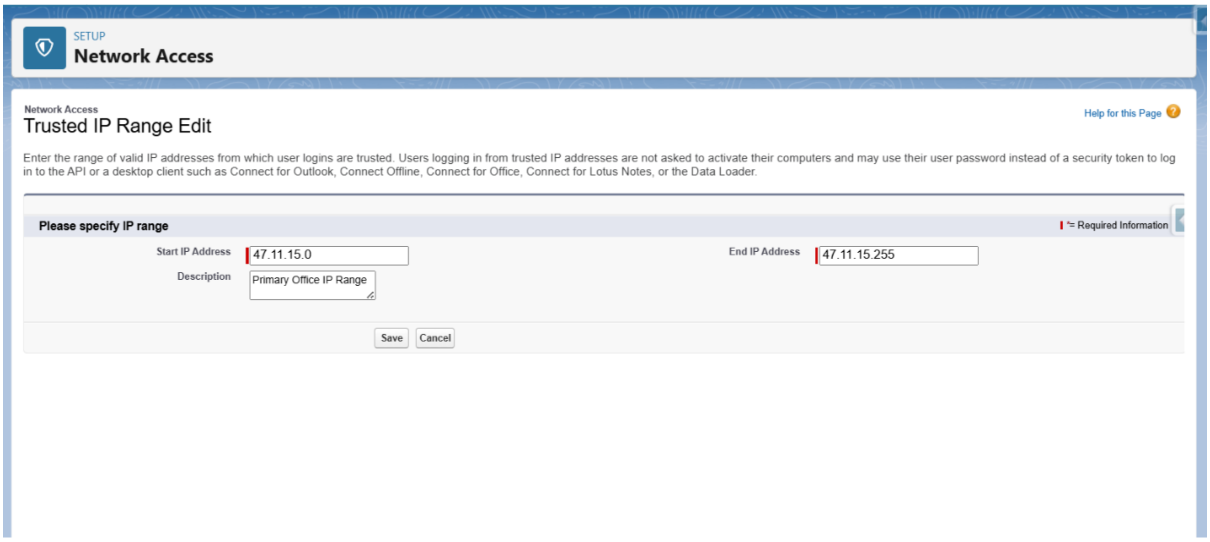
👉 **Grant Access Using Hierarchies → Enabled**



◆ 6. Login & Security Policies

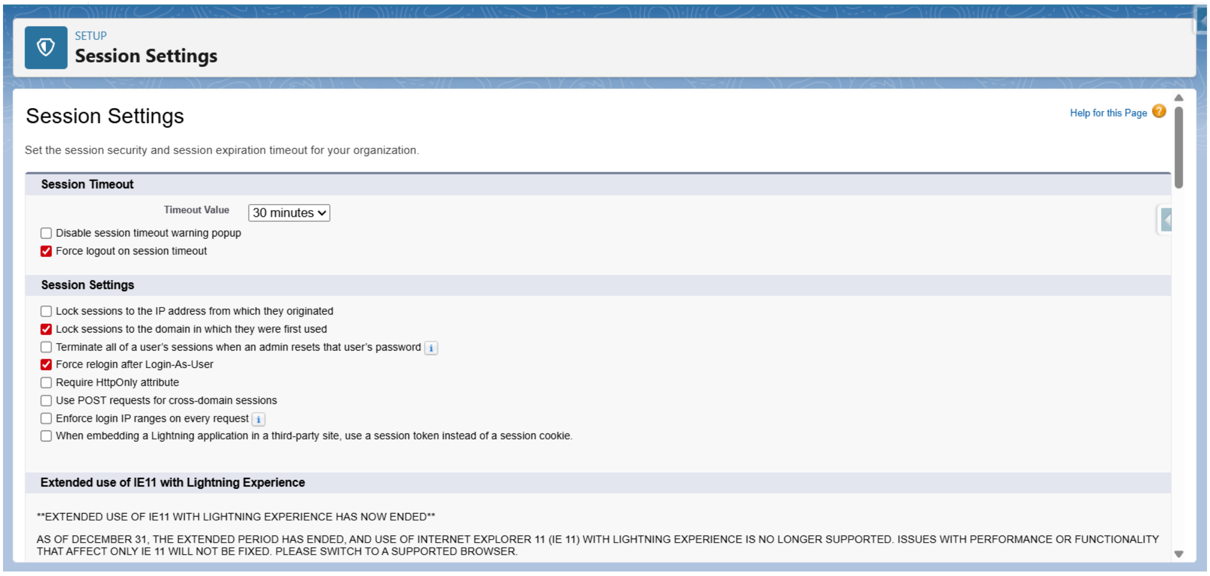
6.1 Trusted IP Ranges (Network Access)

Start IP	End IP	Description
47.11.15.0	47.11.15.255	Primary Office IP Range

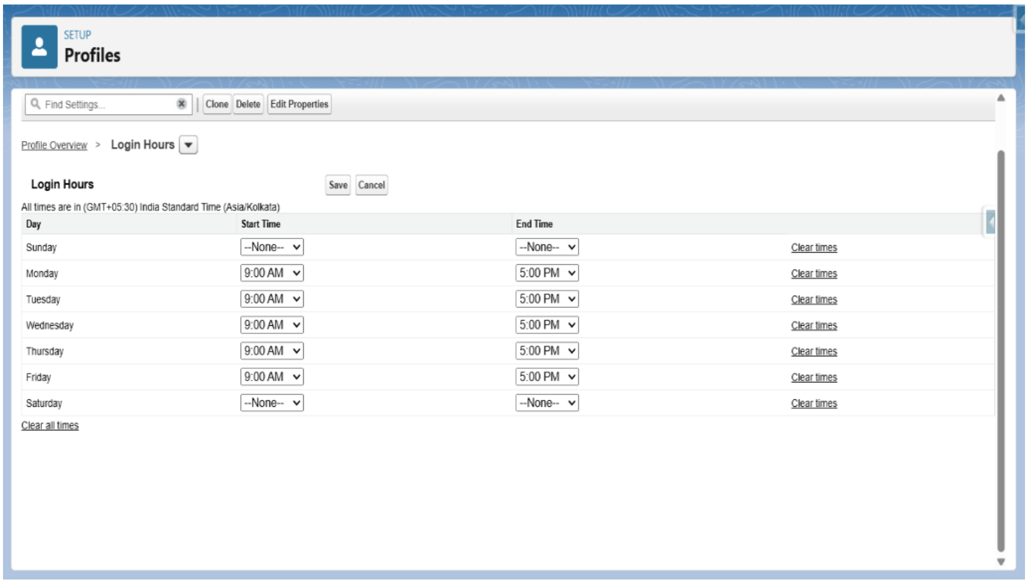


6.2 Session Settings

- Session Timeout: **30 minutes**
- Lock sessions to originating IP: **Disabled (kept flexible for remote users)**



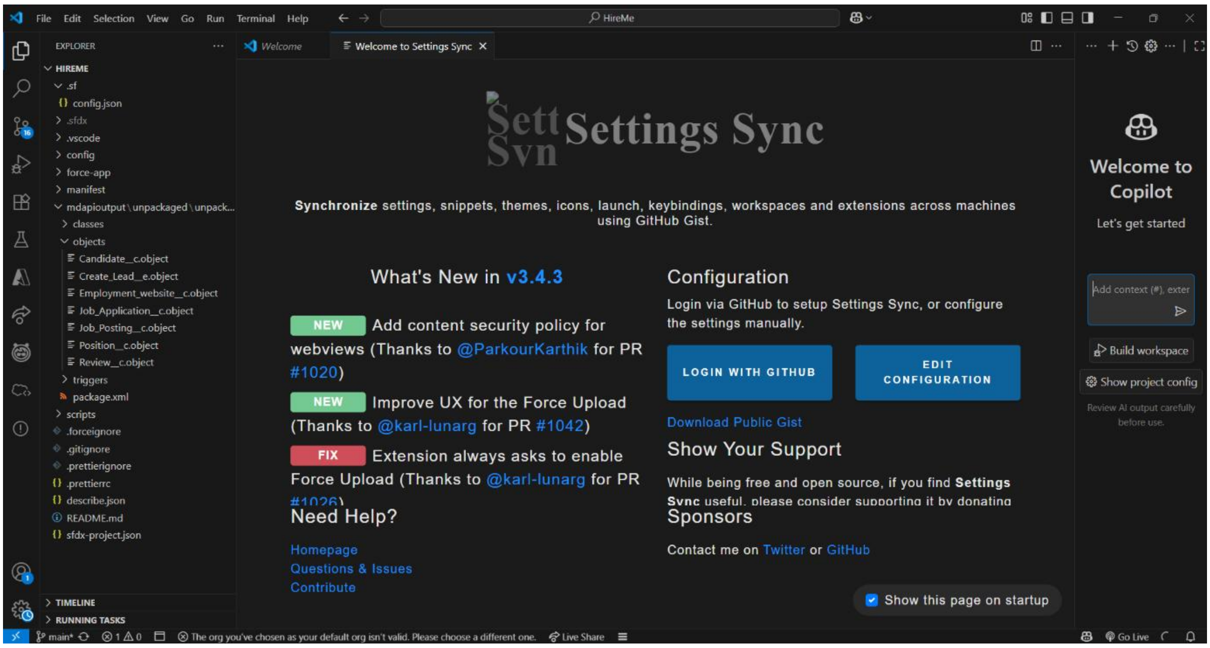
6.3 Login Hours (Profiles)



---

7. Deliverables Produced

- Role Hierarchy Diagram
- OWD & Access Rules Documentation
- Login & Security Policy Document
- Org Setup Verification Report



📌 Phase 2 Status: Completed Successfully ✅

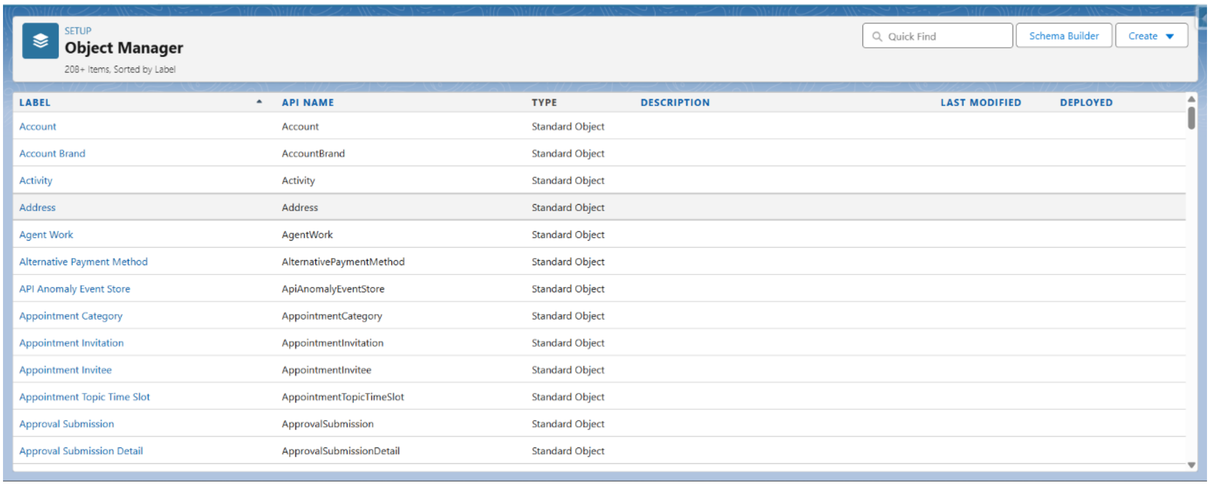
Next Step → Phase 3: Data Modeling & Object Configuration (Custom Objects, Fields, Relationships).

📌 Phase 3: Data Modeling & Relationships

🔹 1. Standard & Custom Objects

- **Standard Objects Used**
    - **User** (mapped to Recruiter, HR Head, Reviewer roles)
    - **Contact** (candidate information reference, if needed)
- **Custom Objects Created**

1. **Position__c** → Represents roles within the organization
2. **Job_Posting__c** → Advertisements linked to positions
3. **Employment_Website__c** → External job boards/platforms
4. **Candidate__c** → Applicant records
5. **Job_Application__c** → Central application object (junction between Candidate & Position)
6. **Review__c** → Stores evaluations, decisions, and next steps

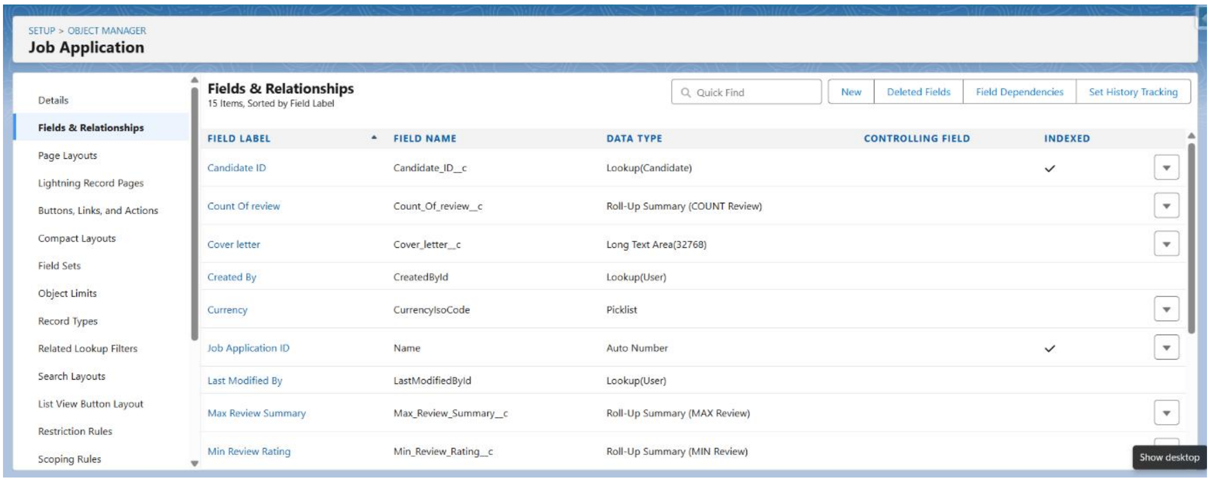


🔹 2. Fields

Created **custom fields** on each object:

- **Position__c**
  - Position Title (Text)
  - Department (Picklist: Tech, HR, Marketing, Sales, etc.)
  - Status (Picklist: Open, Closed, On Hold)
- **Job_Posting__c**
  - Posting Title (Text)
  - Posting Date (Date)
  - Expiry Date (Date)
  - Status (Picklist: Active, Inactive)
- **Employment_Website__c**
  - Website Name (Text)
  - Website URL (URL)
  - Integration Status (Picklist: Connected, Not Connected)
- **Candidate__c**
  - Full Name (Text)
  - Email (Email)
  - Phone (Phone)
  - Resume (File)
- **Job_Application__c**
  - Application Date (Date)
  - Status (Picklist: Submitted, In Review, Shortlisted, Rejected)
  - Source (Picklist: Direct, Website, Referral)
- **Review__c**
  - Review Date (Date)
  - Reviewer (Lookup → User)

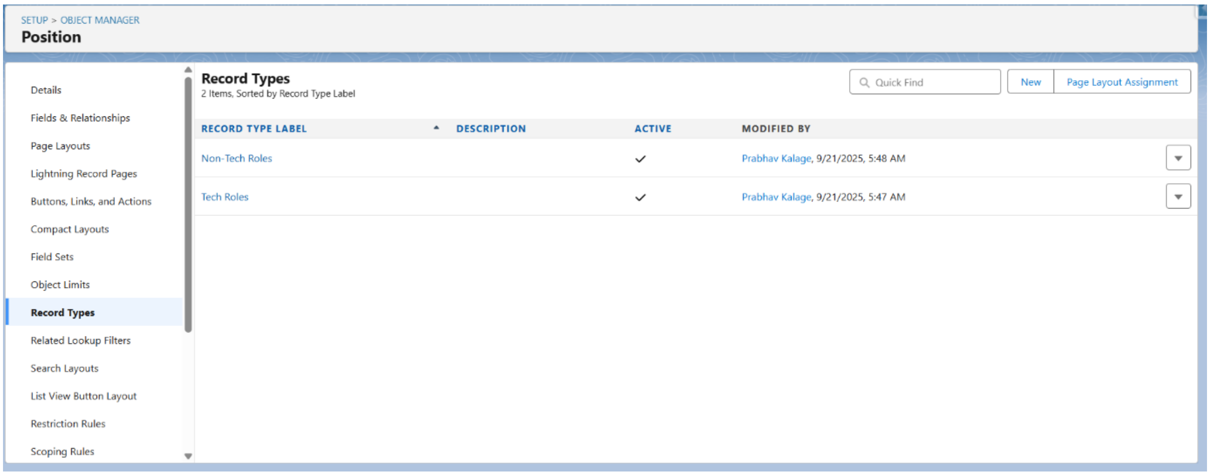
- Review Comments (Long Text)
- Result (Picklist: Selected, On Hold, Rejected)



---

3. Record Types

- **Position__c**
  - Tech Roles
  - Non-Tech Roles
- **Job_Application__c**
  - Fresher Application
  - Experienced Application

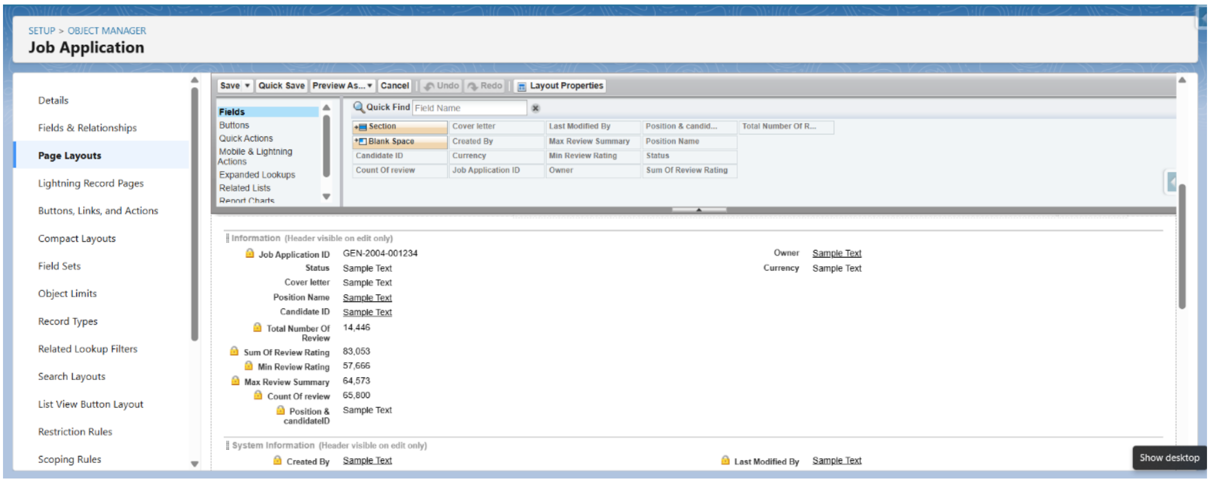


---

4. Page Layouts

- Configured separate layouts for:

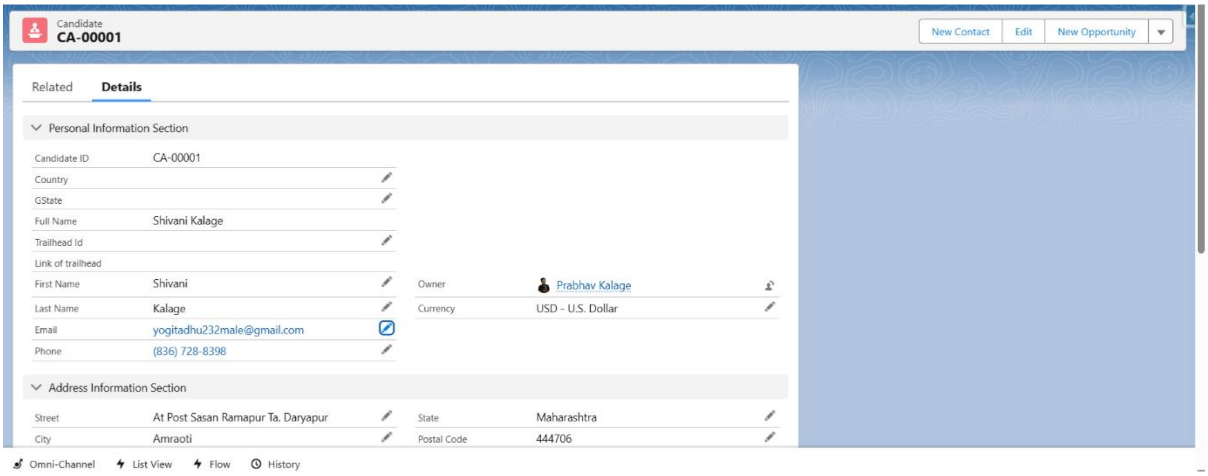
- **Recruiters** (Candidate intake-focused fields)
- **Hiring Reviewer** (Review-specific fields)
- **HR Head** (Summary view of Position, Job Posting, Applications)



---

5. Compact Layouts

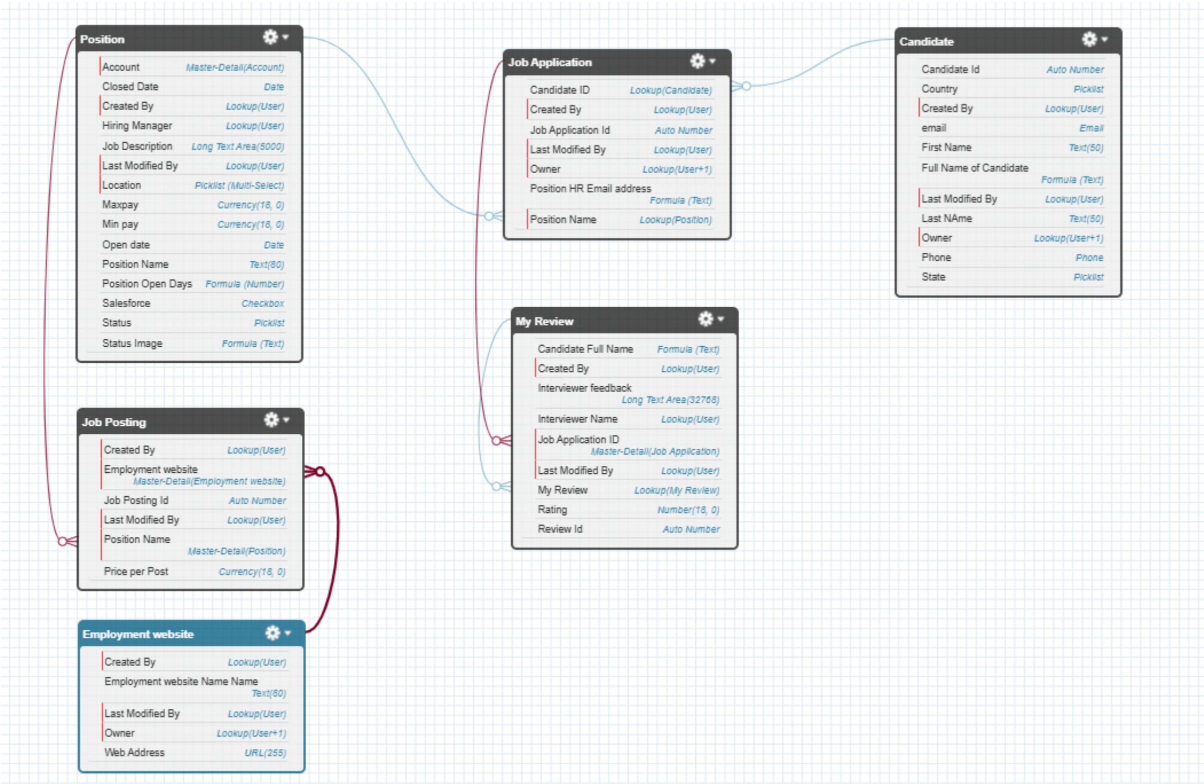
- Defined compact layouts for quick reference:
  - **Candidate__c** → Name, Email, Phone, Application Status
  - **Job_Application__c** → Candidate, Position, Status, Date



---

6. Schema Builder

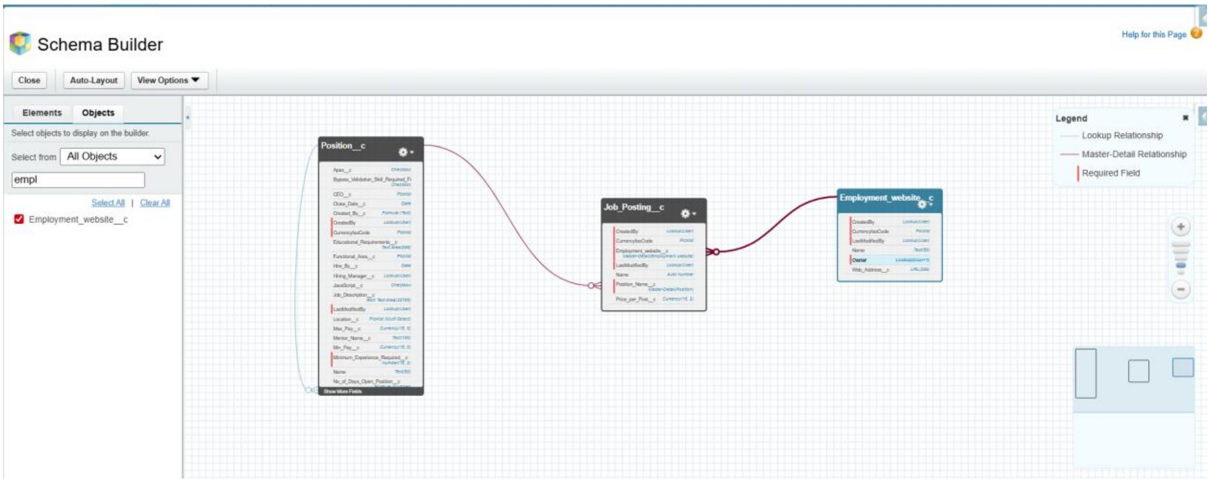
- Visualized the **relationships** among all objects:



🔹 7. Lookup vs Master-Detail vs Hierarchical Relationships

- **Position → Job_Posting** → Master-Detail (Job Posting tied to Position)
- **Job_Posting → Employment_Website** → Lookup (external references, optional)
- **Candidate → Job_Application** → Lookup (candidate may apply multiple times)
- **Position → Job_Application** → Master-Detail (application always tied to a position)
- **Job_Application → Review** → Master-Detail (reviews belong to application)
- **Hierarchical** → Not used (only applicable for User object).

- **8. Junction Objects**
  - **Job_Application__c** functions as a **junction object**:
    - Connects **Candidate__c** ↔ **Position__c**
    - Enables a many-to-many relationship (one candidate can apply to many positions, and one position can have many candidates).



Phase 4: Process Automation

🔹 1. Validation Rules

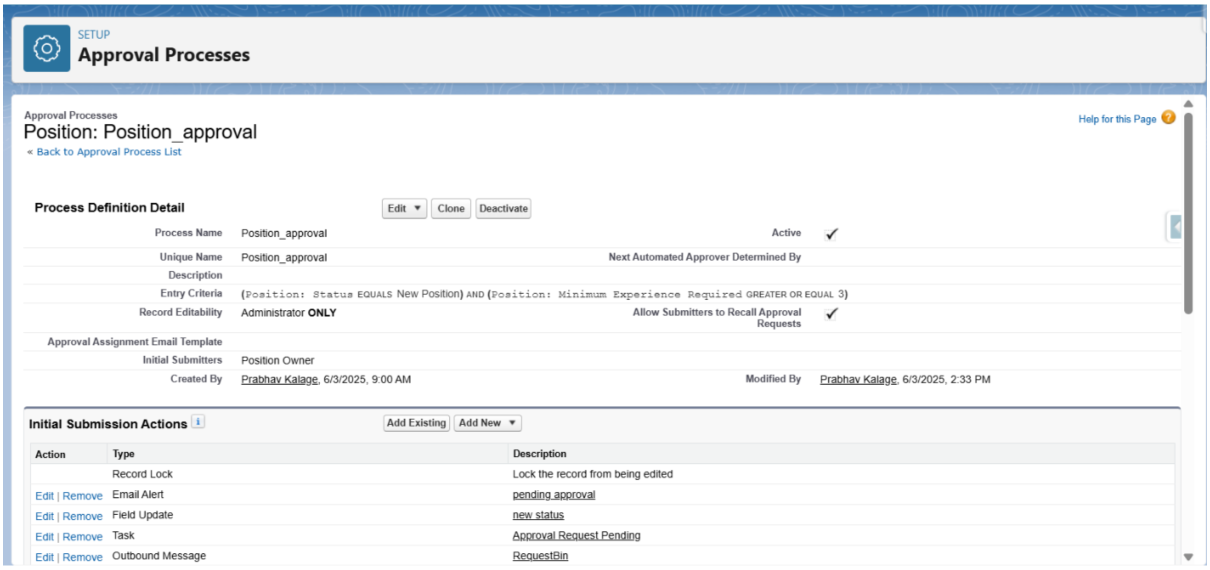
Definition: Enforce data integrity by preventing invalid data from being saved.

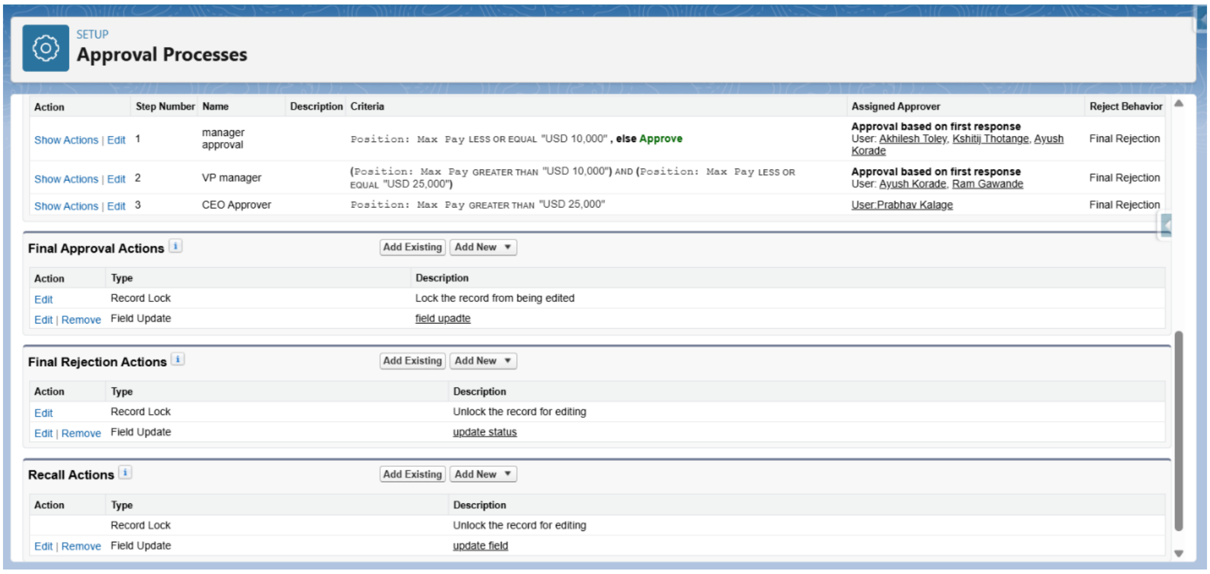
- Example 1 (Position): MIN_Pay__c > MAX_Pay__c → Error: *"Min Pay cannot exceed Max Pay."*
- Example 2 (Candidate): NOT(REGEX(Phone_Number__c , "[7-9]{1}[0-9]{9}")) → Error: *"Phone number must start with 7–9 and be 10 digits."*

🔹 4. Approval Process

Definition: Automates record submission for approval with multiple levels of approvers.

- Example (Position Approval Flow):
    1. Recruiter submits Position → Goes to Recruitment Manager.
    2. If approved → Goes to HR Head.
    3. If approved → Position Status = "Open Approved".
    4. If rejected → Status = "Closed - Not Approved".





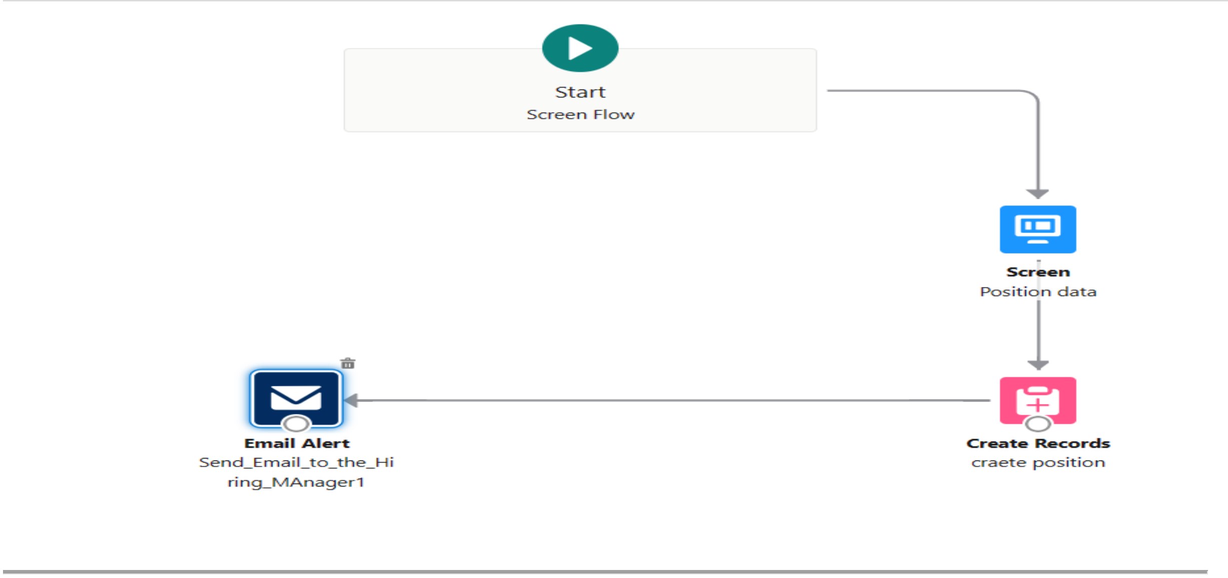
🔹 5. Flow Builder

Definition: Modern automation tool (recommended over Workflow & Process Builder).

Types of Flows:

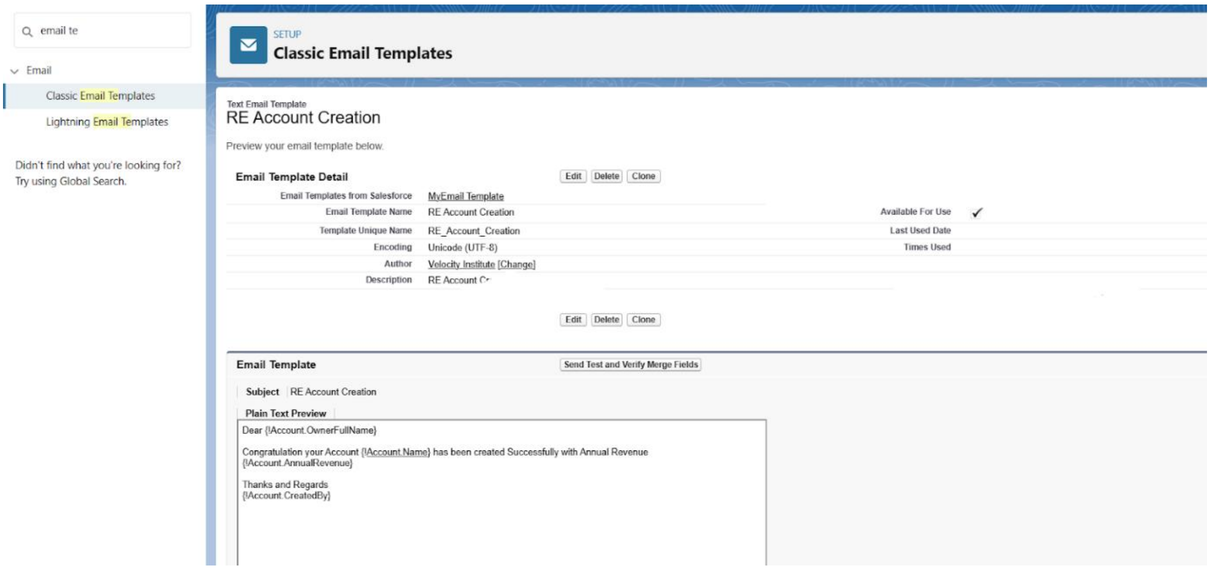
1. **Screen Flow** → Candidate self-service application form.
2. **Record-Triggered Flow** → On Job Application creation → auto-send email to recruiter.
3. **Schedule-Triggered Flow** → Every Monday → Send weekly report of pending applications to HR.

4. **Auto-launched Flow** → When Candidate is marked “Selected” → auto-update Position status.



6. Email Alerts

Definition: Send standard/custom email templates when triggered by automation.



.

Phase 5: Apex Programming

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

In this phase, a **before insert trigger** was created on the Position__c object. The purpose of this trigger is to **auto-populate important fields** when a new Position record is created.

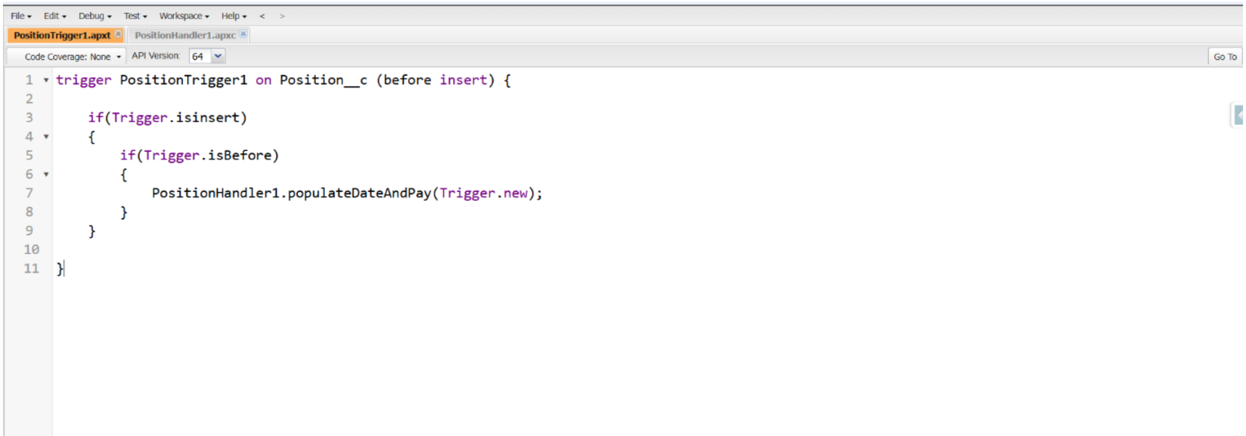
📘 Example: PositionTrigger1 (Before Insert)

---

2. Trigger Design Pattern with Handler Class

The handler class PositionHandler1 contains the business logic. Its job is to **set default values** when a new Position record is created with Status = 'New Position'.

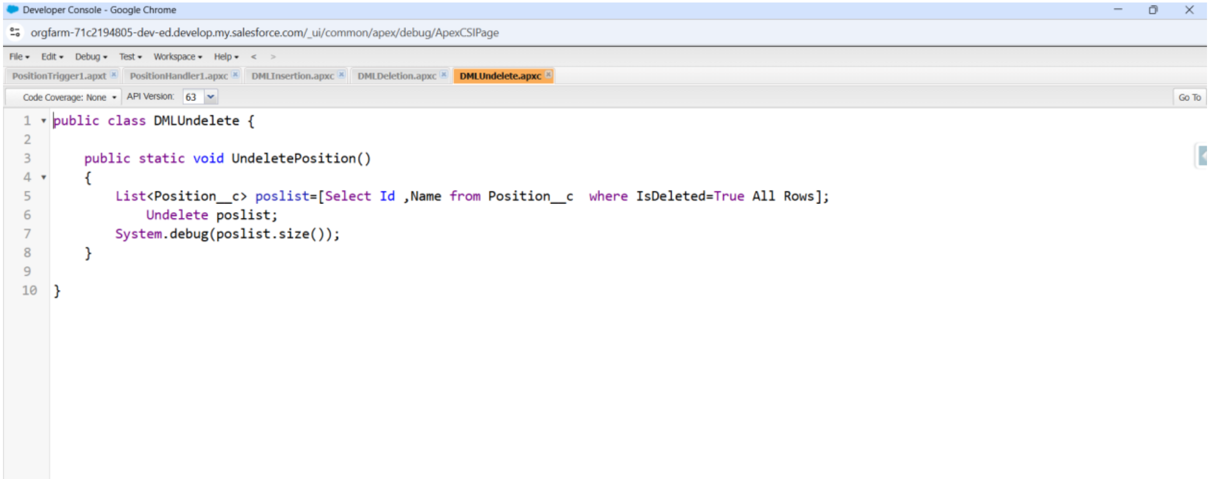
📘 Example: PositionHandler1



3. DML – Undelete Operation

Apex provides the ability to **restore records that were moved to the Recycle Bin** using the undelete keyword. This is helpful when users mistakenly delete records but the data needs to be recovered programmatically.

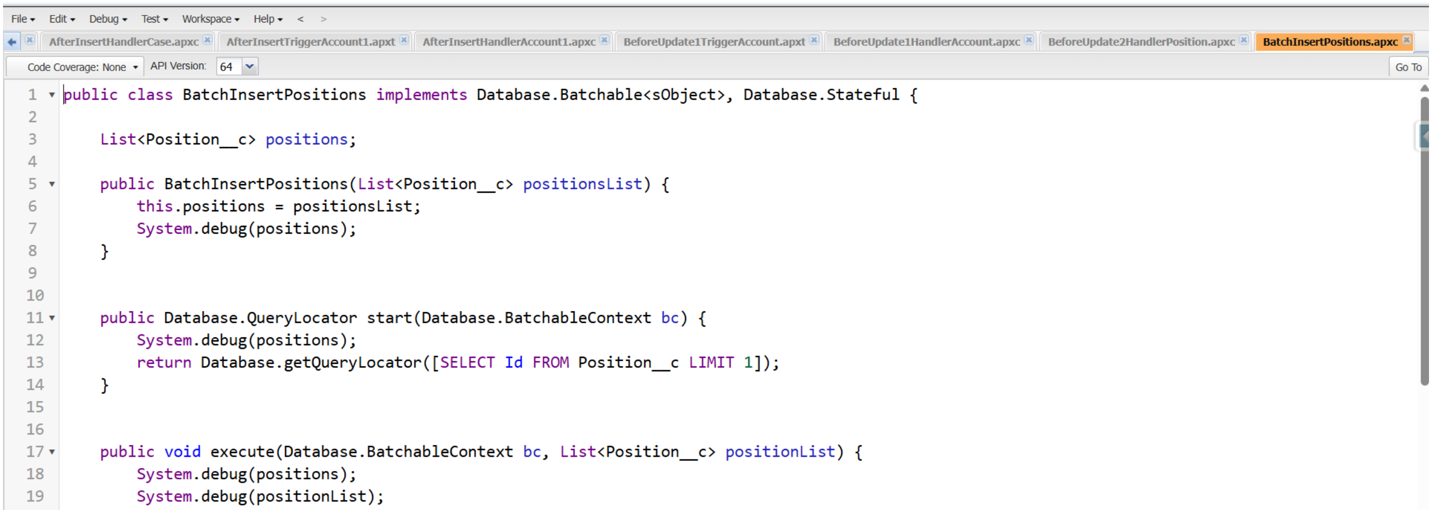
📘 Example: DMLUndelete Class



4. Batch Apex

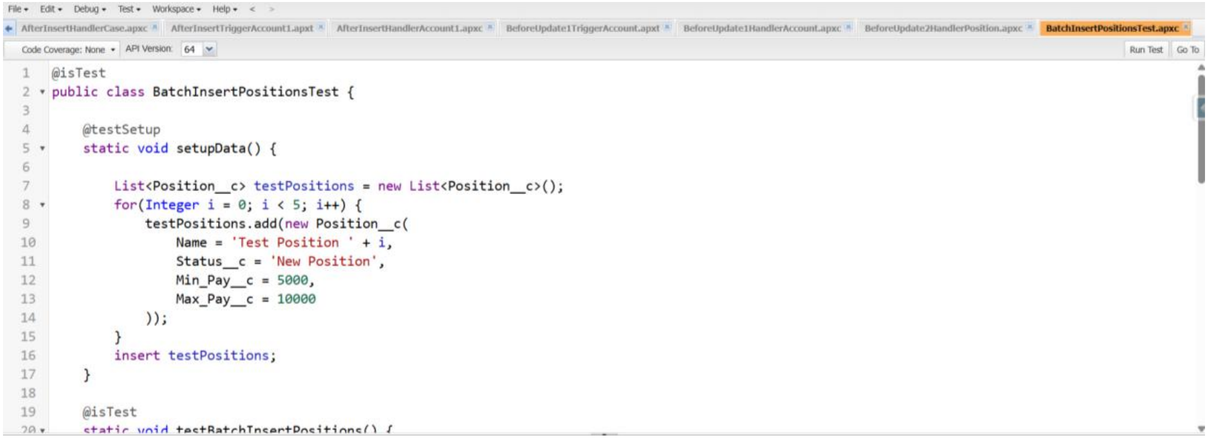
Batch Apex was implemented to handle **large-scale record insertion**. This is particularly useful when data comes from external systems or REST APIs.

📘 Example: BatchInsertPositions

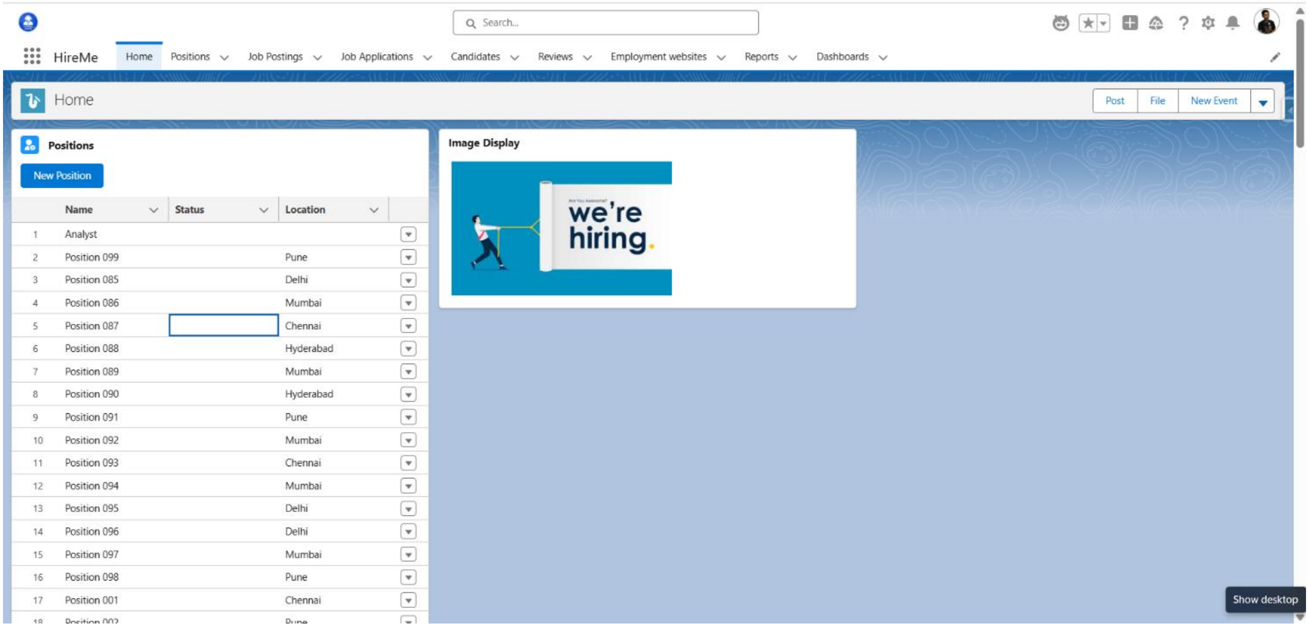


5. Test Class

This test class validates the functionality of the **BatchInsertPositions** class by simulating different scenarios. It ensures that the batch job inserts records correctly and handles empty input lists without errors.



Phase 6: User Interface Development

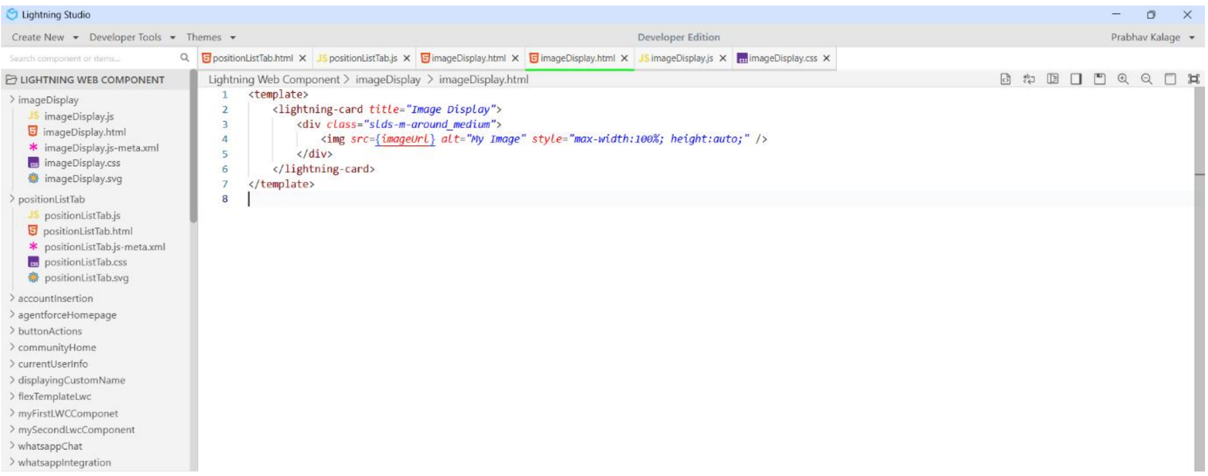


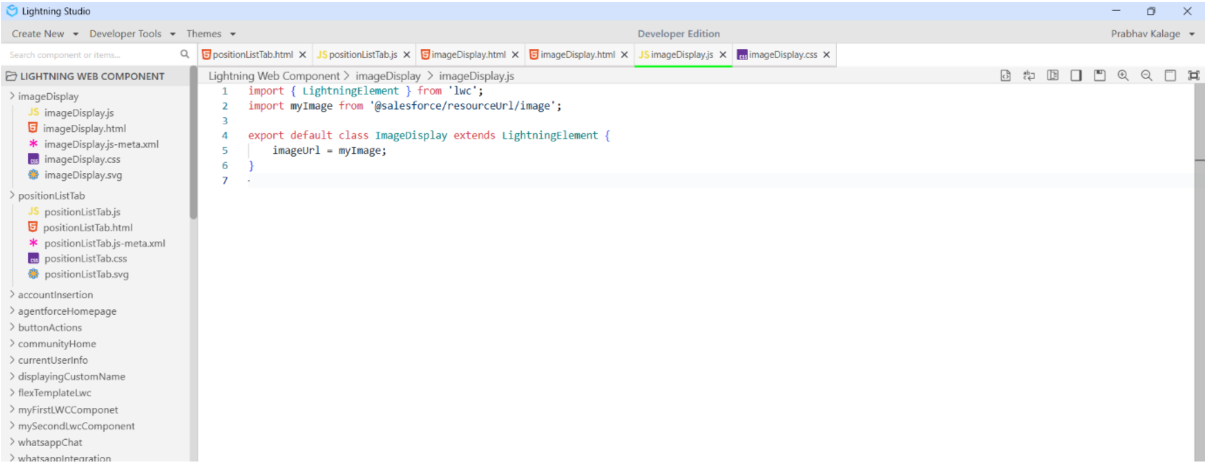
1) Image Display Component (imageDisplay)

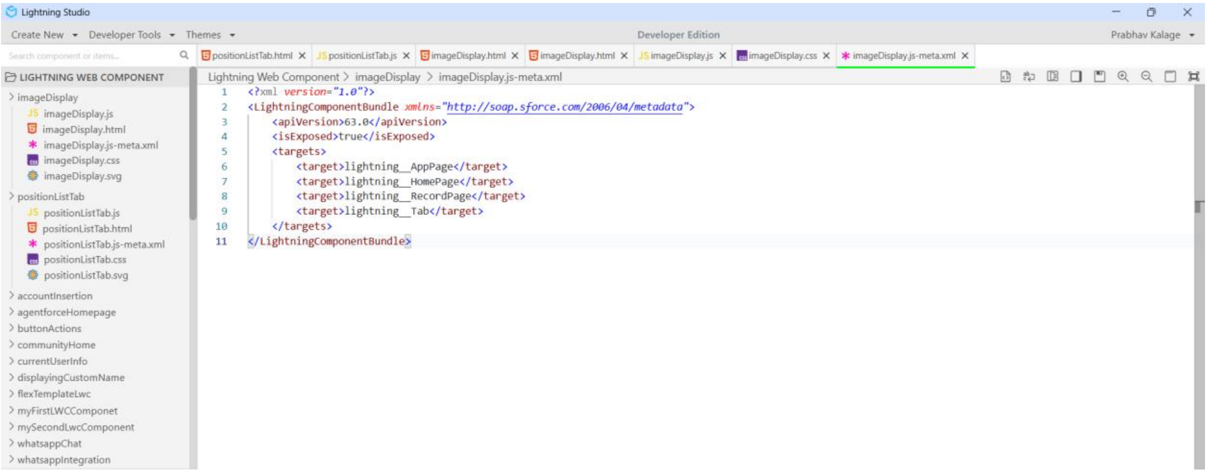
Displays a static image from Salesforce static resources with fixed dimensions (288×175 px) in a compact, centered layout.

Implementation Details:

- Uses @salesforce/resourceUrl to import the image.
- HTML wraps the image in a card and container <div>.
- CSS sets width, height, centering, and optional hover effects.





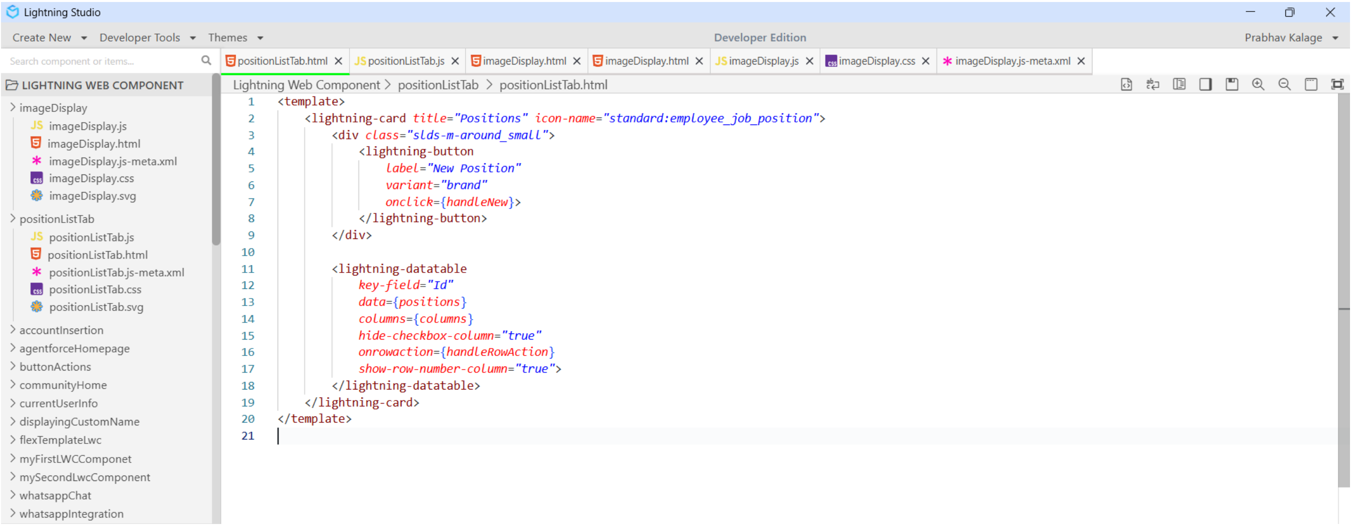


2) Position List Component (positionList)

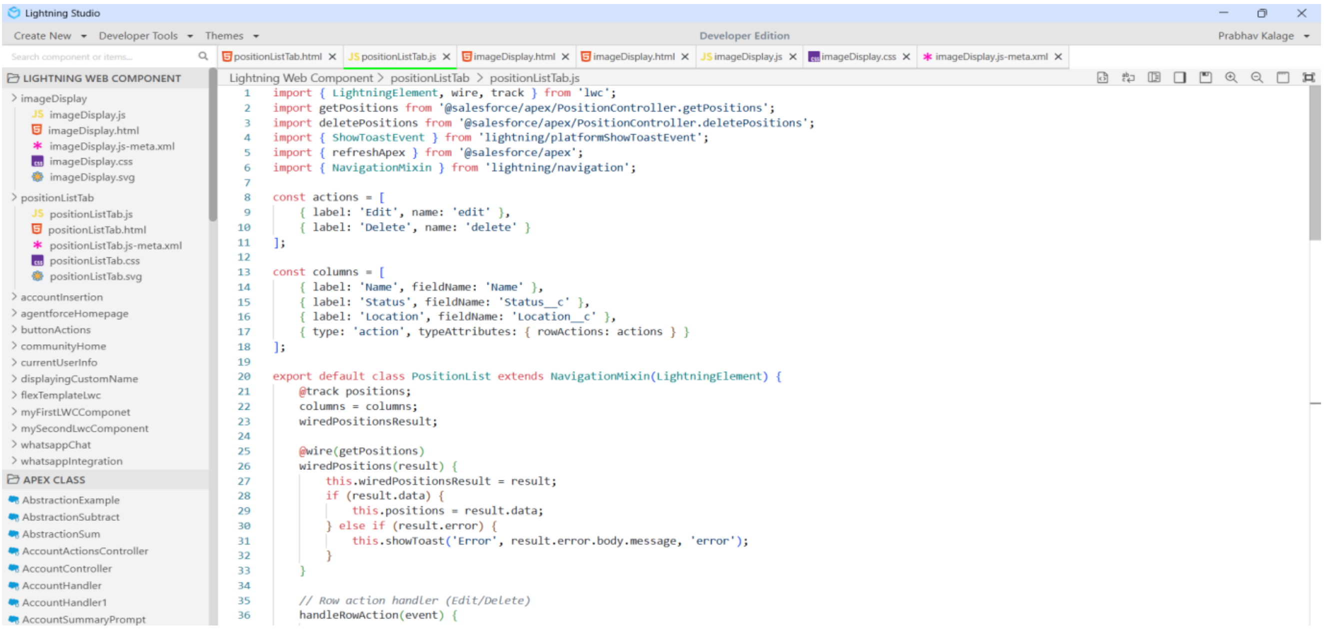
Displays a datatable of Position__c records with Edit, Delete, and New actions, allowing easy record management.

Implementation Details:

- Uses @wire to fetch records from getPositions Apex method.
- NavigationMixin handles navigation to record edit/new pages.
- Toast messages show success/error notifications.
- CSS can be used to adjust layout and spacing.


Lightning Studio
Create New
Developer Tools
Themes
Developer Edition
Prabhav Kalage
positionListTab.html
positionListTab.js
imageDisplay.html
imageDisplay.html
imageDisplay.js
imageDisplay.css
imageDisplay.js-meta.xml
LIGHTNING WEB COMPONENT
imageDisplay
imageDisplay.js
imageDisplay.html
imageDisplay.js-meta.xml
imageDisplay.css
imageDisplay.svg
positionListTab
positionListTab.js
positionListTab.html
positionListTab.js-meta.xml
positionListTab.css
positionListTab.svg
accountInsertion
agentforceHomepage
buttonActions
communityHome
currentUserInfo
displayingCustomName
flexTemplateLwc
myFirstLWCComponet
mySecondLwcComponent
whatsappChat
whatsappIntegration
Lightning Web Component > positionListTab > positionListTab.html
<template>
    <lightning-card title="Positions" icon-name="standard:employee_job_position">
        <div class="slds-m-around_small">
            <lightning-button
                label="New Position"
                variant="brand"
                onclick={handleNew}>
            </lightning-button>
        </div>

        <lightning-datatable
            key-field="Id"
            data={positions}
            columns={columns}
            hide-checkbox-column="true"
            onrowaction={handleRowAction}
            show-row-number-column="true">
        </lightning-datatable>
    </lightning-card>
</template>


Lightning Studio
Create New
Developer Tools
Themes
Developer Edition
Prabhav Kalage
positionListTab.html
positionListTab.js
imageDisplay.html
imageDisplay.html
imageDisplay.js
imageDisplay.css
imageDisplay.js-meta.xml
LIGHTNING WEB COMPONENT
APEX CLASS
AbstractionExample
AbstractionSubtract
AbstractionSum
AccountActionsController
AccountController
AccountHandler
AccountHandler1
AccountSummaryPrompt
Lightning Web Component > positionListTab > positionListTab.js
import { LightningElement, wire, track } from 'lwc';
import getPositions from '@salesforce/apex/PositionController.getPositions';
import deletePositions from '@salesforce/apex/PositionController.deletePositions';
import { ShowToastEvent } from 'lightning/platformShowToastEvent';
import { refreshApex } from '@salesforce/apex';
import { NavigationMixin } from 'lightning/navigation';

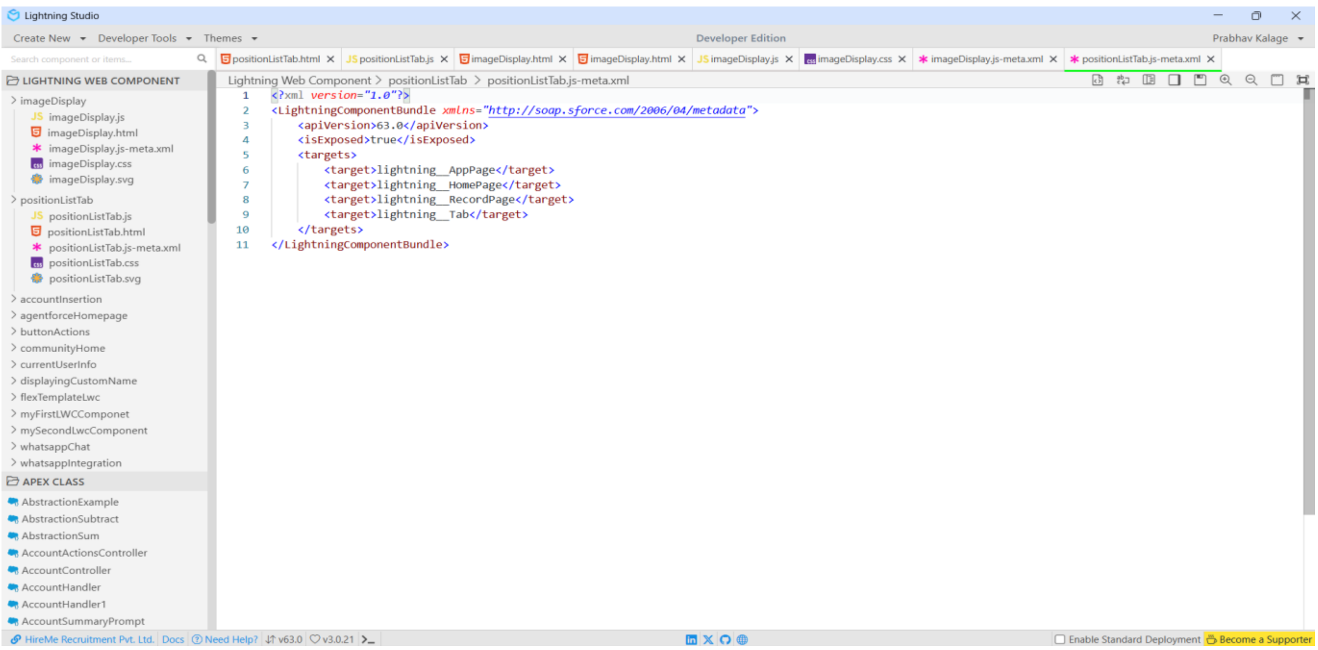
const actions = [
    { label: 'Edit', name: 'edit' },
    { label: 'Delete', name: 'delete' }
];

const columns = [
    { label: 'Name', fieldName: 'Name' },
    { label: 'Status', fieldName: 'Status__c' },
    { label: 'Location', fieldName: 'Location__c' },
    { type: 'action', typeAttributes: { rowActions: actions } }
];

export default class PositionList extends NavigationMixin(LightningElement) {
    @track positions;
    columns = columns;
    wiredPositionsResult;

    @wire(getPositions)
    wiredPositions(result) {
        this.wiredPositionsResult = result;
        if (result.data) {
            this.positions = result.data;
        } else if (result.error) {
            this.showToast('Error', result.error.body.message, 'error');
        }
    }

    // Row action handler (Edit/Delete)
    handleRowAction(event) {

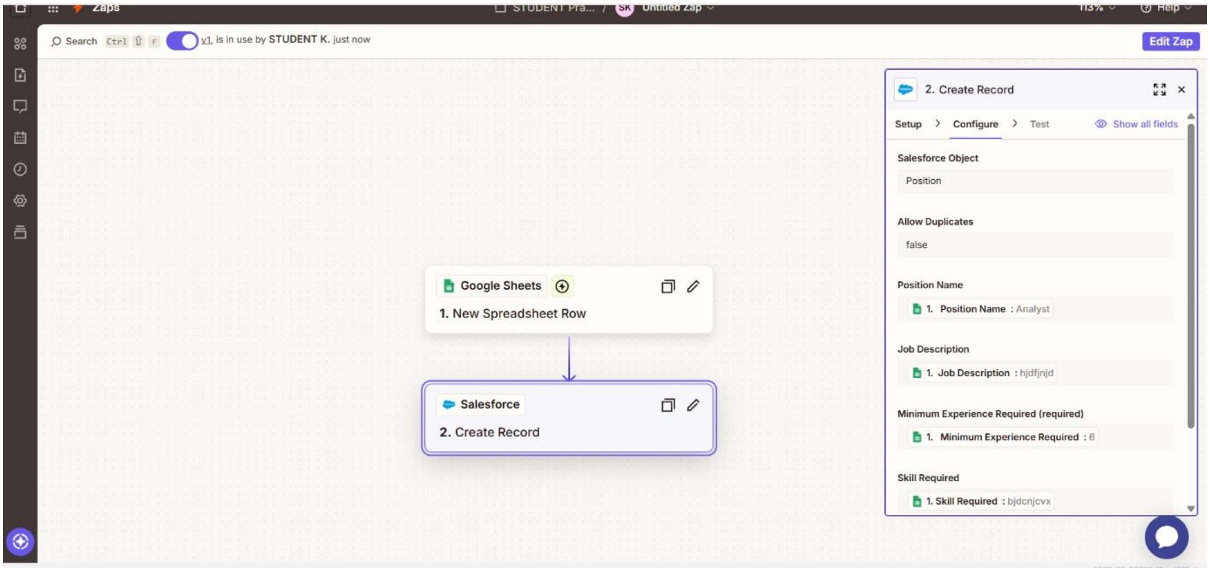

Lightning Studio
Create New
Developer Tools
Themes
Developer Edition
Prabhav Kalage
positionListTab.js-meta.xml
Lightning Web Component > positionListTab > positionListTab.js-meta.xml
<?xml version="1.0"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>63.0</apiVersion>
    <isExposed>true</isExposed>
    <targets>
        <target>lightning__AppPage</target>
        <target>lightning__HomePage</target>
        <target>lightning__RecordPage</target>
        <target>lightning__Tab</target>
    </targets>
</LightningComponentBundle>
HireMe Recruitment Pvt. Ltd.
Docs
Need Help?
v63.0
v3.0.21
Enable Standard Deployment
Become a Supporter

Phase 7: Integration & External Access

This phase focused on integrating Salesforce with external systems and enabling secure, automated data exchange.

Google Form Integration with Salesforce using Zapier:

1. **Create Google Form:** Collect responses (e.g., Job Applications).
2. **Setup Zapier:** Create a new Zap to automate data transfer.
   - **Trigger:** Google Form submission.
   - **Action:** Salesforce "Create Record" in the Position object.
3. **Configure Salesforce in Zapier:** Authenticate via **OAuth**.
4. **Map Fields:** Map Google Form fields (Name, Location, Experience, etc.) to Salesforce Position fields.
5. **Test & Activate:** Each new form submission automatically creates a Salesforce record.

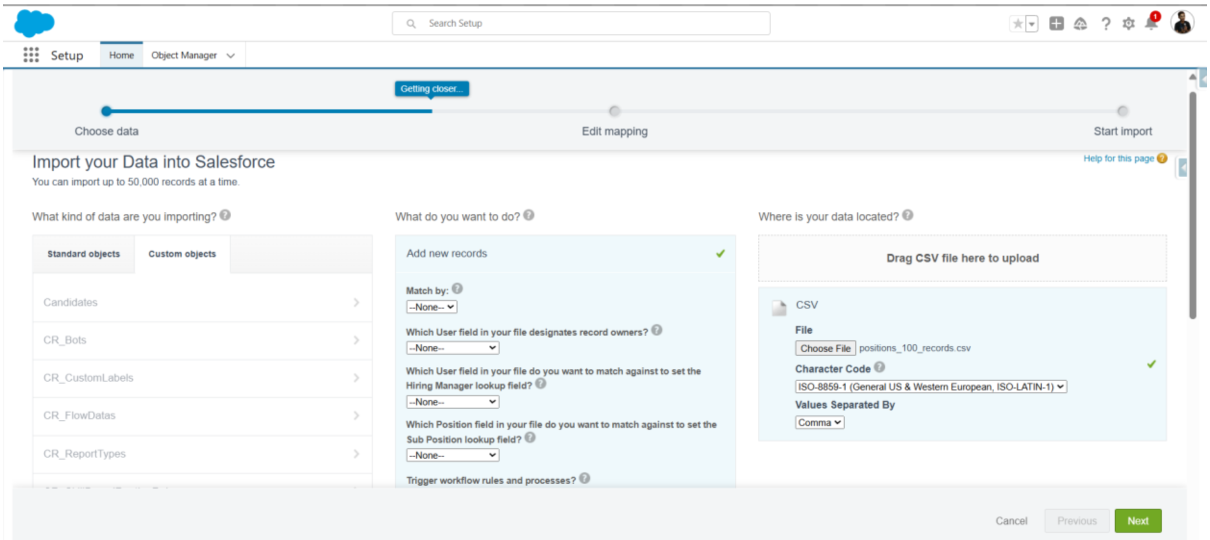


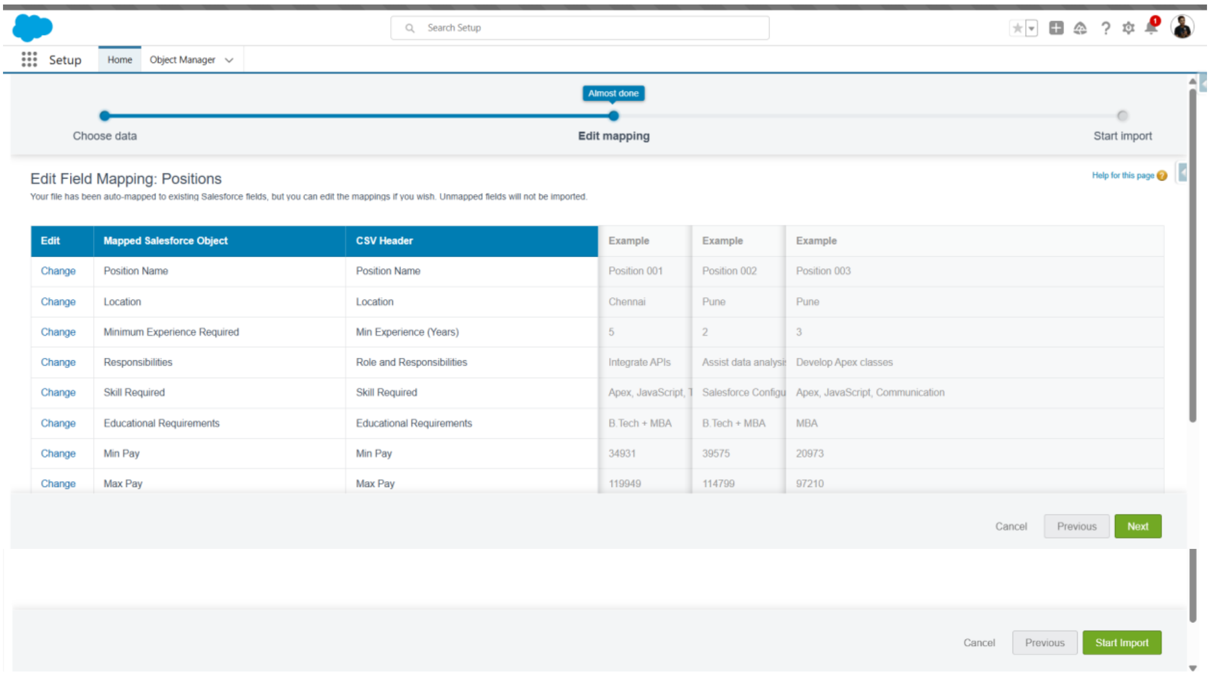
Phase 8: Data Management & Deployment

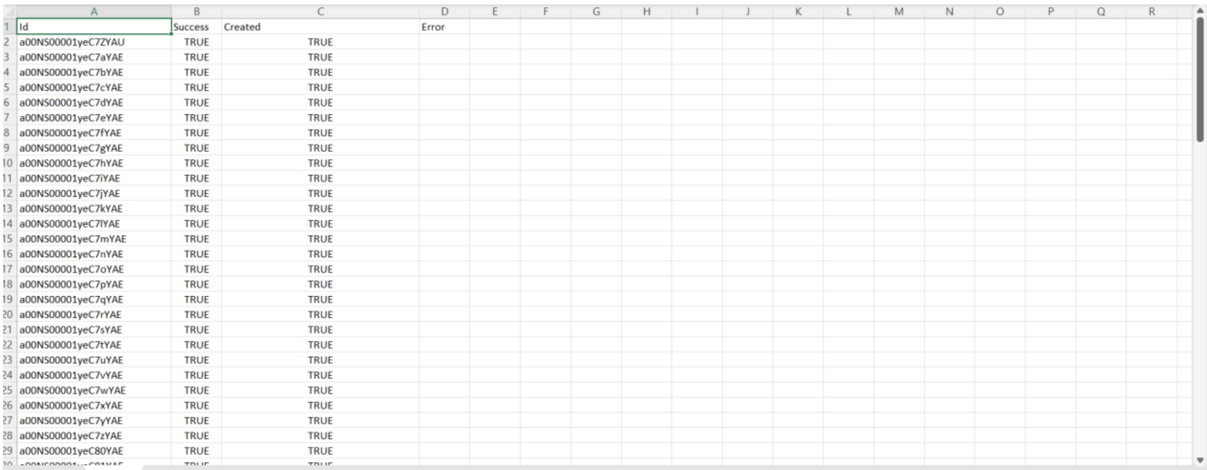
1. Data Import Wizard

Using Salesforce Data Import Wizard, I successfully inserted 100 records for the Position object. The import included fields such as Position Name, Owner, Type, Currency, Location, Status, Experience, Open/Close Dates, Role & Responsibilities, Skills, Educational Requirements, Pay Range, Travel Requirement, and Other Information.

All records were validated for accuracy, and the import process completed without errors, ensuring the data is ready for use in Salesforce workflows, reporting, and assignment processes.







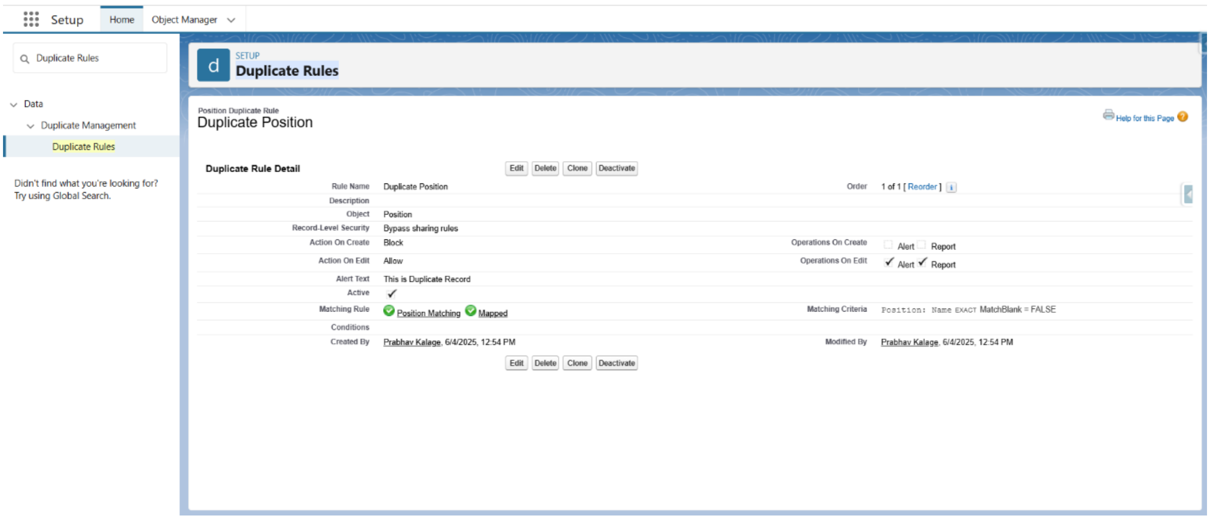
Duplicate Rule :

Duplicate Rule named “Duplicate Position” has been configured for the Position object in Salesforce to prevent duplicate entries.

- Purpose: Ensure data integrity by blocking duplicate Position records.
- Rule Behavior:
  - On Create: Block duplicate records
  - On Edit: Allow, with alert and report generation
- Alert Message: *This is Duplicate Record*

- Matching Logic: Uses Position Matching rule to compare Position Name for exact matches.
- Field Mapping: Confirmation mapping applied.
- Conditions: No additional filters; applies to all Position records.

This setup helps maintain clean and accurate Position data while notifying users of duplicates and allowing controlled updates.



Phase 9: Reporting, Dashboards & Security Review

In this phase, the focus was on generating insights, visualizing data, and ensuring secure access to reports and dashboards. Salesforce's robust reporting features enabled effective tracking of business metrics, while dashboards provided real-time visibility into operations.