

Project Title: SkillMatch Connect-Smart Skill-Based Student Job Recommendation System

Project Overview:

SkillMatch Connect is a Salesforce-based intelligent job recommendation system that connects students, educational institutions, and employers through skill-focused matchmaking. The platform enables students to register their profiles, showcase their skills, and receive personalized job or internship recommendations based on real-time analysis. Employers can post openings with specific skill requirements, while the system automatically maps qualified candidates using Salesforce automation and AI-powered logic. It manages the full recruitment lifecycle—from application to selection—while offering transparency, detailed analytics, and scalable integration for institutions and organizations.

Objectives

- Enable students to create profiles and register their skills, qualifications, and career interests.
- Allow companies and institutions to post job and internship listings with skill criteria.
- Automate skill-based job matching using Salesforce CRM data models and automation tools.
- Provide real-time notifications and application status updates to students via email.
- Generate analytical dashboards and reports for administrators and educational partners.
- Ensure a secure, scalable, and easily maintainable CRM environment to support continuous student-employer engagement.

Phase 1: Problem Understanding & Industry Analysis

Requirement Gathering

- Conducted sessions with NGOs, Youth Coordinators, and Employers.
- Core requirements:
 1. Youth registration with basic details and skills.
 2. Employer job postings.
 3. Auto-matching of youth ↔ jobs based on skills.
 4. Automatic interview creation and tracking.

5. Email notifications at each step.
6. Reports and dashboards for monitoring.

Scope: Digital bridge between youth, Admins and employers.

Stakeholder Analysis

- **Primary:** Youth, Employers, NGOs/Placement Coordinators.
- **Secondary:** System Administrators, Salesforce Platform.

Business Process Mapping

Current Process (Manual): Manual resume submission → NGOs connect youth → Communication gaps → No centralized reporting.

Proposed Process (Automated via Salesforce): Youth registration → Employer job postings → Auto-matching → Interviews scheduled → Emails sent → Dashboards.

Industry-Specific Use Case Analysis

- Existing job portals are generic and not NGO-specific.
- CareerBridge centralizes youth/job data, provides skill-based matching, and automated interview lifecycle.

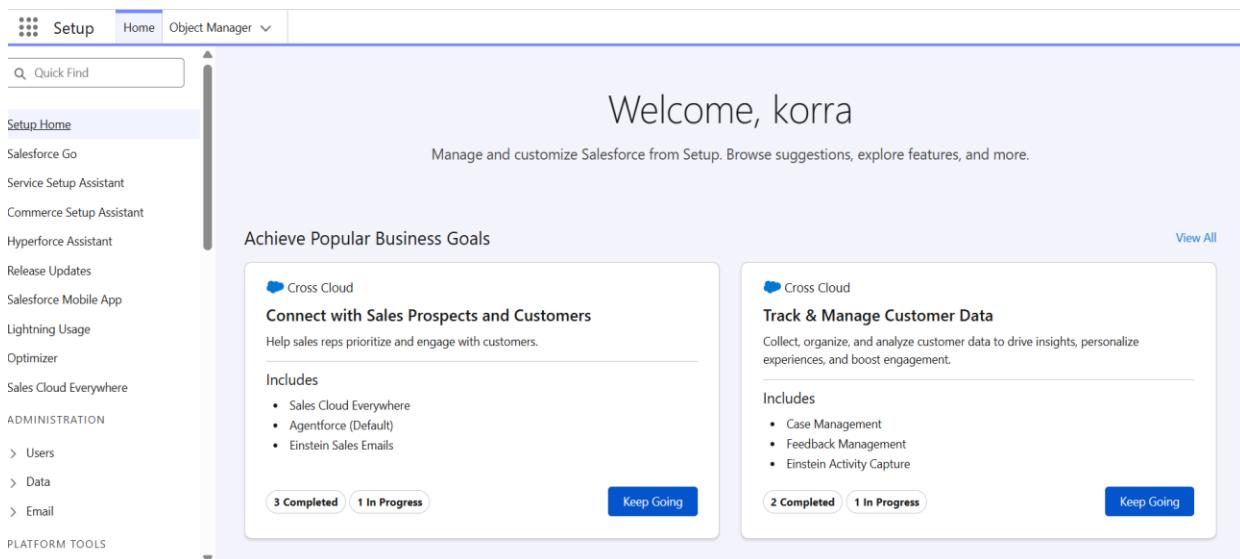
AppExchange Exploration

- Explored recruitment apps; none provide NGO-focused youth employability features.
- Decision: Build custom solution using Salesforce objects (Youth, Job, Skill, Interview), flows, and custom emails.

Phase 2: Org Setup & Configuration

Step 1: Open Setup

1. Login to **Salesforce Lightning**.
2. Click the **Gear (⚙️)** icon in the top-right → select **Setup**.



Step 2: Update Company Information

1. In Setup, use **Quick Find** → type **Company Information** → open it.
2. Click **Edit**.
3. Update:
 - **Organization Name:** KSRM College of Engineering
 - **Default Time Zone:** (09:00 – 18:00)
4. Click **Save**

The screenshot shows the 'Company Information' setup page for the organization 'KSRMCE'. The left sidebar has a search bar and navigation links for Company Settings, Business Hours, Calendar Settings, Public Calendars and Resources, and Company Information. The main content area displays the organization's profile with tabs for User Licenses, Permission Set Licenses, Feature Licenses, and Usage-based Entitlements. The 'Organization Detail' section contains fields for Organization Name (KSRMCE), Primary Contact (OrgFarm EPIC), Division, Address (United States, January), Fiscal Year Starts In, Activate Multiple Currencies, Enable Data Translation, Newsletter, Admin Newsletter, Hide Notices About System Maintenance, Hide Notices About System Downtime, Locale Formats (ICU), and various system statistics like API Requests and Restricted Logins.

Step 3: Set Business Hours

1. Quick Find → Business Hours → click New.
2. Fill in:
 - **Name:** Default Hours
 - **Hours:** 09:00 – 18:00 (or your actual business hours)
3. Click Save.

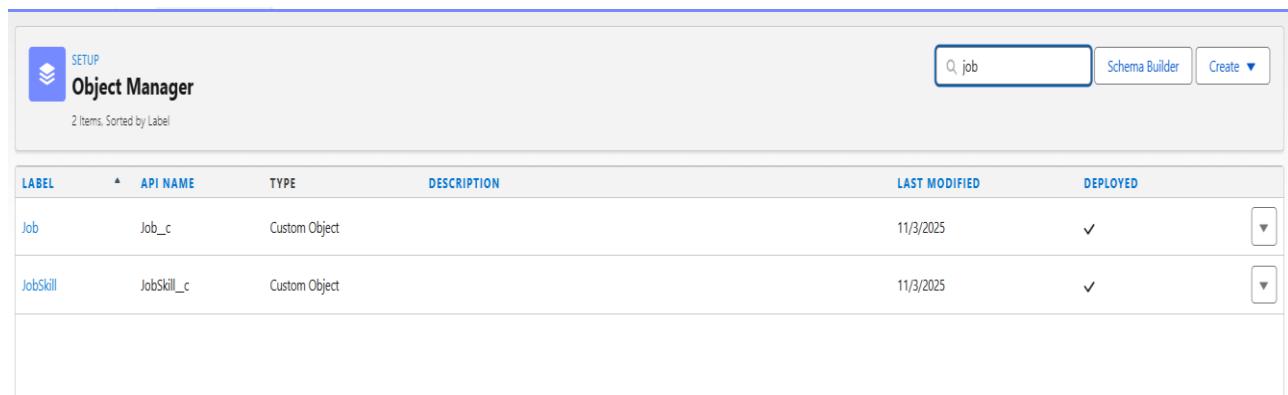
Why: Defines working hours for workflows, notifications, and approval processes.

The screenshot shows the 'Object Manager' setup page. The top navigation bar includes 'Setup', 'Home', and 'Object Manager'. The main content area displays a table of objects with columns for Label, API Name, Type, Description, Last Modified, and Deployed. There are two entries: 'Youth' (API Name: Youth_c, Type: Custom Object, Last Modified: 11/3/2025, Deployed: ✓) and 'YouthSkill' (API Name: YouthSkill_c, Type: Custom Object, Last Modified: 11/3/2025, Deployed: ✓). A search bar at the top right allows filtering by label.

Step 4: Add Holidays

1. Quick Find → **Holidays** → click **New**.
2. Add important dates, for example:
 - Independence Day
 - Republic Day
3. Click **Save**.

Why: Salesforce respects holidays for time-dependent automation (tasks, emails).



The screenshot shows the Salesforce Object Manager. At the top, there's a header with 'SETUP' and 'Object Manager'. Below it is a search bar with '0 job' and buttons for 'Schema Builder' and 'Create ▾'. A message says '2 Items. Sorted by Label'. The main area is a table with columns: LABEL, API NAME, TYPE, DESCRIPTION, LAST MODIFIED, and DEPLOYED. It lists two items: 'Job' (API Name: Job_c, Type: Custom Object) and 'JobSkill' (API Name: JobSkill_c, Type: Custom Object). Both were last modified on 11/3/2025 and have a checked 'DEPLOYED' status.

LABEL	API NAME	TYPE	DESCRIPTION	LAST MODIFIED	DEPLOYED
Job	Job_c	Custom Object		11/3/2025	✓
JobSkill	JobSkill_c	Custom Object		11/3/2025	✓

Step 5: Create Test Users

You need at least **3 users**: Youth Coordinator, Employer, NGO Admin.

1. Quick Find → **Users** → click **New User**.
2. Fill example details

Name	Username	Profile	Role
Youth Coordinator	your.email+yc@sandbox.my.salesforce.com	System Administrator	Youth Coordinator
Employer	employer@sandbox.my.salesforce.com	Standard User	Employer
NGO Admin	ngo@sandbox.my.salesforce.com	Standard User	NGO Manager

3. Click **Save** after each user.

The screenshot shows the Salesforce 'All Users' page under the 'SETUP' tab. The page title is 'Users'. It displays a list of users with columns for Action, Full Name, Alias, Username, Role, Active, and Profile. The users listed are:

Action	Full Name	Alias	Username	Role	Active	Profile
<input type="checkbox"/>	admin_Placement	padmi	placement@admin.com	SVP_Human Resources	✓	System Administrator
<input type="checkbox"/>	admin_Youth	yadmi	tooth.admin@my.com		✓	System Administrator
<input type="checkbox"/>	Chatter_Expert	Chatter	chatty.00dg50000012imkeaa.dxus5joe1po@chatter.salesforce.com		✓	Chatter Free User
<input type="checkbox"/>	Coordinator_Youth	ycoor	youth@coordinator.com	CEO	✓	Standard Platform User
<input type="checkbox"/>	EPIC_OrgFarm	QEPI	epic.fsb3b2bd790.eorgfarm.salesforce.com		✓	System Administrator
<input type="checkbox"/>	sunil kumar_korra	kor	korrasunilkumar04509@agentforce.com		✓	System Administrator
<input type="checkbox"/>	User_Integration	integ	integration@00dg50000012imkeaa.com		✓	Analytics Cloud Integration User
<input type="checkbox"/>	User_Security	sec	insightsecurity@00dg50000012imkeaa.com		✓	Analytics Cloud Security User

Step 6: Configure Profiles (Object Permissions)

1. Quick Find → **Profiles** → open a profile (e.g., *Standard User*).
2. Scroll to **Object Settings** → select object *Youth__c* → click **Edit**.
3. Grant permissions:
 - **Read, Create, Edit** (as required)

4. Click Save

The screenshot shows the Salesforce Setup interface under the Profiles section. It is titled "Field-Level Security for profile Standard User". The table lists various fields such as Field Name, Created By, Email, Last Modified By, Location, Owner, Preferred Role, Qualification, and Youth ID, along with their field types (Lookup, Email, Text, etc.) and access rights (Read Access and Edit Access). A "Help for this Page" link is visible in the top right corner.

Step 7: Create Role Hierarchy

1. Quick Find → Roles → click Set Up Roles → Add Role.

2. Example hierarchy:

CareerBridge

└─ NGO Manager

└─ Youth Coordinator

└─ Recruiter

3. Click Save.

The screenshot shows the Salesforce Setup interface under the Roles section. It is titled "Creating the Role Hierarchy". The page displays a tree view of roles under "Your Organization's Role Hierarchy". The tree includes nodes for "KSRMCE", "CareerBridge", "CEO", "CareerBridge1", "CFO", "CFO1", "COO", and "Sales", each with "Edit", "Del", and "Assign" buttons. A "Show in tree view" link is located in the top right corner.

Step 8: Permission Sets

1. Quick Find → **Permission Sets** → click **New**.
2. Label: Interview Scheduler PS → **Save**.
3. Click **Manage Assignments** → assign to users who need extra permissions.

Why: Grants additional permissions without changing profiles.

Step 9: Set Org-Wide Defaults (OWD) & Sharing

1. Quick Find → **Sharing Settings** → click **Edit**.
2. Set defaults:

Object	Default Access
---------------	-----------------------

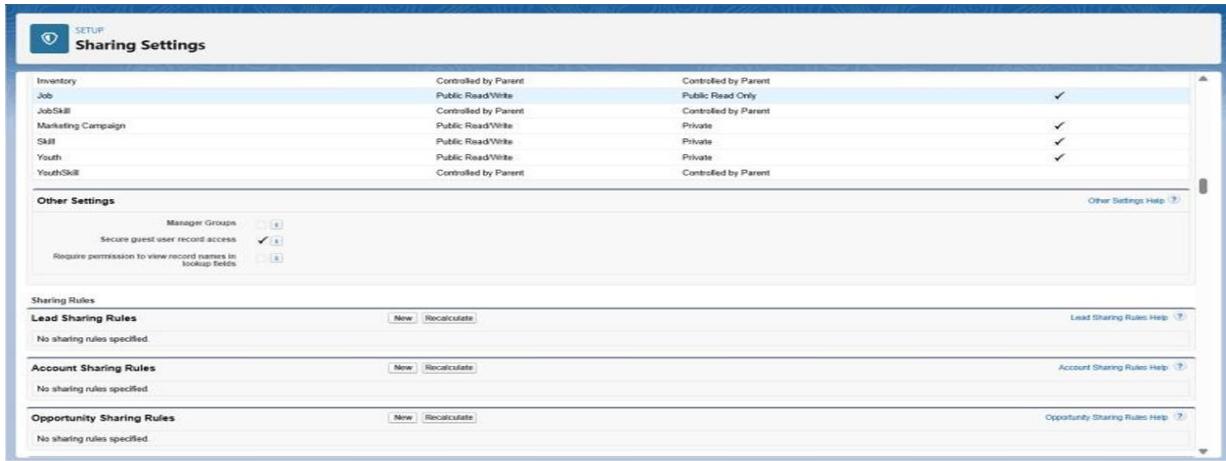
Youth__c	Private
----------	---------

Job_c	Public Read Only
-------	------------------

Interview_c	Private
-------------	---------

1. Create a **Sharing Rule**:

- Scroll to **Youth__c** → click **New**.
- Name: Share Youth to Recruiters
- Rule Type: Owner-based
- Shared To: Role → Recruiter
- Click **Save**.



Step 10: Enable Admin Login Access

1. Quick Find → **Users** → select a user → click the down-arrow → **Login**.
2. If option missing: Setup → **Login Access Policies** → enable.

Why: Admins can log in as other users for testing.

Phase 3: Data Modeling & Relationships

This phase covers creating custom objects, fields, relationships, layouts, and adding sample data.

Step 1: Create Custom Objects

Navigation:

Setup → Object Manager → Create → Custom Object

Objects to create:

Object Name	API Name	Purpose
-------------	----------	---------

Youth	Youth_c	Tracks youth registration info
Job	Job_c	Tracks job postings

Skill	Skill_c	Stores skill names like Java, Python
Interview	Interview_c	Stores interview details
YouthSkill	YouthSkill_c	Links Youth → Skill (Master-Detail on Youth, Lookup to Skill)
JobSkill	JobSkill_c	Links Job → Skill (Master-Detail on Job, Lookup to Skill)

The screenshot shows the Salesforce Object Manager interface. At the top, there are tabs for Setup, Home, and Object Manager, with Object Manager selected. A search bar contains the text "you". Below the header, it says "Object Manager" and "2 items. Sorted by Label". A table lists three custom objects:

Label	API Name	Type	Description	Last Modified	Deployed
Youth	Youth_c	Custom Object		11/3/2025	✓
YouthSkill	YouthSkill_c	Custom Object		11/3/2025	✓

The screenshot shows the Salesforce Object Manager interface. At the top, there are tabs for Setup, Home, and Object Manager, with Object Manager selected. A search bar contains the text "job". Below the header, it says "Object Manager" and "2 items. Sorted by Label". A table lists two custom objects:

Label	API Name	Type	Description	Last Modified	Deployed
Job	Job_c	Custom Object		11/3/2025	✓
JobSkill	JobSkill_c	Custom Object		11/3/2025	✓

The screenshot shows the Salesforce Object Manager interface. At the top, there are tabs for Setup, Home, and Object Manager, with Object Manager selected. A search bar contains the text "inter". Below the header, it says "Object Manager" and "2 items. Sorted by Label". A table lists two custom objects:

Label	API Name	Type	Description	Last Modified	Deployed
Interview	Interview_c	Custom Object		11/3/2025	✓
Interview_Skill	Interview_Skill_c	Custom Object		11/3/2025	✓

Step 2: Create Fields

Each object needs fields to capture record details.

Youth_c Fields

Field Name	Type	Notes
Email_c	Email	Email address of youth
Location_c	Text(255)	City/region
Qualification_c	Text	Education level

Preferred_Role__c Picklist Options: Developer, Designer, Sales

Job__c Fields

Field Name	Type	Notes
Location__c	Text(255)	City/region of job
Description__c	Long Text	Job description

Required_Experience__c Number/Text Experience in years

Status__c	Picklist	Options: Open, Closed
-----------	----------	-----------------------

Interview__c Fields

Field Name	Type	Notes
Interview_Date__c	DateTime	Scheduled date/time
Status__c	Picklist	Scheduled, Completed, Cancelled, Feedback_Given
Candidate__c	Lookup → Youth__c	Links to the youth being interviewed
Job__c	Lookup → Job__c	Links to the job
Interviewer__c	Lookup → User	Recruiter/interviewer
Feedback__c	Long Text Area	Interview feedback

The screenshot shows the Salesforce Object Manager interface for the 'Youth' object. The left sidebar lists various setup options like Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, etc. The main area displays the 'Fields & Relationships' section. It shows 8 items sorted by Field Label. The table includes columns for FIELD LABEL, FIELD NAME, DATA TYPE, CONTROLLING FIELD, and INDEXED. The fields listed are:

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedById	Lookup(User)		
Email	Email_c	Email		
Last Modified By	LastModifiedById	Lookup(User)		
Location	Location__c	Text(255)		
Owner	OwnerId	Lookup(User,Group)		
Preferred Role	Preferred_Role__c	Picklist		
Qualification	Qualification__c	Text(100)		
Youth ID	Name	Auto Number		

Fields & Relationships					
	FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Page Layouts	Created By	CreatedBy	Lookup(User)		
Lightning Record Pages	Description	Description_c	Long Text Area(32768)		
Buttons, Links, and Actions	Job Name	Name	Text(80)	✓	✓
Compact Layouts	Last Modified By	LastModifiedBy	Lookup(User)		
Field Sets	Location	Location_c	Text(255)		
Object Limits	Owner	OwnerId	Lookup(User,Group)	✓	
Record Types	Required Experience	Required_Experience_c	Number(18, 0)		
Related Lookup Filters	Status	Status_c	Picklist		
Search Layouts					
List View Button Layout					
Restriction Rules					
Scoping Rules					
Object Access					

Fields & Relationships					
	FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Page Layouts	Candidate	Candidate_c	Lookup>Youth	✓	
Lightning Record Pages	Candidate Name	Candidate_Name_c	Formula (Text)		
Buttons, Links, and Actions	Created By	CreatedBy	Lookup(User)		
Compact Layouts	Feedback	Feedback_c	Long Text Area(32768)		
Field Sets	Interview Date	Interview_Date_c	Date/Time		
Object Limits	Interview ID	Name	Auto Number	✓	
Record Types	Interviewer	Interviewer_c	Lookup(User)	✓	
Related Lookup Filters	Job	Job_c	Lookup(Job)	✓	
Search Layouts	Job Title	Job_Title_c	Formula (Text)		
List View Button Layout	Last Modified By	LastModifiedBy	Lookup(User)		
Restriction Rules	Owner	OwnerId	Lookup(User,Group)	✓	
Scoping Rules	Result	Result_c	Picklist		
Object Access	Status	Status_c	Picklist		
Triggers					
Flow Triggers					
Validation Rules					
Conditional Field Formatting					

Step 3: Create Relationships

YouthSkill_c

- Master-Detail → Youth_c
- Lookup → Skill_c

JobSkill_c

- Master-Detail → Job_c

- Lookup → Skill_c

Step 4 : Configure Page

Layouts Youth_c Layout

- Fields: Email, Location, Qualification, Preferred Role
- Related Lists: YouthSkill_c, Interview_c

Job_c Layout

- Fields: Location, Description, Required Experience, Status
- Related Lists: JobSkill_c, Interview_c

Interview_c Layout

- Fields: Candidate, Job, Interview Date, Interviewer, Status, Feedback

Step 5 : Add Sample Records

- **Skills:** Java, Python, HTML
- **Youth:** Test Youth → Email: test@y.com, Location: Hyderabad
- **Job:** Java Developer → Location: Hyderabad, Status: Open

Link YouthSkill & JobSkill

- Test Youth → Java (YouthSkill)
- Java Developer Job → Java (JobSkill)

CareerBridge Jobs JobSkills Youths YouthSkills Skills Interviews Dashboards Reports Accounts Contacts

YouthSkills Recently Viewed

3 items • Updated a few seconds ago

	□ YouthSkill (junction) Name
1	□ Sales Engineer
2	□ Developer-python
3	□ Developer-html

New Import Assign Label

Search this list...

CareerBridge Jobs JobSkills Youths YouthSkills Skills Interviews Dashboards Reports Accounts Contacts

JobSkills Recently Viewed

2 items • Updated a few seconds ago

	□ JobSkill Name
1	□ Designer-python
2	□ Developer-html

New Import Assign Label

Search this list...

Phase 4: Process Automation (Admin)

Goal: Automate youth–job matching, interview scheduling, and email notifications, while ensuring data integrity.

Step 1: Create Email Templates

1. Click **Setup** (⚙️) → In Quick Find, type **Email Templates** → Select **Email Templates**.
2. Click **New Email Template** → Select **Lightning Email Template**.
3. Fill details like **Name** and **Subject**, then design the email body.
4. Save the template.
 - **Welcome Email** → Sent after youth registration.

The screenshot shows the 'Classic Email Templates' page in Salesforce. A new template named 'welcome youth' has been created. The 'Email Template Detail' section includes fields for Email Template Name (welcome youth), Template Unique Name (welcome_youth), Encoding (Unicode (UTF-8)), Author (korra sunil kumar [Change]), and Description. The 'Email Template' section shows the subject 'Welcome to SkillMatch-Connect' and a plain text preview message. Buttons for 'Edit', 'Delete', and 'Clone' are visible throughout the interface.

- **Interview Scheduled Email** → Sent after youth is matched with a job and interview is created.

The screenshot shows the 'Classic Email Templates' page in Salesforce. The template is named 'Interview' and has a unique name of 'Interview_scheduled'. It was created by 'korra sunil kumar' on 11/3/2025 at 8:55 PM. The email template detail section includes fields for Email Template Name, Unique Name, Encoding, Author, Description, and Created By. The preview section shows the plain text content:

```

Subject: Interview Scheduled

Plain Text Preview
Hello {[Interview__c.Candidate_Name__c]}.

Congratulations! You have been shortlisted for the {[Interview__c.Job_Title__c]} position.

Your interview is scheduled on {[Interview__c.Interview_Date__c]} {[Interview__c.Interview_Time_Only__c]}.

Please be prepared.

Best regards,
SkillMatch-Connect

```

- **Interview Completed Email** → Sent after interview status is marked as Completed.

The screenshot shows the 'Classic Email Templates' page in Salesforce. The template is named 'Interview Complete' and has a unique name of 'Interview_Complete'. It was created by 'korra sunil kumar' on 11/3/2025 at 8:58 PM. The email template detail section includes fields for Email Template Name, Unique Name, Encoding, Author, Description, and Created By. The preview section shows the plain text content:

```

Subject: Interview Completed Scheduled

Plain Text Preview
Hello {[Interview__c.Candidate_Name__c]}.

Your interview for {[Interview__c.Job__r.Name]} has been successfully marked as Completed.

Result: {[Interview__c.Result__c]}

Thank you for attending.

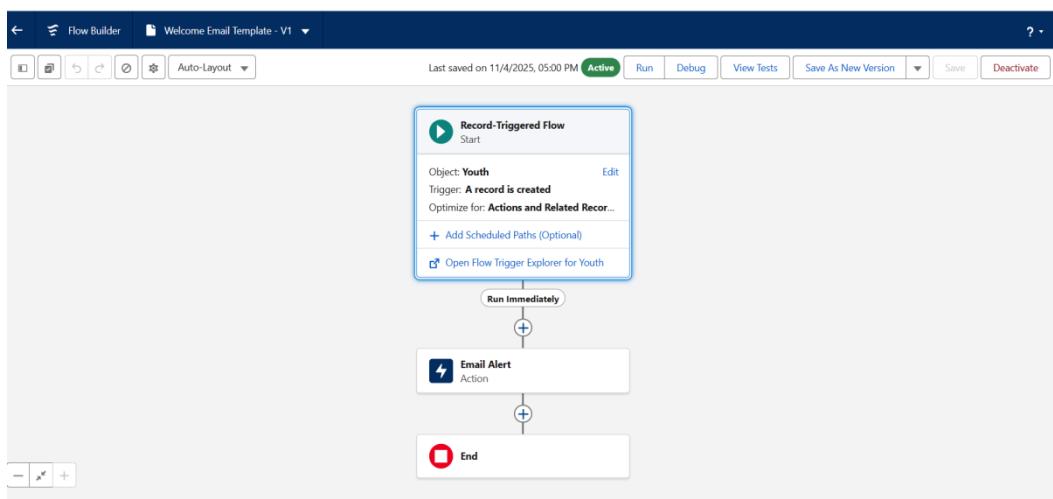
Best regards,
SkillMatch-Connect

```

Step 2: Build Flows

1. Welcome Email Flow (After Registration)

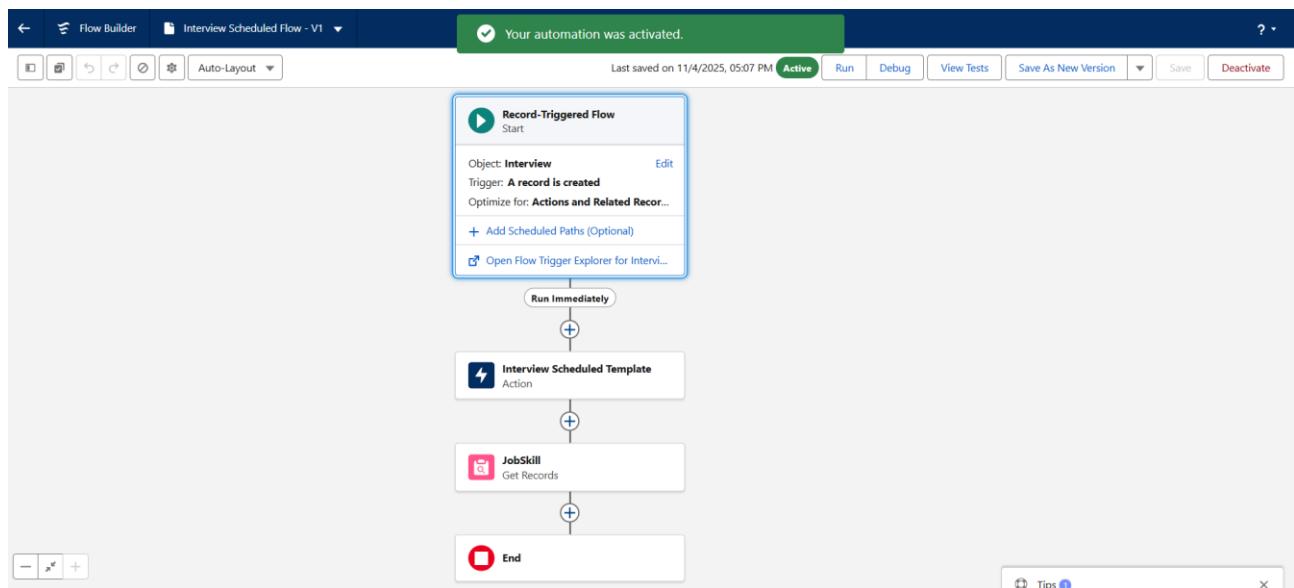
1. Click **Setup** (gear icon) → Quick Find → **Flows** → **New Flow**.
2. Select **Record-Triggered Flow**.
3. Choose **Object: Youth__c** → Trigger: **When record is created** → **After Save**.
4. Add **Action** → **Send Email** → Select **Welcome Email Template**.
5. Save and Activate.



2. Interview Scheduled Flow (After Match & Scheduling)

1. Click **Setup** → **Flows** → **New Flow**.
2. Select **Record-Triggered Flow** → **Object: Job_c** → **Trigger: Created or Updated** → **After Save**.
3. Add **Get Records** → **JobSkill_c** (fetch required skills).
4. Add **Get Records** → **YouthSkill_c** (find youth with matching skills).
5. Add **Loop** → For each Youth, **Create Interview_c record** (Candidate, Job, Status = Scheduled, Date).
6. Add **Action** → **Send Email** → Select **Interview Scheduled Template**.

Save and Activate

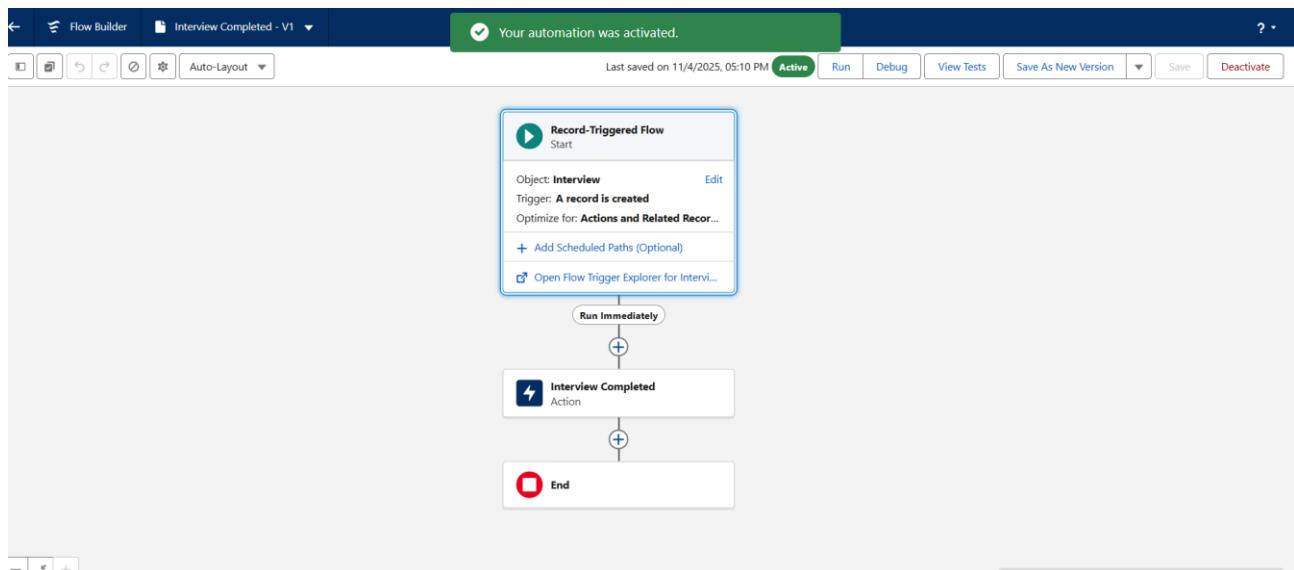


3. Interview Completed Flow (After Interview Completion)

6. Click **Setup** → **Flows** → **New Flow**.
7. Select **Record-Triggered Flow** → **Object: Interview_c**.
8. Trigger: **When record is updated** → **After Save**.
9. Add **Decision Element** → Condition: **Status_c = Completed**.

10. Add Action → Send Email → Select Interview Completed Template.

11. Save and Activate.



Step 3: Add Validation Rules

1. Click **Setup** → **Object Manager** → **Youth_c** → **Validation Rules** → New.
2. Create Rule: **Email Required**
 - Formula: ISBLANK(Email__c)
 - Error: “Email is required.”

Validation Rule Detail	
Rule Name	Email_Required
Error Condition Formula	ISBLANK(\$User.Email)
Error Message	Email is required.
Description	
Created By	korra sunil kumar, 11/4/2025, 3:49 AM
Modified By	korra sunil kumar, 11/4/2025, 3:49 AM

3. Create Rule: Location Required

- Formula: ISBLANK(Location__c)
- Error: “Location is required.”

Youth Validation Rule [Help for this Page](#)

[Back to Youth](#)

Validation Rule Detail

Rule Name	Location_Required	Edit Clone	Active <input checked="" type="checkbox"/>
Error Condition Formula	ISBLANK(Location__c)	Error Location Top of Page	
Error Message	Location is required.		
Description	Youth Validation Rule ~ Salesforce - Developer Edition	Location.	
Created By	YOUTH COORDINATOR 9/13/2025, 9:15 AM	Modified By	YOUTH COORDINATOR 9/13/2025, 9:15 AM
Edit Clone			

Phase 5: Apex Programming (Developer)

Goal:

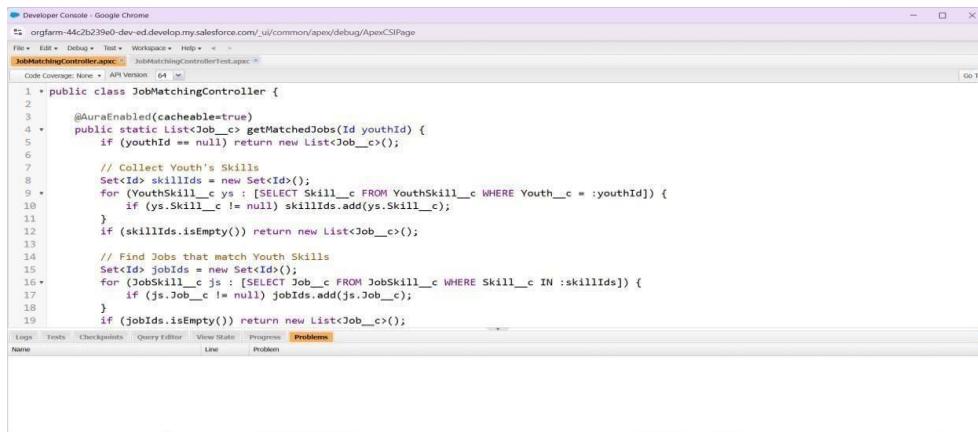
- Match youth to jobs based on skills using Apex.
- Automatically create Interview__c records.
- Handle bulk data safely and meet Salesforce deployment standards.

Step 1: Apex Class — JobMatchingController

Purpose: Perform matching and create interviews programmatically.

Steps to create Apex Class:

1. Click **Setup** → **Quick Find** → **Apex Classes** → **New**.
2. Copy and paste the JobMatchingController code below.
3. Click **Save**



The screenshot shows the Salesforce Developer Console in Google Chrome. The URL is org://44c2b239e-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCIPage. The tab title is JobMatchingController.apxc. The code editor contains the following Apex class:

```
1 * public class JobMatchingController {
2
3     @AuraEnabled(cacheable=true)
4     public static List<Job__c> getMatchedJobs(Id youthId) {
5         if (youthId == null) return new List<Job__c>();
6
7         // Collect Youth's Skills
8         Set<Id> skillIds = new Set<Id>();
9         for (YouthSkill__c ys : [SELECT Skill__c FROM YouthSkill__c WHERE Youth__c = :youthId]) {
10             if (ys.Skill__c != null) skillIds.add(ys.Skill__c);
11         }
12         if (skillIds.isEmpty()) return new List<Job__c>();
13
14         // Find Jobs that match Youth Skills
15         Set<Id> jobIds = new Set<Id>();
16         for (JobSkill__c js : [SELECT Job__c FROM JobSkill__c WHERE Skill__c IN :skillIds]) {
17             if (js.Job__c != null) jobIds.add(js.Job__c);
18         }
19         if (jobIds.isEmpty()) return new List<Job__c>();
}
```

The console interface includes tabs for Logs, Tests, Checkpoints, Query Editor, View State, Progress, and Problems. The Problems tab is currently selected.

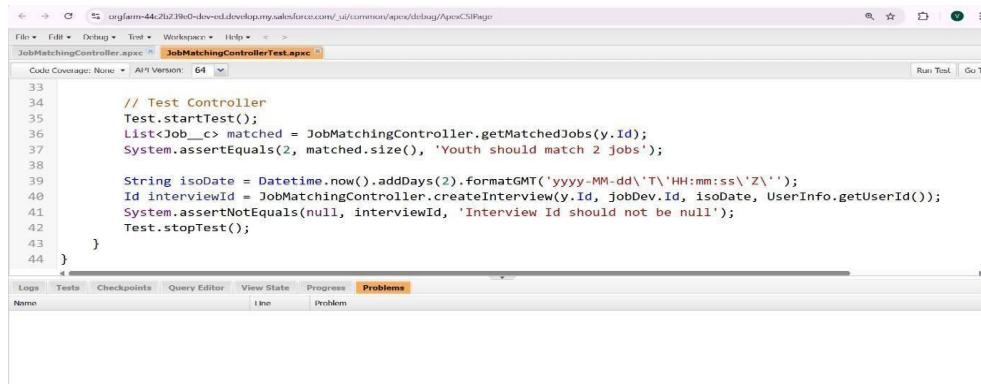
Code Logic Overview:

- **Method 1: getMatchedJobs(youthId)**
 - Fetch youth skills.
 - Find jobs requiring those skills.
 - Return matched jobs (up to 100 records).

Method 2: createInterview(youthId, jobId, interviewDateIso, interviewerId)

- Converts ISO date string → Datetime.
- Creates Interview__c record with Status__c = Scheduled.
- Returns the created interview Id.

code works for these tasks. **Small improvement:** In AutoInterviewScheduler, query COUNT() inside a loop — for very large data, consider bulkifying (collect all youth + jobs, then insert interviews in one DML).



The screenshot shows the Salesforce IDE interface with the code editor open. The file is named JobMatchingControllerTest.apxc. The code is a test class for the JobMatchingController. It includes imports for System, Test, and various Salesforce objects. The class contains a single method, testMatchedJobs, which performs assertions to verify the functionality of the getMatchedJobs method. The code uses the Test.startTest and Test.stopTest methods to run assertions within a transaction context. The code editor has syntax highlighting and line numbers. Below the code editor is a toolbar with buttons for Log, Tests, Checkpoints, Query Editor, View State, Progress, and Problems. The Problems tab is currently selected, showing no issues.

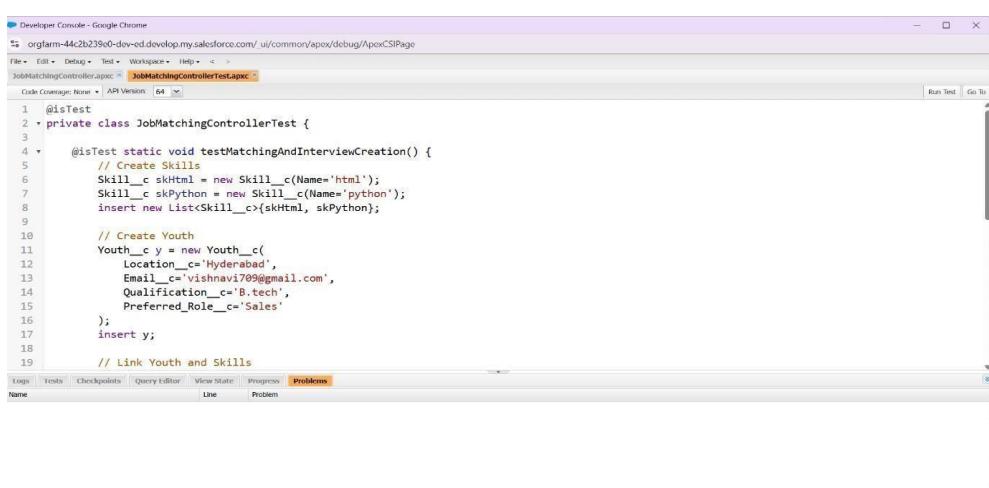
```
33
34     // Test Controller
35     Test.startTest();
36     List<Job__c> matched = JobMatchingController.getMatchedJobs(y.Id);
37     System.assertEquals(2, matched.size(), 'Youth should match 2 jobs');
38
39     String isoDate = Datetime.now().addDays(2).formatGMT('yyyy-MM-dd\'T\''HH:mm:ss'Z');
40     Id interviewId = JobMatchingController.createInterview(y.Id, jobDev.Id, isoDate, userInfo.getUserId());
41     System.assertNotEquals(null, interviewId, 'Interview Id should not be null');
42     Test.stopTest();
43 }
44 }
```

Step 2: Apex Test Class — JobMatchingControllerTest

Purpose: Ensure at least 75% coverage and test functionality.

Steps to create Test Class:

1. Click **Setup** → **Apex Classes** → **New**.
2. Copy and paste the JobMatchingControllerTest code below.
3. Click **Save**.



The screenshot shows the Salesforce Developer Console in Google Chrome. The URL is org://44c2b239e0-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCsPage. The tab title is JobMatchingController.apc. The code editor contains the following Apex test class:

```
1  @isTest
2  private class JobMatchingControllerTest {
3
4    @isTest static void testMatchingAndInterviewCreation() {
5      // Create Skills
6      Skill__c skHtml = new Skill__c(Name='html');
7      Skill__c skPython = new Skill__c(Name='python');
8      insert new List<Skill__c>{skHtml, skPython};
9
10     // Create Youth
11     Youth__c y = new Youth__c(
12       Location__c='Hyderabad',
13       Email__c='vishnav1709@gmail.com',
14       Qualifications__c='B.tech',
15       Preferred_Role__c='Sales'
16     );
17     insert y;
18
19     // Link Youth and Skills
}
```

The code editor has tabs for Log, Tests, Checkpoints, Query Editor, View State, Progress, and Problems. The Problems tab is selected, showing one problem: "Line 19: // Link Youth and Skills".

Test Logic Overview:

- Creates test Skill__c, Youth__c, YouthSkill__c, Job__c, JobSkill__c.
- Tests getMatchedJobs() → asserts correct number of jobs.
- Tests createInterview() → asserts interview Id is not null

The screenshot shows the Salesforce IDE interface with the file `JobMatchingControllerTest.apxc` open. The code is a test class for the `JobMatchingController`. A modal window titled "Success" is displayed, indicating that the test was run successfully. The test code includes assertions for matched jobs and interview IDs.

```

33
34     // Test Controller
35     Test.startTest();
36     List<Job__c> matched = JobMatchingController.getMatchedJobs(y.Id);
37     System.assertEquals(2, matched.size(), 'Youth should match 2 jobs');
38
39     String isoDate = Datetime.now().addDays(2).formatGMT('yyyy-MM-dd\''T\HH:mm:ss\''Z\'');
40     Id interviewId = JobMatchingController.createInterview(youthId, isoDate, userInfo.getUserId());
41     System.assertNotEquals(null, interviewId, 'Interview ID must not be null');
42     Test.stopTest();
43 }
44

```

Step 3: Execute Apex for Individual Youth

Steps:

1. Open **Developer Console → Execute Anonymous Window.**
2. Run this snippet (replace YID-0010 with actual youth auto-number):

The screenshot shows the Salesforce Developer Console with an anonymous window open. The code in the window is an Apex test method named `testMatchingAndInterviewCreation`. It creates a skill, a youth, and a youth skill link, then finds the youth by name, gets the matched jobs, and creates an interview for the first matched job.

```

1 *isTest static void testMatchingAndInterviewCreation() {
2     // Create Skills
3     Skill__c skHTML = new Skill__c(Name='HTML');
4     Skill__c skPython = new Skill__c(Name='Python');
5     insert new List<Skill__c>(skHTML, skPython);
6
7     // Create Youth
8     Youth__c y = new Youth__c(
9         FirstName='Vishnavi',
10        LastName='Jain',
11        Email='vishnavi799@gmail.com',
12        Qualification__c='0.tech',
13        Preferred_Role__c='Sales';
14    );
15    insert y;
16
17    // Link Youth and Skills
18    YouthSkill__c ysHTML = new YouthSkill__c(Youth__c=y, Skill__c=skHTML.Id);
19    YouthSkill__c ysPython = new YouthSkill__c(Youth__c=y, Skill__c=skPython.Id);
20    insert new List<YouthSkill__c>(ysHTML, ysPython);
21
22}

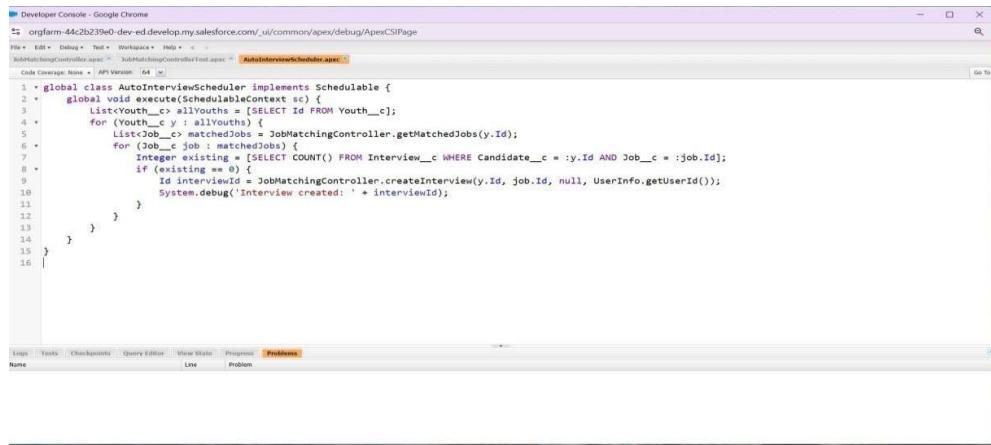
```

This works — it will create an interview for the first matched job

Step 4: Scheduler — AutoInterviewScheduler

Purpose: Automatically create interviews for all youth. **Steps to schedule:**

1. Click Setup → Apex Classes → Schedule Apex → New.
2. Choose **AutoInterviewScheduler** class.
3. Set frequency (daily/hourly) and start/end dates.
4. Click **Save** to activate scheduler.



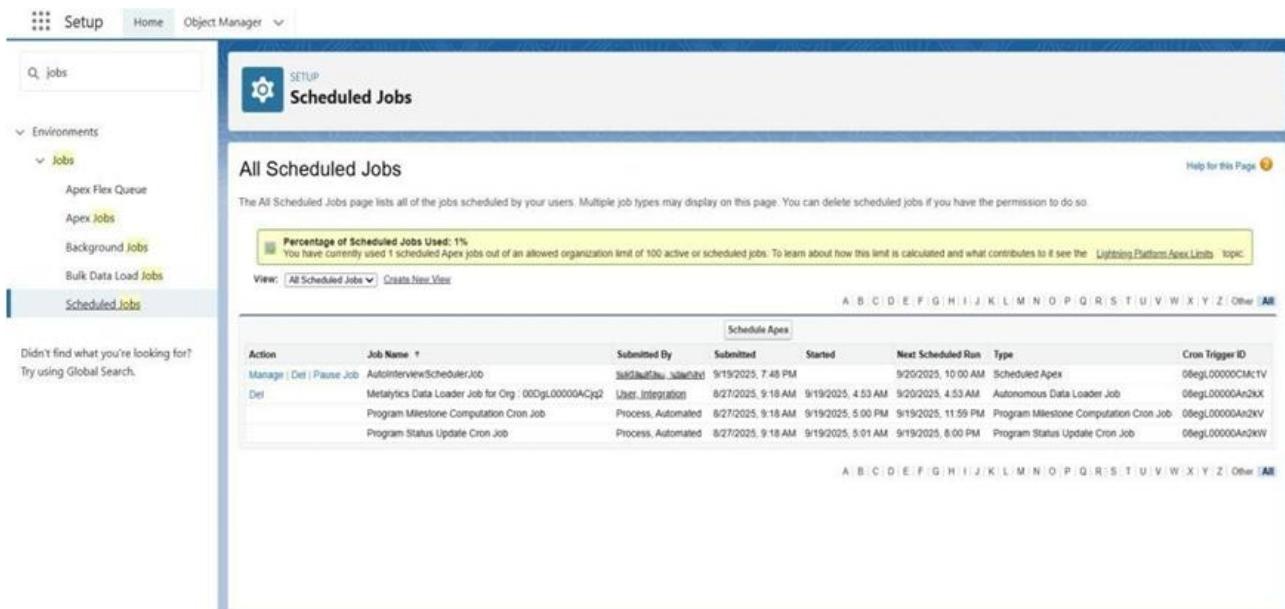
```

1  global class AutoInterviewScheduler implements Schedulable {
2      global void execute(SchedulableContext sc) {
3          List>Youth__c> allYouths = [SELECT Id FROM Youth__c];
4          for (Youth__c y : allYouths) {
5              List>Job__c> matchedJobs = JobMatchingController.getMatchedJobs(y.Id);
6              for (Job__c job : matchedJobs) {
7                  Integer existing = [SELECT COUNT() FROM Interview__c WHERE Candidate__c = :y.Id AND Job__c = :job.Id];
8                  if (existing == 0) {
9                      Id interviewId = JobMatchingController.createInterview(y.Id, job.Id, null, UserInfo.getUserId());
10                     System.debug('Interview created: ' + interviewId);
11                 }
12             }
13         }
14     }
15 }
16

```

Logic:

- Loops through all youth → gets matched jobs → creates interview if none exists.
- System.debug() logs each creation.



All Scheduled Jobs

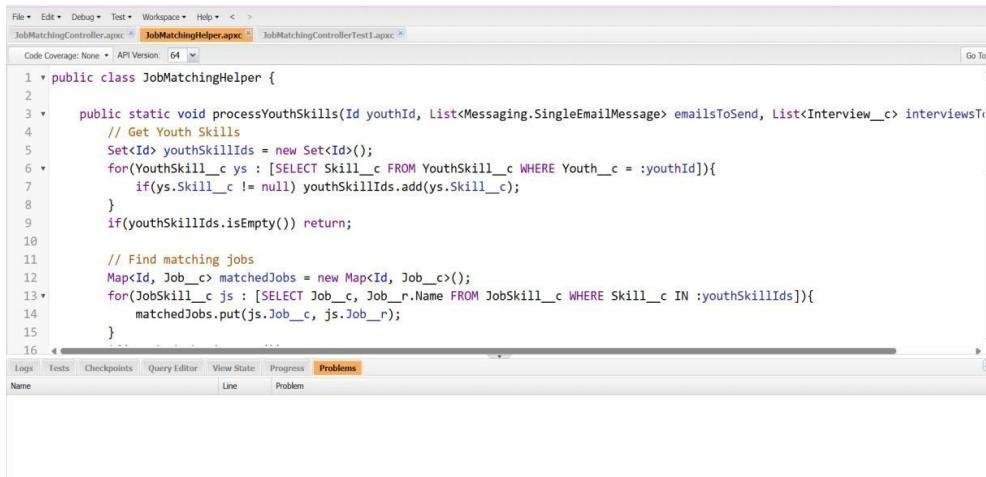
Action	Job Name	Submitted By	Started	Next Scheduled Run	Type	Cron Trigger ID
Manage Del Pause Job	AutoInterviewScheduler:Job	System User: Admin	9/19/2025, 7:48 PM	9/20/2025, 10:00 AM	Scheduled Apex	06egL00000CMctIV
Del	Metalytics Data Loader Job for Org: 00Ogl00000ACjq2	User_Integration	8/27/2025, 9:18 AM	9/19/2025, 4:53 AM	Autonomous Data Loader Job	06egL00000Aa02KX
	Program Milestone Computation Cron Job	Process, Automated	8/27/2025, 9:18 AM	9/19/2025, 5:00 PM	Program Milestone Computation Cron Job	06egL00000Aa02KV
	Program Status Update Cron Job	Process, Automated	8/27/2025, 9:18 AM	9/19/2025, 5:01 AM	Program Status Update Cron Job	06egL00000Aa02KX

Step 5: Apex Helper Class — JobMatchingHelper

Purpose Handle automatic interview creation and email sending when youth skills match jobs.

Steps to create Helper Class:

1. Click **Setup** → **Apex Classes** → **New**.
2. Copy and paste the **JobMatchingHelper** Code below.
3. Click **Save**.



The screenshot shows the Salesforce IDE interface with the code editor open. The file tab at the top says "JobMatchingController.apxc" and "JobMatchingHelper.apxc". The code editor contains the following Apex class:

```
1 * public class JobMatchingHelper {  
2  
3     public static void processYouthSkills(Id youthId, List<Messaging.SingleEmailMessage> emailsToSend, List<Interview__c> interviewsToCreate){  
4         // Get Youth Skills  
5         Set<Id> youthSkillIds = new Set<Id>();  
6         for(YouthSkill__c ys : [SELECT Skill__c FROM YouthSkill__c WHERE Youth__c = :youthId]){  
7             if(ys.Skill__c != null) youthSkillIds.add(ys.Skill__c);  
8         }  
9         if(youthSkillIds.isEmpty()) return;  
10        // Find matching jobs  
11        Map<Id, Job__c> matchedJobs = new Map<Id, Job__c>();  
12        for(JobSkill__c js : [SELECT Job__c, Job__r.Name FROM JobSkill__c WHERE Skill__c IN :youthSkillIds]){  
13            matchedJobs.put(js.Job__c, js.Job__r);  
14        }  
15    }  
16}
```

The status bar at the bottom of the code editor shows "Logs", "Tests", "Checkpoints", "Query Editor", "View State", "Progress", and "Problems". The "Problems" tab is selected. There are no problems listed in the problems list.

Phase 6: User Interface Development

Step1:Lightning App Builder

- Go to setup -> App Manager -> New Lighting App.
- Name the app CareerBridge.
- Add navigation items: Home, Youth, Job, Interviews, Reports, Dashboards
- Assign to profiles and Finish.

New Lightning App

App Details & Branding

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

App Details

*App Name

*Developer Name

Description

App Branding

Image

Primary Color Hex
Value #007002

Org Theme Options
 Use the app's image and color instead of the org's custom theme

App Launcher Preview



Lightning App Builder | App Settings | Pages | CareerBridge | ? Help

App Settings

App Details & Branding

App Options

Utility Items (Desktop Only)

Navigation Items

User Profiles

Navigation Items

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

Available Items	Selected Items
<input type="button" value="Create"/>	<input type="button" value="Create"/>
<input type="button" value="Type to filter list..."/>	<input type="button" value="Type to filter list..."/>
Activation Targets	Jobs
Activations	JobSkills
All Sites	Youths
Alternative Payment Methods	YouthSkills
Analytics	Skills
App Launcher	Interviews
Appointment Categories	Dashboards
Appointment Invitations	Reports
Approval Requests	Accounts
Approval Submission Details	Contacts

Step 2: Record Pages

- Custom layouts for specific object records (**Youth, Job, Interview, Skill**).
- You can decide what **fields, related lists, and components appear**.

Youth Record Page: Show youth details, skills, and recommended jobs.

The screenshot shows a software interface titled "CareerBridge". The top navigation bar includes links for Jobs, JobSkills, Youths, YouthSkills, Skills, Interviews, Dashboards, Reports, Accounts, and Contacts. A search bar at the top right contains the placeholder "Search...". Below the navigation is a toolbar with icons for New, Import, Change Owner, and Assign Label. A "Recently Viewed" section displays a list of 7 items, all labeled "Developer":

Job Name
1 Developer
2 Machine Learning Engineer
3 Full Stack Developer
4 Data Analyst
5 Software Engineer
6 designer
7 Developer

Job Record Page: Show job details, required skills, and linked interviews

The screenshot shows a software interface titled "CareerBridge". The top navigation bar includes links for Jobs, JobSkills, Youths, YouthSkills, Skills, Interviews, Dashboards, Reports, Accounts, and Contacts. A search bar at the top right contains the placeholder "Search...". Below the navigation is a toolbar with icons for New Contact, Edit, and New Opportunity. The main area displays a "Youth" record for "YID-0009". The "Details" tab is selected, showing the following fields:

Related	Details
Youth ID	YID-0009
Email	wawanavi709@gmail.com
Location	Hyderabad
Qualification	B.tech
Preferred Role	Developer

The "Owner" field shows a user icon and the name "Rajesh Kumar". On the right side of the screen, there is a large blue decorative graphic.

- **Interview Record Page:** Show candidate, job, date, and status.

Step 3:Tabs

- Setup → App Manager → Edit CareerBridge → Navigation Items.
- Add or reorder tabs (**YouthSkill, Youth, Jobs, Interviews, JobsSkill, Skill**).
- Save.

The screenshot shows the 'Custom Tabs' section of the Salesforce Setup. It includes a 'Custom Object Tabs' table with the following data:

Action	Label	Tab Style	Description
Edit Del	Interviews	Presenter	
Edit Del	Jobs	Presenter	
Edit Del	JobSkills	Bank	
Edit Del	Skills	Laptop	
Edit Del	Youths	People	
Edit Del	YouthSkills	Books	

Below this are sections for 'Web Tabs' (No Web Tabs have been defined) and 'Visualforce Tabs' (No Visualforce Tabs have been defined).

Step 4:Home Page Layouts

- The home page can be customized for different profiles (e.g., Youth, Admin).
- Provides dashboards, tasks, and quick actions.
 - **CareerBridge Example:**
 - For Youth: Show registered jobs, upcoming interviews, and assigned skills.
 - For Admin: Display KPIs like total youth registered, jobs posted, and interviews scheduled.

The screenshot shows the CareerBridge application's interface. At the top, there is a navigation bar with links for Jobs, JobSkills, Youths, YouthSkills, Skills, Interviews, Dashboards, Reports, Accounts, and Contacts. Below the navigation bar, a search bar is followed by a toolbar with icons for New, Import, Change Owner, and Assign Label. A secondary search bar labeled "Search this list..." is also present. The main content area displays a list titled "Recently Viewed" under the "Jobs" category. The list contains 7 items, all of which are "Developer" roles, listed from 1 to 7. Each item has a checkbox next to it. The interface uses a clean, modern design with a light blue header and white background.

Phase 7: Integration & External Access

1. API Limits

- Salesforce tracks API calls made by external apps or integrations.
- In CareerBridge, all actions (Youth registration, Job posting, Flows, Emails) happen **inside Salesforce**.
- **No external API calls are used**, so API Limits **do not affect your project**.
- You can check usage in **Setup → System Overview**, but nothing needs to be configured.

The screenshot shows the Salesforce System Overview page. The top navigation bar includes links for Setup, Home, and Start Manager. A search bar is located at the top right. The main content area is divided into several sections:

- System**: Shows various system statistics like Custom Objects (11), Custom Fields (11), and Databases (523 KB).
- API Usage**: Displays the number of API calls made, categorized by type: API Calls (3), API Requests (3), API Closed (6), and Error Rate (0.16%).
- Most Used Libraries**: Lists the most frequently used libraries: Analytics Cloud Integration User (2), Salesforce (4), and Marketing (4).
- User Metrics**: Shows user statistics for various components: Custom Apps (2), Active Salesforce Users (0), Active Groups (6), Custom Tabs (6), and Visualforce Pages (0).

 The page uses a dark-themed interface with light-colored cards for each metric, providing a clear overview of system performance and usage.

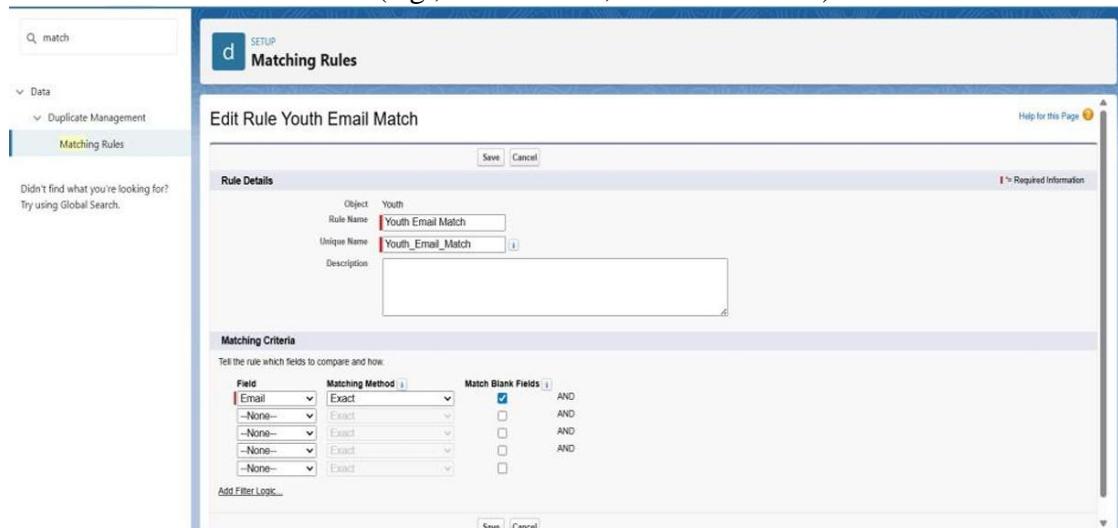
Phase 8: Data Management & Deployment

Step 1: Duplicate Management

Preventing duplicate records is critical, especially for **Youth** and **Job** objects.

A. Matching Rules

- Define the criteria to identify duplicates.
- **Steps:**
 1. Go to **Setup** → **Matching Rules** → **New**.
 2. Example: Match **Youth_c.Email_c**.
 3. Define criteria (e.g., exact match, case-insensitive) → Save → Activate.

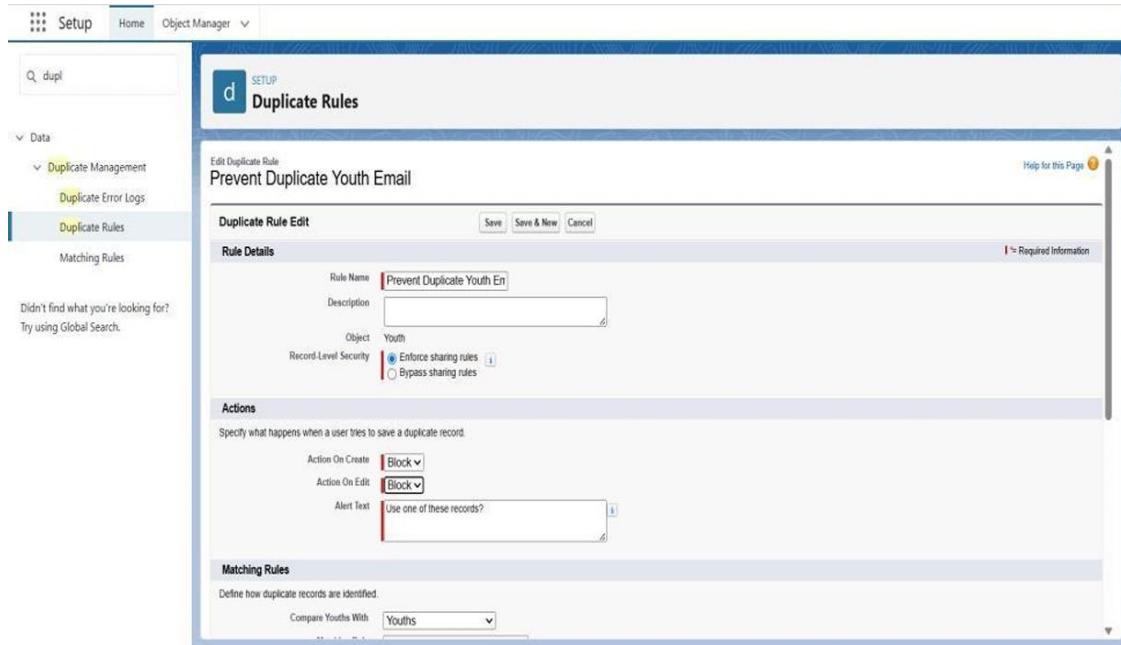


B. Duplicate Rules

- Decide what happens when duplicates are detected.
 - **Steps:**
 1. Go to **Setup** → **Duplicate Rules** → **New**.
 2. Apply your **Matching Rule** to the object (e.g., **Youth__c**).
 3. Choose **Action**:
 - **Block** → Prevents user from creating duplicates.

- **Alert** → Warns the user, but allows creation.

4. Activate the rule.



Step 2: Data Backup

Always back up before importing or deploying, in case something goes wrong.

- **Steps:**
 1. Go to **Setup** → **Data Export** → **Export Now**.
 2. Choose objects to backup (e.g., Youth__c, Job__c).
 3. Download the zip files.

Phase 9: Reporting, Dashboards & Security Review

Goal:

- Track progress through reports and dashboards
 - Protect sensitive data from unauthorized access
 - Ensure stakeholders see the right information in real time

Step 1: Reporting

Reports help track Youth, Jobs, and Interviews effectively.

A. Create Reports

1. Go to **App Launcher → Reports → New Report**
2. Select the report type: e.g., **Interview** or **Custom Report**
3. Add fields:
 - Candidate Name (Youth__c Name)
 - Job Title (Job__c Name)
 - Interview Date (Interview_Date__c)
 - Status (Status__c)
4. Save & Run the report

B. Useful Reports

- **Interviews by Status** → Group by Status__c (Shows Scheduled vs Completed interviews)
- **Interviews by Result** → Group by Result__c (Shows Qualified, Shortlisted, Rejected outcomes)
- **Interviews Over Time** → Group by Last Interview Date (Shows trends or frequency of interviews)

The screenshot shows a software application window titled "Report Interviews Interviews Report". The top navigation bar includes links for Jobs, JobSkills, Youths, YouthSkills, Skills, Interviews, Dashboards, Reports, Accounts, and Contacts. The main content area displays a table with the following data:

Status	Result	Interview Date	Candidate Name	Job
Scheduled (2)	(2)	9/16/2025 (1)	YD-0008	Developer
		Subtotal		
		9/18/2025 (1)	YD-0001	Java Developer
		Subtotal		
		Subtotal		
Completed (4)	Qualified (1)	9/18/2025 (1)	YD-0009	Developer
		Subtotal		
		Subtotal		
Rejected (2)	(2)	9/13/2025 (2)	YD-0009	Designer
		Subtotal		
		Subtotal		
Shortlisted (1)	(1)	9/1/2025 (1)	YD-0009	Designer
		Subtotal		
		Subtotal		
		Total (8)		

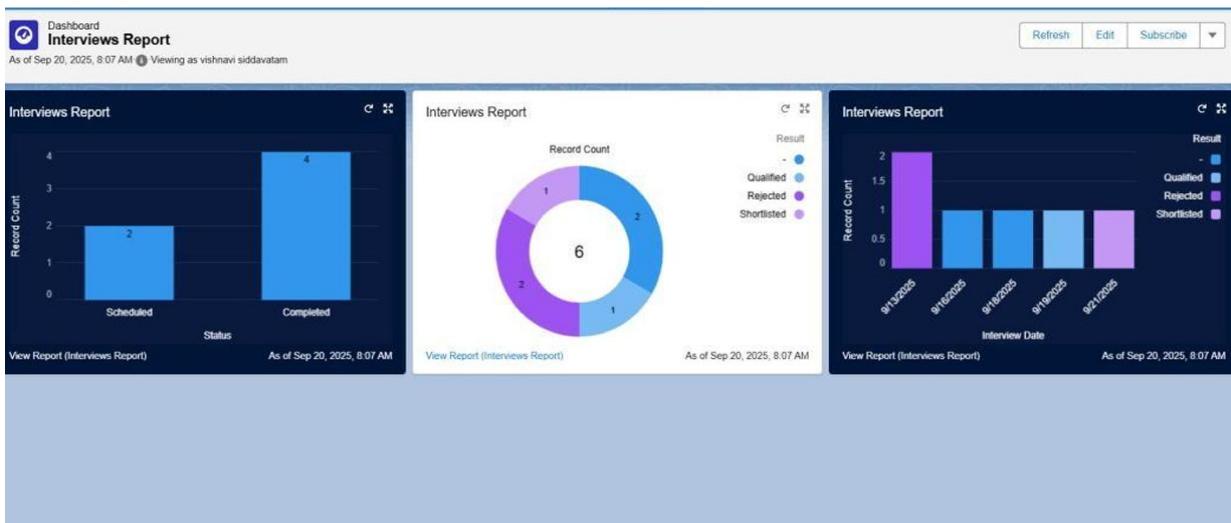
At the bottom of the table, there are four checkboxes: Raw Count (unchecked), Detail Rows (unchecked), Subtotals (checked), and Grand Total (checked).

Step 2: Dashboards

Dashboards visually display metrics from reports.

A. Create Dashboard

1. Go to **App Launcher** → **Dashboards** → **New**
2. Add components:
 - **Metric:** Count of Completed Interviews
 - **Bar Chart:** Interviews by Status
 - **Line Chart:** Interviews per Week
 - **Table:** Recent Interviews
3. Enable Dynamic Dashboards → Set “View Dashboard As = Dynamic”

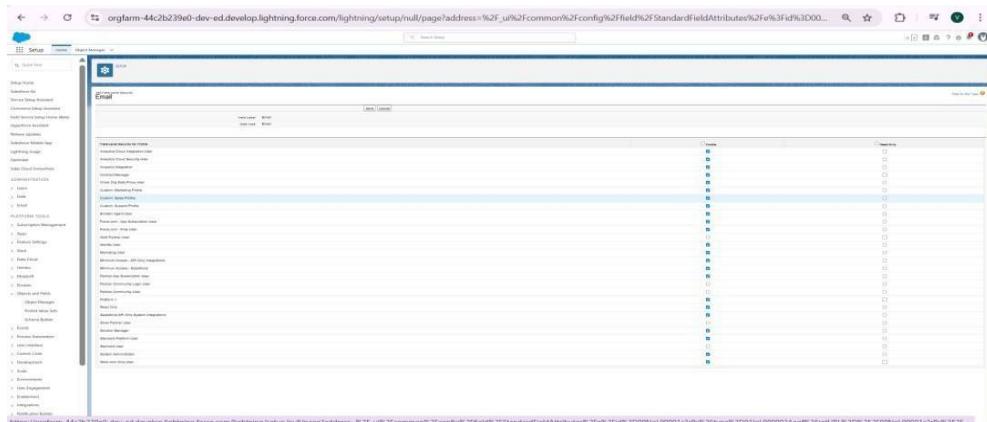


Step 3: Security

Protect sensitive data (like Youth emails, interview feedback) and control who can see what.

A. Field-Level Security (FLS)

1. Setup → Object Manager → Youth_c → Fields → Email_c → Set Field-Level Security



2. Hide fields from standard users or users without permission

B. Sharing & Org-Wide Defaults (OWD)

1. Setup → Sharing Settings
2. Ensure Youth_c = Private

C. Audit Trail

1. Setup → View Setup Audit Trail
2. Track unauthorized changes or admin activity

 **SETUP**

View Setup Audit Trail

The last 20 entries for your organization are listed below. You can [download](#) your organization's setup audit trail for the last six months (Excel .csv file).

View Setup Audit Trail

Date	User	Source Namespace Prefix	Action	Section	Delegate User ?
11/4/2025, 9:10:42 PM PST	korrasunilkumar04509@agentforce.com		Added standard button override on Youths: View (Lightning Page null)	Custom Objects	
11/4/2025, 9:09:51 PM PST	korrasunilkumar04509@agentforce.com		Created Lightning Page_Youth	Lightning Pages	
11/4/2025, 8:56:02 PM PST	korrasunilkumar04509@agentforce.com		Created custom app SkillMatch	Custom Apps	
11/4/2025, 8:56:02 PM PST	korrasunilkumar04509@agentforce.com		Created Lightning Page_SkillMatch UtilityBar	Lightning Pages	
11/4/2025, 8:54:37 PM PST	korrasunilkumar04509@agentforce.com		Deleted custom app SkillMatch	Custom Apps	
11/4/2025, 8:54:37 PM PST	korrasunilkumar04509@agentforce.com		Deleted Lightning Page_SkillMatch UtilityBar	Lightning Pages	
11/4/2025, 8:54:37 PM PST	korrasunilkumar04509@agentforce.com		Created Lightning Page_SkillMatch UtilityBar	Lightning Pages	
11/4/2025, 8:54:09 PM PST	korrasunilkumar04509@agentforce.com		Created custom app SkillMatch	Custom Apps	
11/4/2025, 8:54:09 PM PST	korrasunilkumar04509@agentforce.com		New Youths validation rule "Email_Required"	Validation Rules	
11/4/2025, 3:49:39 AM PST	korrasunilkumar04509@agentforce.com		Activated flow with Name "Interview Completed" and Unique Name "Interview_Completed"	Flows	
11/4/2025, 3:40:30 AM PST	korrasunilkumar04509@agentforce.com		Created flow with Name "Interview Completed" and Unique Name "Interview_Completed"	Flows	
11/4/2025, 3:40:16 AM PST	korrasunilkumar04509@agentforce.com		Activated flow with Name "Interview Scheduled Flow" and Unique Name "Interview_Scheduled_Flow"	Flows	
11/4/2025, 3:37:47 AM PST	korrasunilkumar04509@agentforce.com		Deleted flow version #1 "Interview Scheduled Flow" for flow with Unique Name "Interview_Scheduled_Flow"	Flows	
11/4/2025, 3:37:43 AM PST	korrasunilkumar04509@agentforce.com		Created flow version #1 "Interview Scheduled Flow" for flow with Unique Name "Interview_Scheduled_Flow"	Flows	
11/4/2025, 3:37:42 AM PST	korrasunilkumar04509@agentforce.com		Created flow with Name "Interview Scheduled Flow" and Unique Name "Interview_Scheduled_Flow"	Flows	
11/4/2025, 3:30:52 AM PST	korrasunilkumar04509@agentforce.com		Activated flow with Name "Welcome Email Template" and Unique Name "Welcome_Email_Template"	Flows	
11/4/2025, 3:30:48 AM PST	korrasunilkumar04509@agentforce.com		Deleted flow version #1 "Welcome Email Template" for flow with Unique Name "Welcome_Email_Template"	Flows	
11/4/2025, 3:30:43 AM PST	korrasunilkumar04509@agentforce.com		Created flow version #1 "Welcome Email Template" for flow with Unique Name "Welcome_Email_Template"	Flows	
11/4/2025, 3:24:39 AM PST	korrasunilkumar04509@agentforce.com		Deactivated flow with Name "Welcome Email Template" and Unique Name "Welcome_Email_Template"	Flows	
11/4/2025, 3:24:39 AM PST	korrasunilkumar04509@agentforce.com		Deleted flow version #1 "Welcome Email Template" for flow with Unique Name "Welcome_Email_Template"	Flows	

[Download setup audit trail for last six months \(Excel csv file\)](#)

develop.my.salesforce-setup.com/lightning/setup/SecurityEvents/home

Phase 10: Quality Assurance Testing

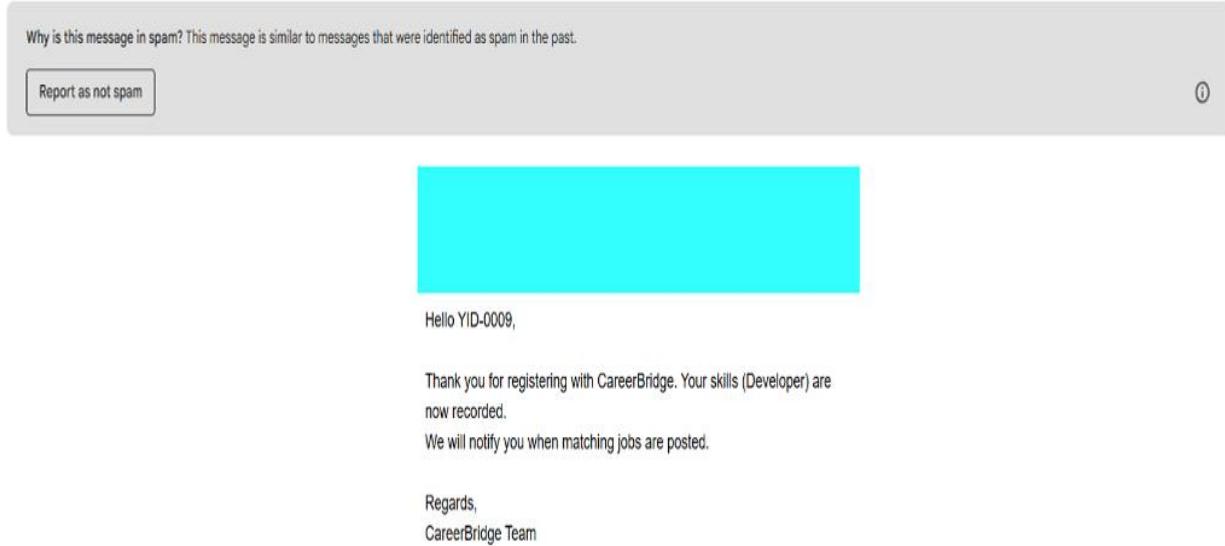
Test Case 1 – Youth Registration → Welcome Email

- **Use Case / Scenario:** Youth registers on the platform.
- **Test Steps (Input):**

- Fill Youth form → Name = Vishnavi, Email = vishnavi@gmail.com, Skills = Java, Python.

- **Expected Result:**

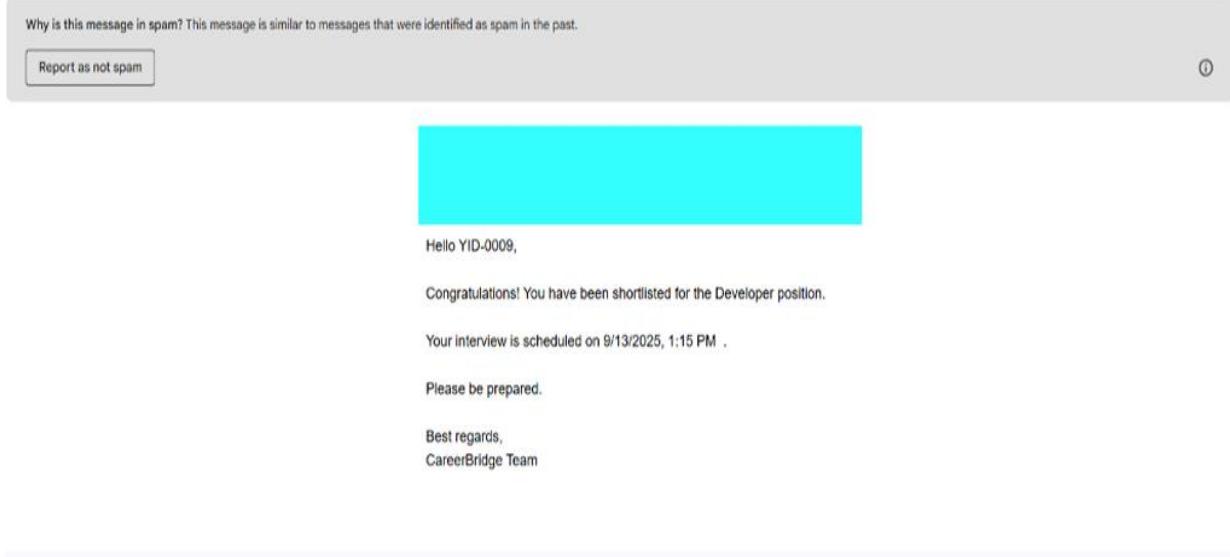
- Youth record saved in **Youth_c** object.
- **Welcome Email** triggered:
“Hello , thank you for registering. Your skills (Python) are now recorded.”



Test Case 2 – Employer Posts Job → Skills Match → Interview Scheduled Email

- **Use Case / Scenario:** Employer posts job requiring skills.
- **Test Steps (Input):**

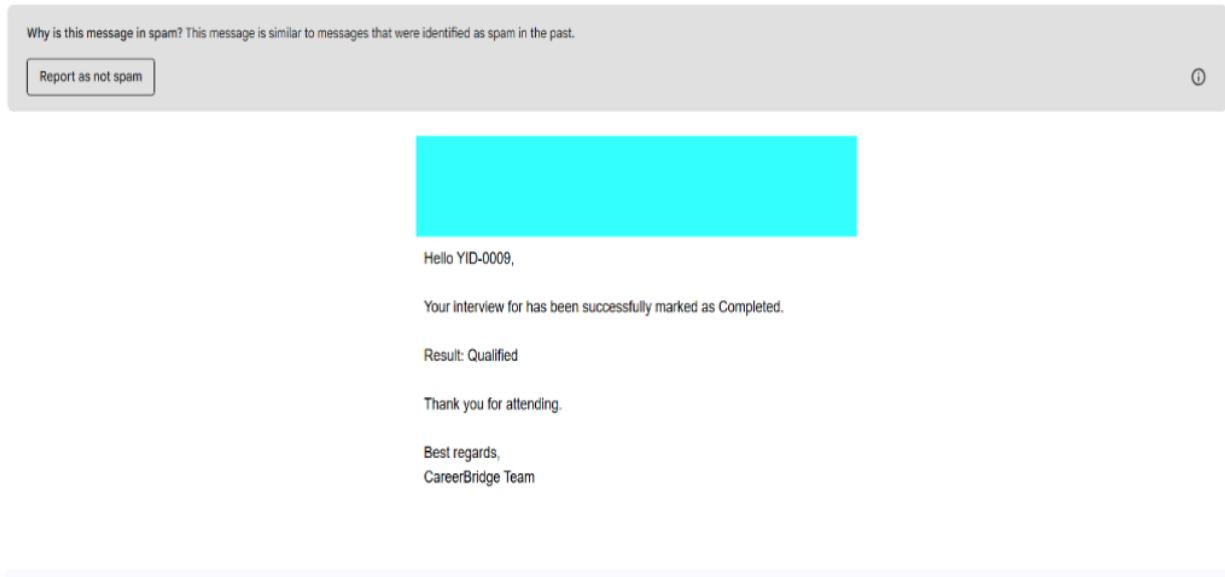
- Job Title = Java Developer, Location = Bangalore, Required Skill = Java.
- **Expected Result:**
 - Job record created in **Job__c** object.
 - System checks youth skills → Match found with Vishnavi.
 - **Interview__c record** auto-created with Status = Scheduled.
 - **Interview Scheduled Email** triggered:
“Hello Vishnavi, congratulations! You are shortlisted for Java Developer.
Interview is scheduled on [Date & Time].”



Test Case 3 – Youth Attends Interview → Interview Completed Email

- **Use Case / Scenario:** Interview is completed.

- **Test Steps (Input):**
 - Coordinator updates Interview record → Change Status__c = Completed.
- **Expected Result:**
 - Status changes to Completed.
 - **Interview Completed Email** triggered:
“Hello Vishnavi, your interview for Java Developer has been marked as Completed. Thank you for attending.”



Test Case 4 – Validation Rules (Error Handling)

- **Use Case / Scenario:** Youth tries to register without Email or Skills.
- **Test Steps (Input):**

- Leave Email blank → Save.
 - Leave Location blank → Save.
- **Expected Result:**
 - System displays validation error messages:
 - ISBLANK>Email__c → “Email is required.”
 - ISBLANK>Location__c → “Location must be entered.”

Youth ID: YID-0009

Owner: vishnavi siddavatam

Email:

Location: Hyderabad

Qualification: B.tech

Preferred Role: Developer

Created By:

We hit a snag. [Review the errors on this page.](#)

Conclusion

The CareerBridge CRM project was successfully implemented and tested across all phases.

In Phase 10, Quality Assurance Testing validated that every Salesforce feature—record creation, flows, triggers, automatic emails, and validation rules—worked as expected.

The end-to-end flow ensures:

- Youth registration captures complete details and sends a **Welcome Email**.
- Employers post jobs, and the system automatically matches required skills with youth profiles.
- Matching results trigger interview scheduling along with an **Interview Scheduled Email**.
- Once interviews are completed, coordinators update the status, and the system sends an **Interview Completed Email**.
- Validation rules prevent saving incomplete or incorrect data, ensuring accuracy.

This testing confirms that the platform is:

- **Reliable** – All workflows run without errors.
- **Automated** – Reduces manual work for NGOs and coordinators.
- **User-friendly** – Provides transparency to youth, NGOs, and employers.

Final Outcome: CareerBridge CRM successfully connects youth skills to jobs, supports NGOs in managing opportunities, and helps employers find the right candidates efficiently