

Project Title: CareerBridge CRM – Connecting Youth Skills to Jobs

Project Overview:

CareerBridge CRM is a Salesforce-based solution designed to bridge the gap between youth talent, NGOs, and employers. The system allows youth to register their skills, employers to post jobs, and automatically matches candidates to jobs based on their skills. It manages the entire interview lifecycle, sends email notifications, and provides dashboards and reports for NGOs to track outcomes. The solution ensures efficiency, transparency, and scalability in youth employment initiatives.

Objectives

- Enable youth to register with essential details and skills.
- Allow employers to post jobs with skill requirements.
- Automate skill-based job matching and interview scheduling.
- Provide real-time email notifications to youth.
- Generate reports and dashboards for NGO monitoring.
- Ensure secure, scalable, and maintainable Salesforce CRM.

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, NGOs, 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 for NGOs.

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.
-

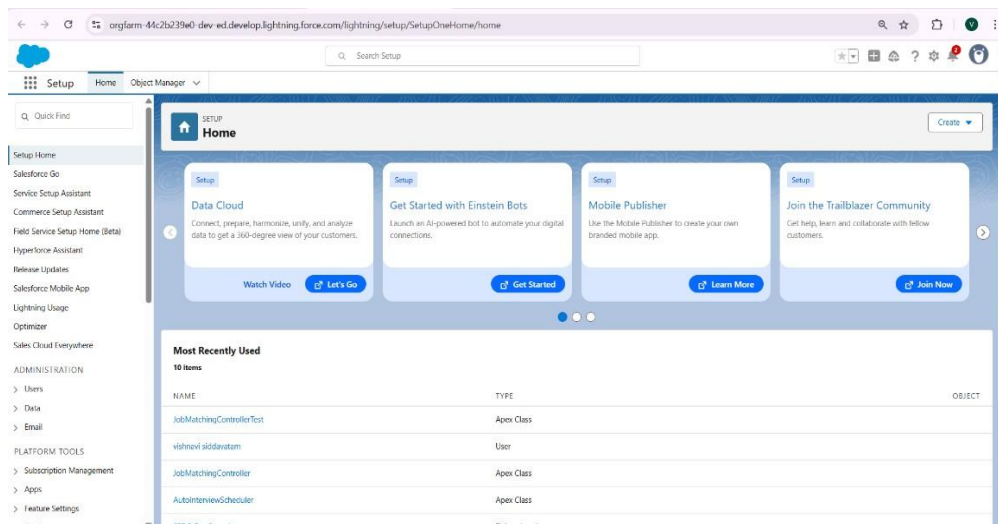
AppExchange Exploration

- 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

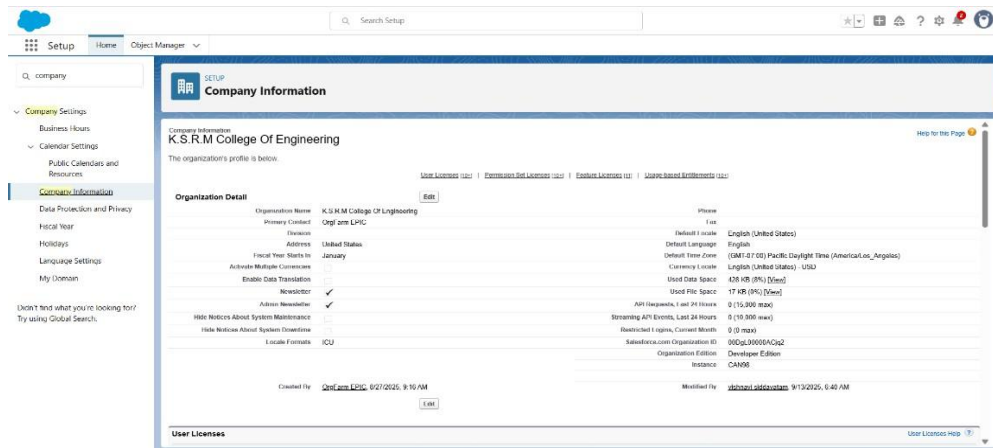
Step 1: Open Setup

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**



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.

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).

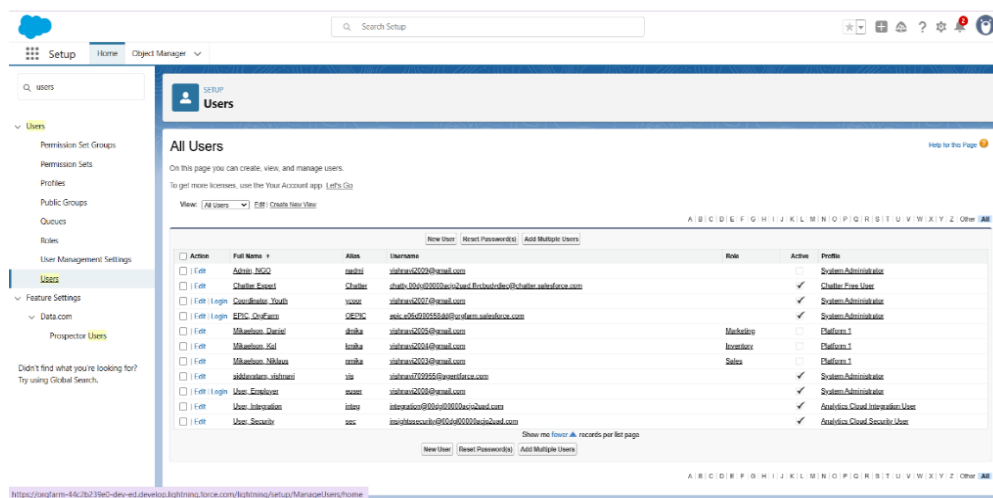
Step 5: Create Test Users

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

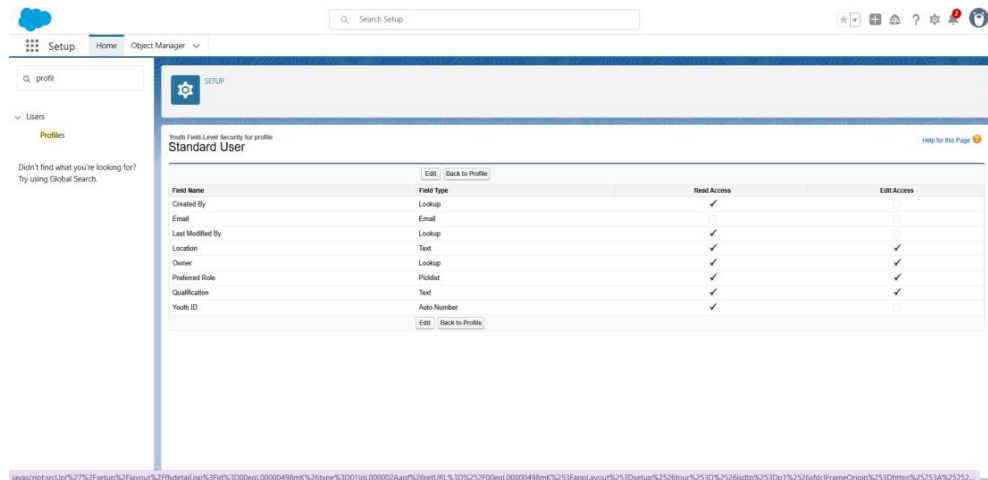
3. Click **Save** after each user.



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



Step 7: Create Role Hierarchy

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

2. Example hierarchy:

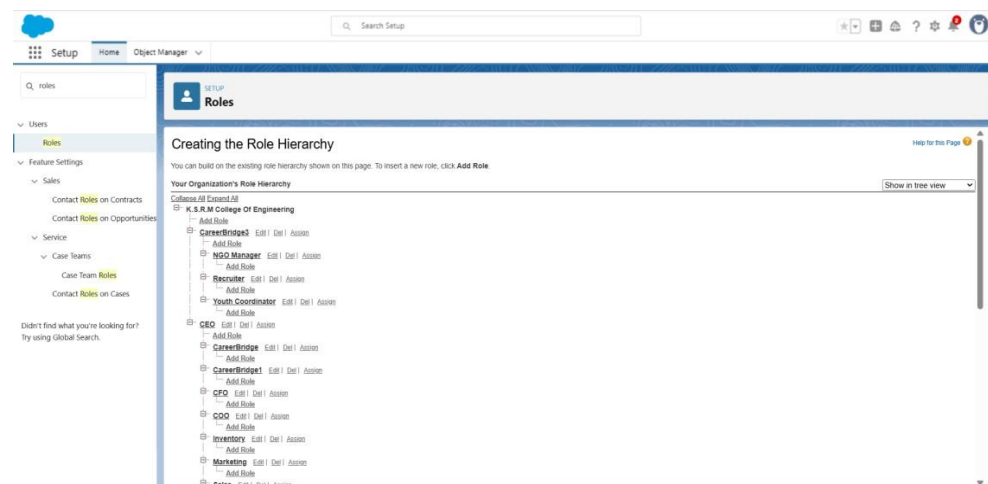
CareerBridge

├── NGO Manager

├── Youth Coordinator

└── Recruiter

3. Click **Save**.



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:

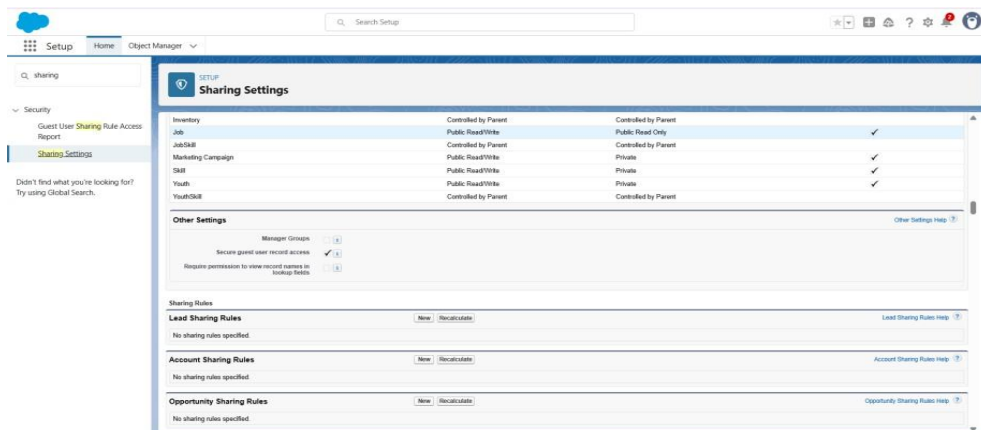
Object	Default Access
--------	----------------

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:

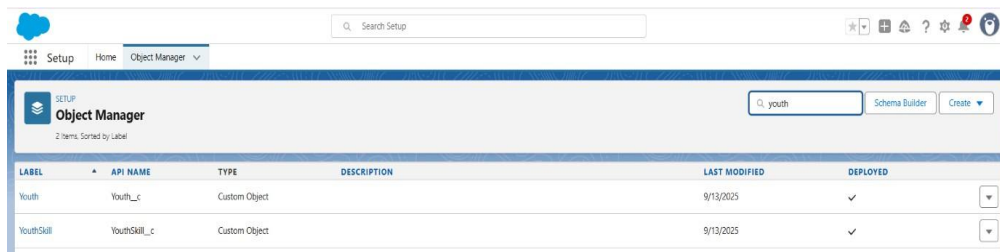
Object Name	API Name	Purpose
Youth	Youth__c	Tracks youth registration info
Job	Job__c	Tracks job postings

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. The search bar contains 'youth'. The table lists two objects: Youth and YouthSkill, both of type Custom Object, last modified on 9/13/2025, and deployed.

LABEL	API NAME	TYPE	DESCRIPTION	LAST MODIFIED	DEPLOYED
Youth	Youth__c	Custom Object		9/13/2025	✓
YouthSkill	YouthSkill__c	Custom Object		9/13/2025	✓



The screenshot shows the Salesforce Object Manager interface. The search bar contains 'job'. The table lists two objects: Job and JobSkill, both of type Custom Object, last modified on 9/13/2025, and deployed.

LABEL	API NAME	TYPE	DESCRIPTION	LAST MODIFIED	DEPLOYED
Job	Job__c	Custom Object		9/13/2025	✓
JobSkill	JobSkill__c	Custom Object		9/13/2025	✓



The screenshot shows the Salesforce Object Manager interface. The search bar contains 'job'. The table lists two objects: Job and JobSkill, both of type Custom Object, last modified on 9/13/2025, and deployed.

LABEL	API NAME	TYPE	DESCRIPTION	LAST MODIFIED	DEPLOYED
Job	Job__c	Custom Object		9/13/2025	✓
JobSkill	JobSkill__c	Custom Object		9/13/2025	✓

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

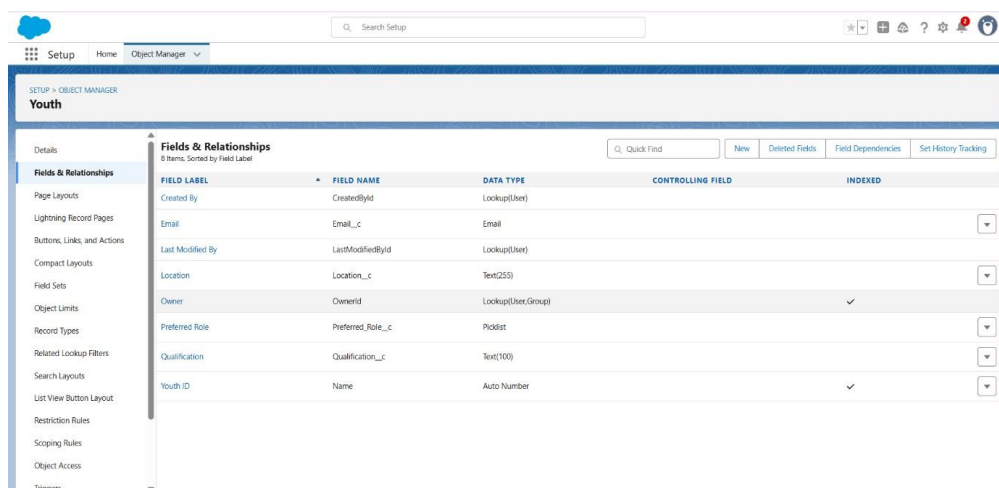
Preferred_Role__c Picklist Options: Developer, Designer, Sales

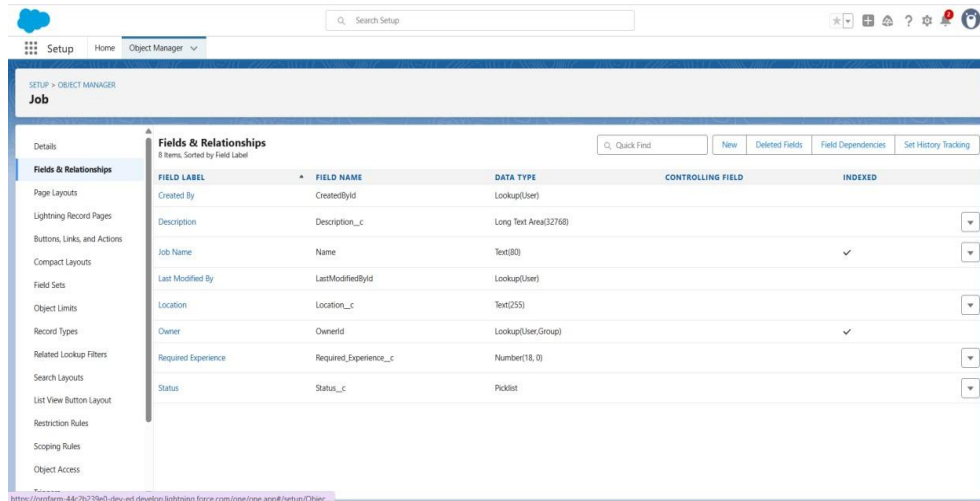
Job__c Fields

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





SETUP > OBJECT MANAGER

Job

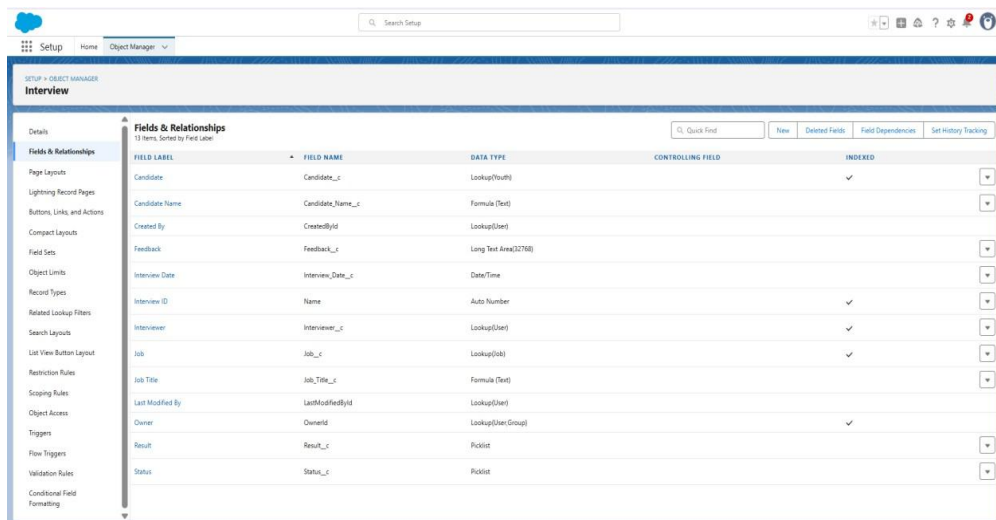
Details

Fields & Relationships
8 Items, Sorted by Field Label

Quick Find: [] New Deleted Fields Field Dependencies Set History Tracking

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedById	Lookup(User)		
Description	Description__c	Long Text Area(32768)		
Job Name	Name	Text(80)		✓
Last Modified By	LastModifiedById	Lookup(User)		
Location	Location__c	Text(255)		
Owner	OwnerId	Lookup(User, Group)		✓
Required Experience	Required_Experience__c	Number(18, 0)		
Status	Status__c	Picklist		

<https://corfarm-44c2b239a0-dev-ed.develop.lightning.force.com/one/one.apex#/setup/ObjectManager/Job>



SETUP > OBJECT MANAGER

Interview

Details

Fields & Relationships
13 Items, Sorted by Field Label

Quick Find: [] New Deleted Fields Field Dependencies Set History Tracking

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Candidate	Candidate__c	Lookup(Youth)		✓
Candidate Name	Candidate_Name__c	Formula (Text)		
Created By	CreatedById	Lookup(User)		
Feedback	Feedback__c	Long Text Area(32768)		
Interview Date	Interview_Date__c	Date/Time		
Interview ID	Name	Auto Number		✓
Interviewer	Interviewer__c	Lookup(User)		✓
Job	Job__c	Lookup(Job)		✓
Job Title	Job_Title__c	Formula (Text)		
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User, Group)		✓
Result	Result__c	Picklist		
Status	Status__c	Picklist		

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

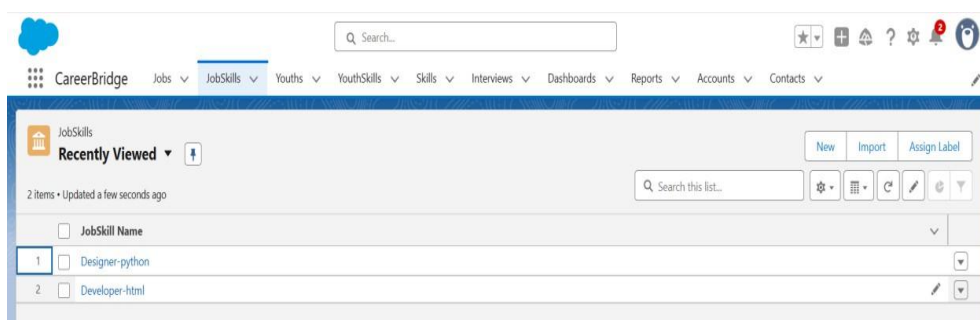
- Fields: Candidate, Job, Interview Date, Interviewer, Status, Feedback

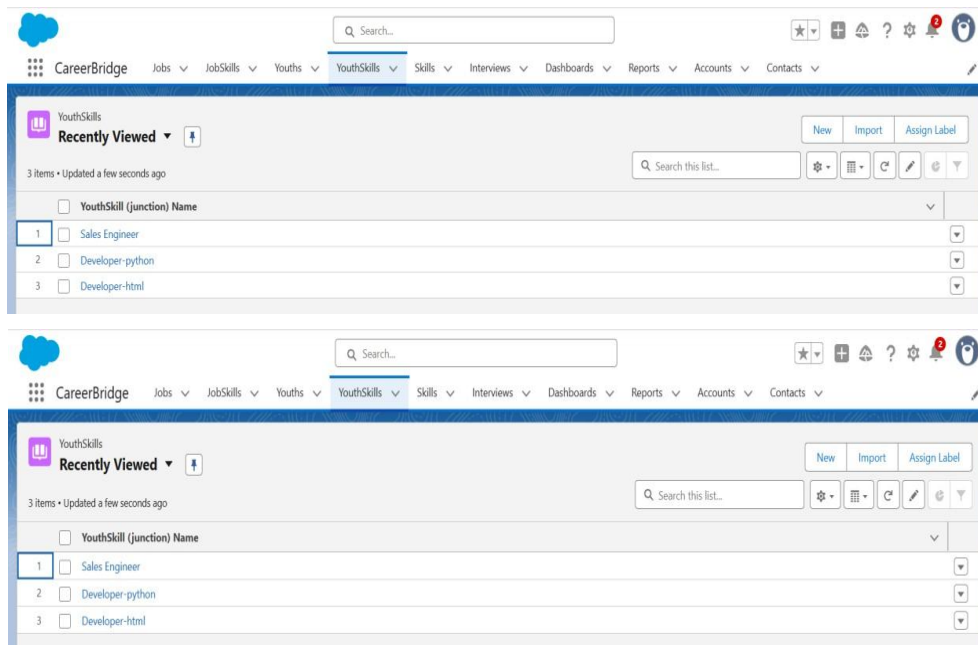
Step 5 : Add Sample Records

- **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)



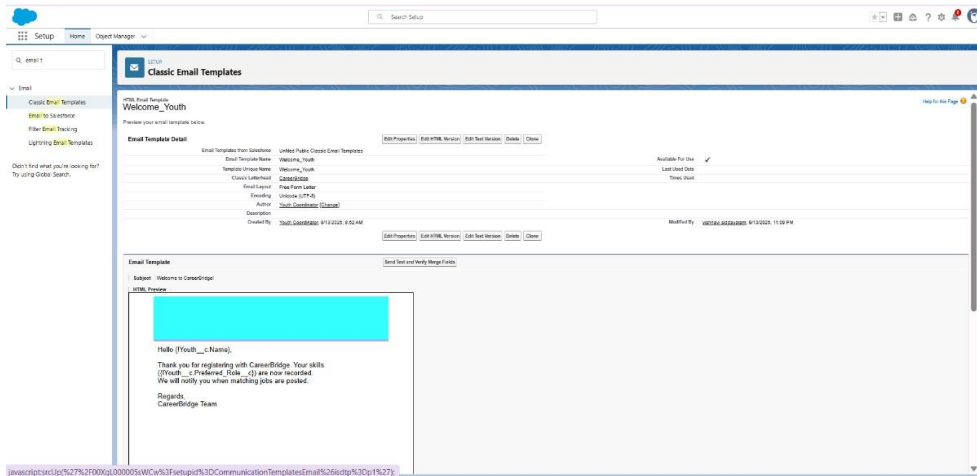


Phase 4: Process Automation (Admin)

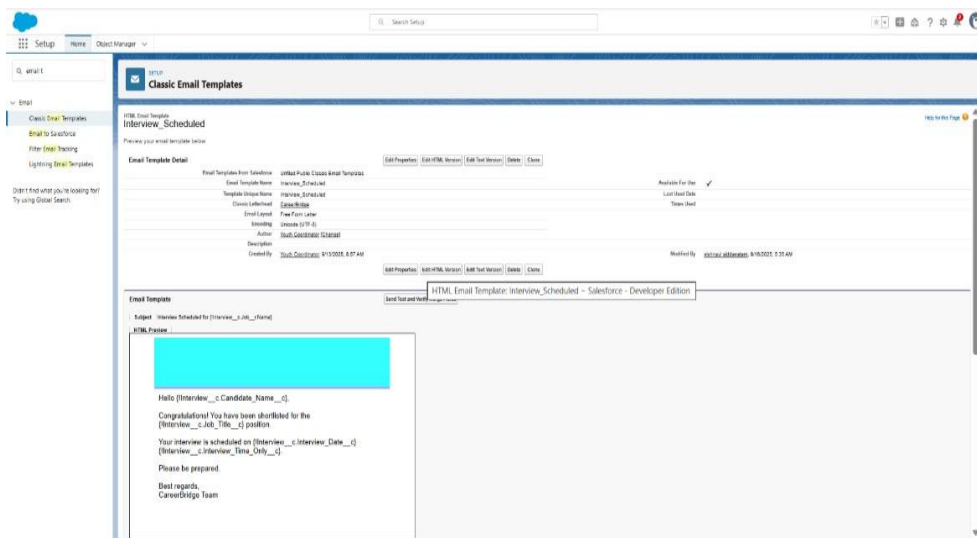
Goal: Automate youth–job matching, interview scheduling, and email notifications, while ensuring data integrity.

Step 1: Create Email Templates

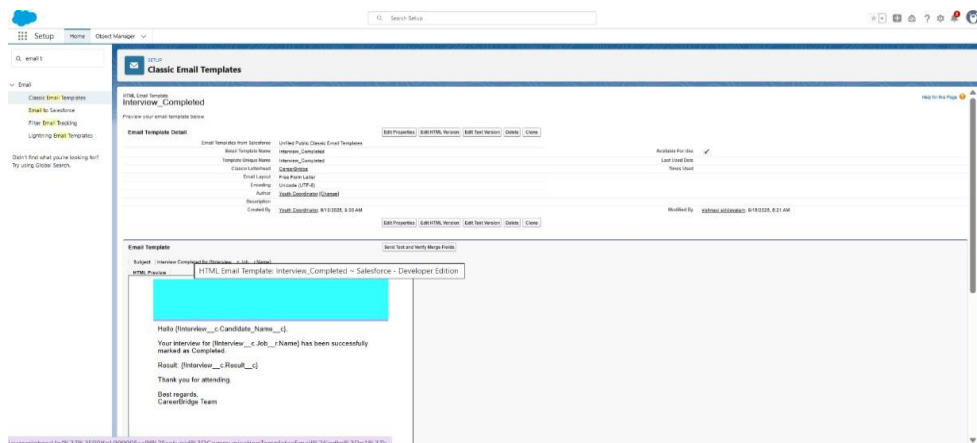
1. Click **Setup** (⚙️) → In Quick Find, type **Email Templates** → Select **Email Templates**.
 2. Click **New Email Template** → Select **Lightning Email Template**.
 3. Fill details like **Name** and **Subject**, then design the email body.
 4. Save the template.
- **Welcome Email** → Sent after youth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



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



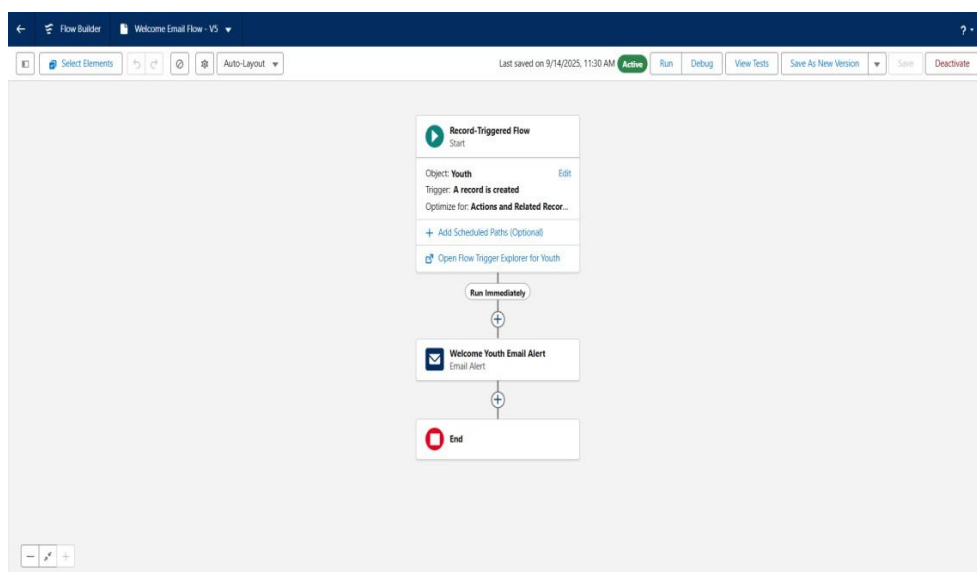
- **Interview Completed Email** → Sent after interview status is marked as Completed.



Step 2: Build Flows

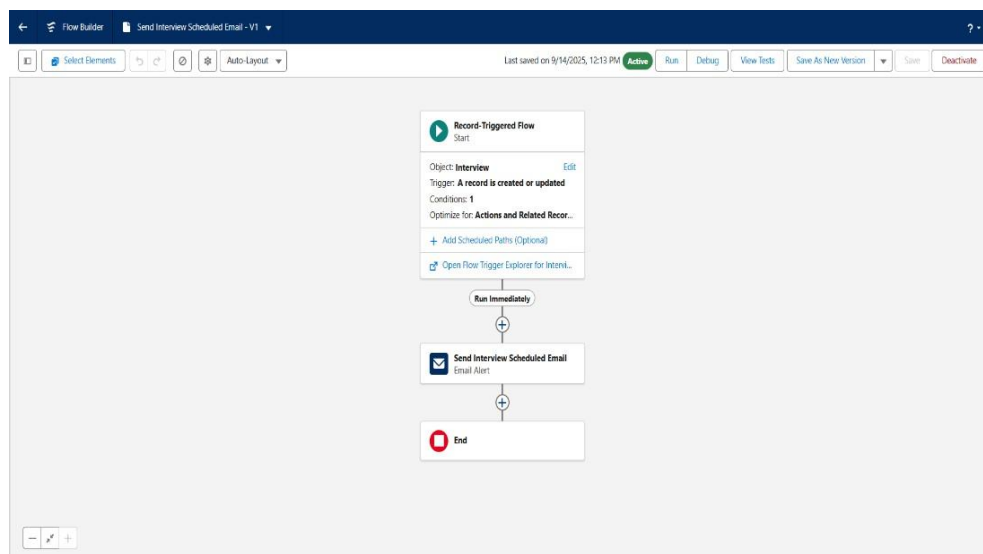
1. Welcome Email Flow (After Registration)

1. Click **Setup** (⚙️) → 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

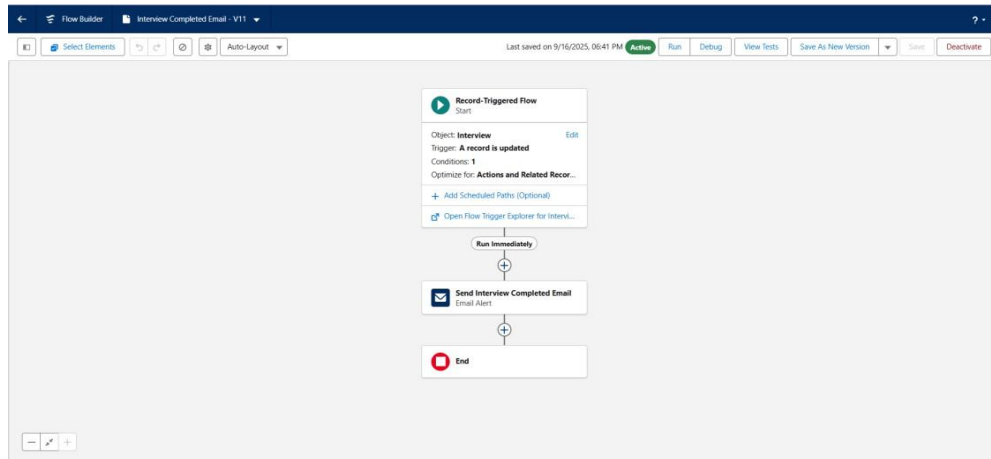


3. Interview Completed Flow (After Interview Completion)

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.”

The screenshot shows the 'Youth Validation Rule' configuration page in the Salesforce Object Manager. The page is titled 'Youth Validation Rule' and includes a 'Back to Youth' link. The 'Validation Rule Detail' section shows the rule name 'Email_Required', the formula 'ISBLANK(Email__c)', and the error message 'Email is required.'. The rule is currently active, as indicated by the 'Active' checkbox. The 'Error Location' is set to 'Top of Page'. The 'Description' is 'Prevents saving a Youth record without an Email'. The 'Created By' is 'Youth Coordinate' and the 'Modified By' is 'Youth Coordinate'. The 'Created' date is '9/13/2025, 9:11 AM' and the 'Modified' date is '9/13/2025, 9:11 AM'. The page includes a left sidebar with navigation links for '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', and 'Triggers'.

CareerBridge

Search...

Jobs JobSkills Youths YouthSkills Skills Interviews Dashboards Reports Accounts Contacts

Youth YID-0009

New Contact Edit New Opportunity

Related Details

* Required Information

Youth ID: YID-0009

Owner: vishnavi siddavatam

Email: [Empty Field]

Location: Hyderabad

Qualification: Btech

Preferred Role: Developer

Created By: vishnavi siddavatam, 9/13/2025, 11:10 PM

By: siddavatam, 9/14/2025, 12:08 AM

We hit a snag. Review the errors on this page. Email is required.

Cancel Save

3.Create Rule: **Location Required**

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

Setup Home Object Manager

Search Setup

SETUP > OBJECT MANAGER

Youth

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

Youth Validation Rule

Back to Youth

Validation Rule Detail

Rule Name: Location_Required

Error Condition Formula: ISBLANK(Location__c)

Error Message: Location is required

Description: Youth Validation Rule - Salesforce - Developer Edition

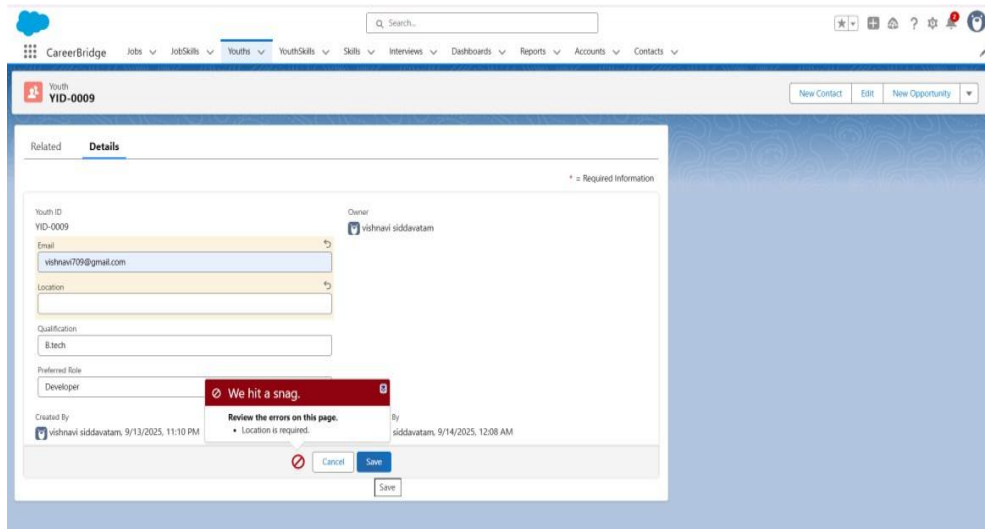
Created By: Youth Coordinator

Modified By: Youth Coordinator

Active: [Checked]

Error Location: Top of Page

Modified: 9/13/2025 9:15 AM



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**

```

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>();
20     }
21 }

```

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).

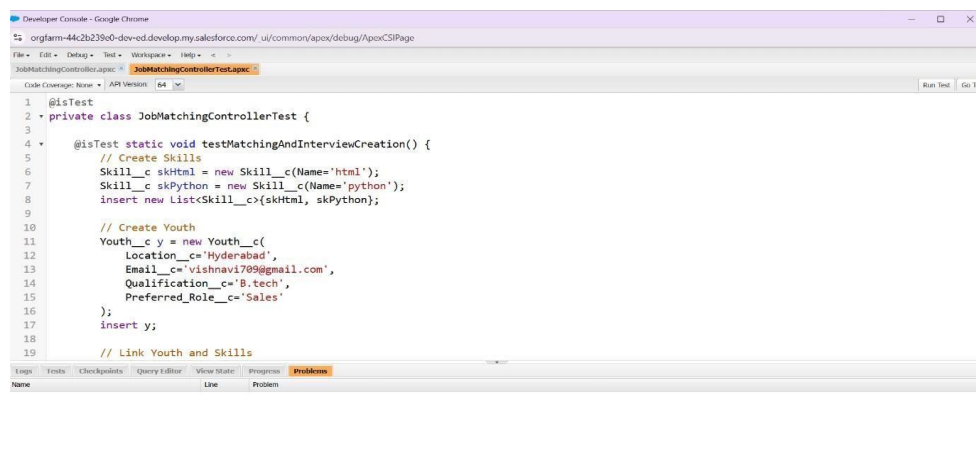


Step 2: Apex Test Class — JobMatchingControllerTest

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

Steps to create Test Class:

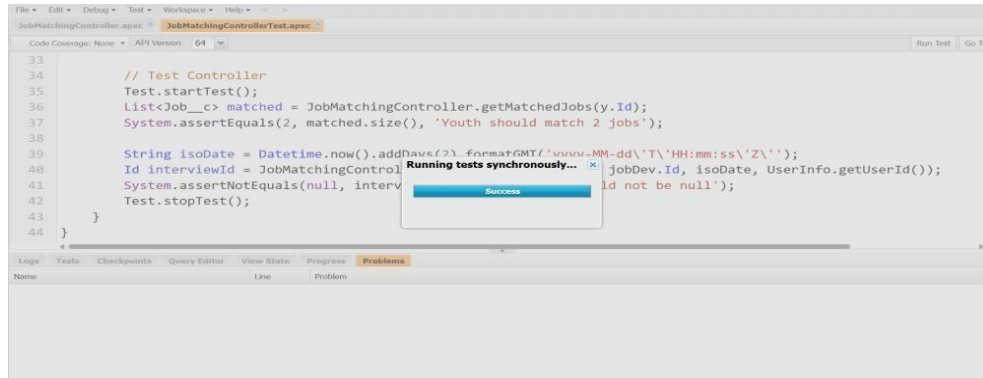
1. Click **Setup** → **Apex Classes** → **New**.
2. Copy and paste the JobMatchingControllerTest code below.
3. Click **Save**.



Test Logic Overview:

- Creates test Skill__c, Youth__c, YouthSkill__c, Job__c, JobSkill__c.
- Tests getMatchedJobs() → asserts correct number of jobs.

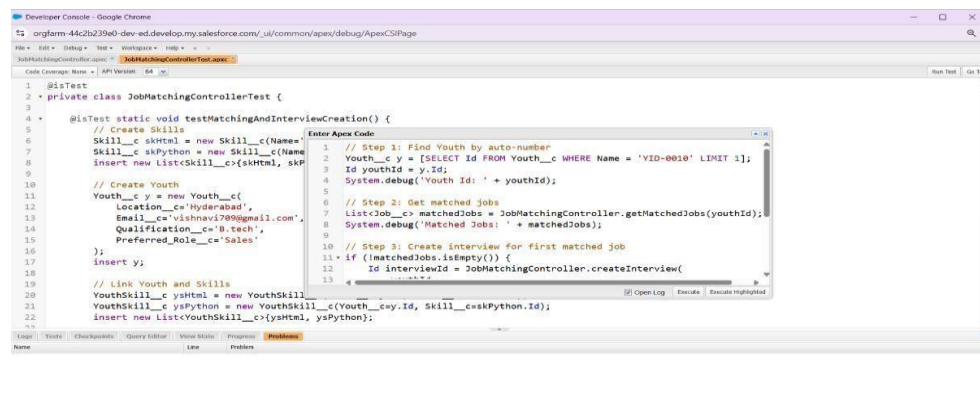
- Tests createInterview() → asserts interview Id is not null.



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):



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.

Percentage of Scheduled Jobs Used: 1%
You have currently used 1 scheduled Apex jobs out of an allowed organization limit of 100 active or scheduled jobs. To learn about how this limit is calculated and what contributes to it see the [Lightning Platform Apex Limits](#) topic.

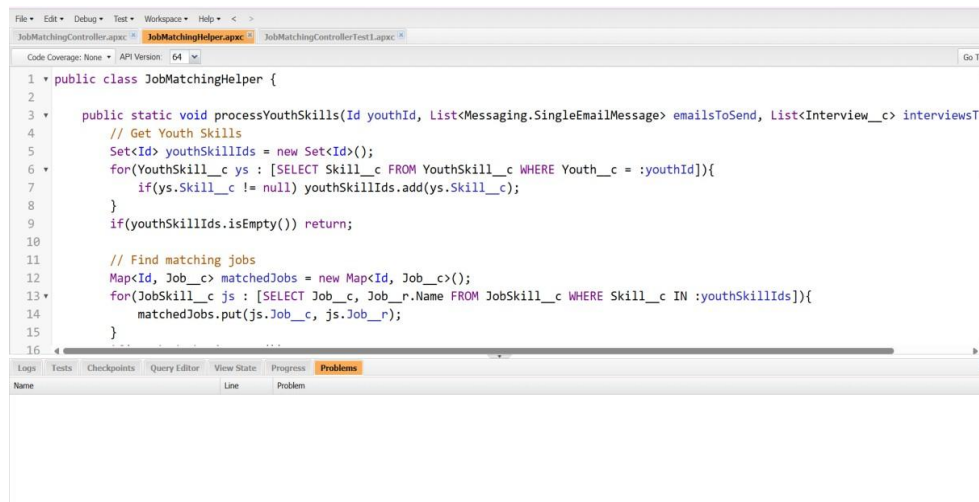
Action	Job Name	Submitted By	Submitted	Started	Next Scheduled Run	Type	Cron Trigger ID
Manage Del Pause Job	AutoInterviewScheduler Job	jsdistrations_@stbzad	9/19/2025, 7:48 PM		9/20/2025, 10:00 AM	Scheduled Apex	08egL00000C8McTV
Del	Metastatics Data Loader Job for Org: (00DgLO0000A0A0Qp)	User: jldistrations	8/27/2025, 9:18 AM	9/19/2025, 4:53 AM	9/20/2025, 4:53 AM	Autonomous Data Loader Job	08egL00000A0c8LX
	Program Milestone Computation Cron Job	Process, Automated	8/27/2025, 9:18 AM	9/19/2025, 5:00 PM	9/19/2025, 11:59 PM	Program Milestone Computation Cron Job	08egL00000A0c8WV
	Program Status Update Cron Job	Process, Automated	8/27/2025, 9:18 AM	9/19/2025, 5:01 AM	9/19/2025, 8:00 PM	Program Status Update Cron Job	08egL00000A0c8WV

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**.

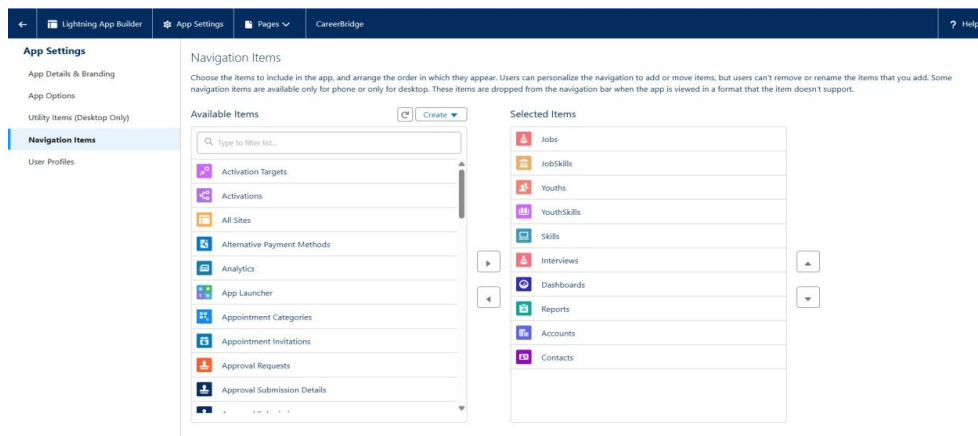
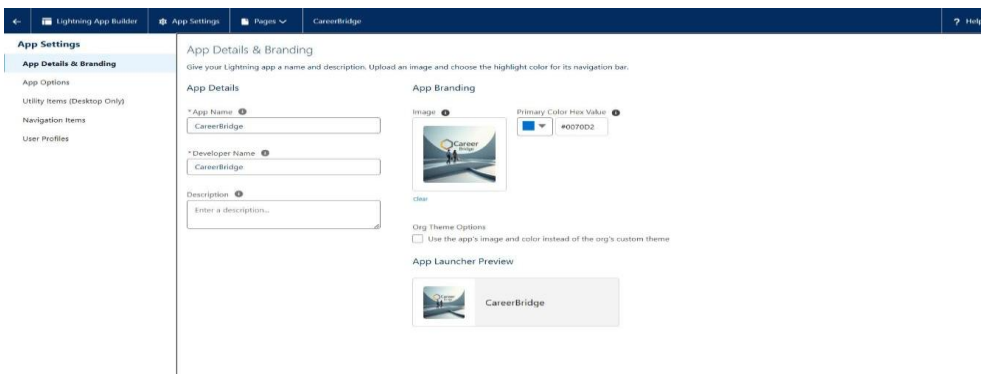


```
1 public class JobMatchingHelper {
2
3     public static void processYouthSkills(Id youthId, List<Messaging.SingleEmailMessage> emailsToSend, List<Interview__c> interviewsToCreate) {
4         // Get Youth Skills
5         Set<Id> youthSkillIds = new Set<Id>();
6         for(YouthSkill__c ys : [SELECT 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
11         // Find matching jobs
12         Map<Id, Job__c> matchedJobs = new Map<Id, Job__c>();
13         for(JobSkill__c js : [SELECT Job__c, Job__r.Name FROM JobSkill__c WHERE Skill__c IN :youthSkillIds]){
14             matchedJobs.put(js.Job__c, js.Job__r);
15         }
16     }
17 }
```

Phase 6: User Interface Development

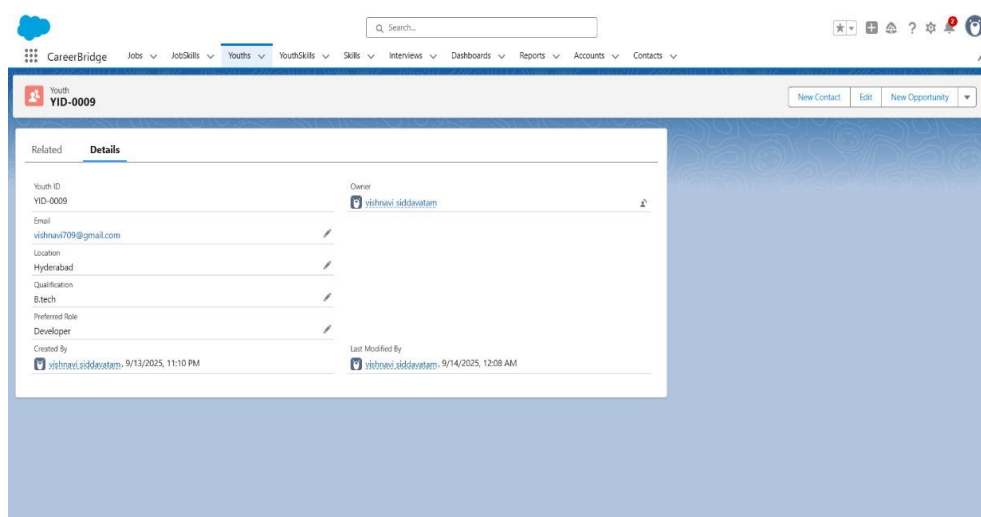
Step1:Lightning App Builder

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

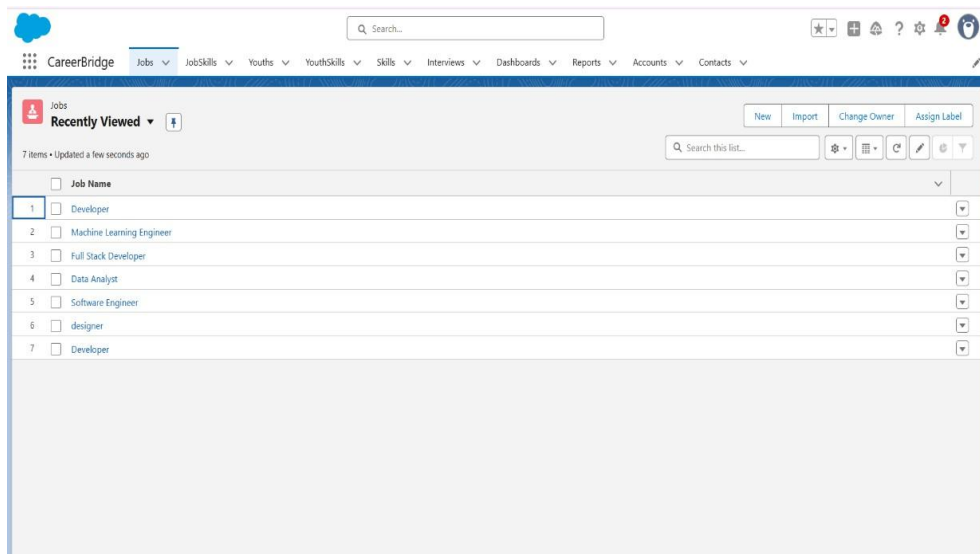


Step 2: Record Pages

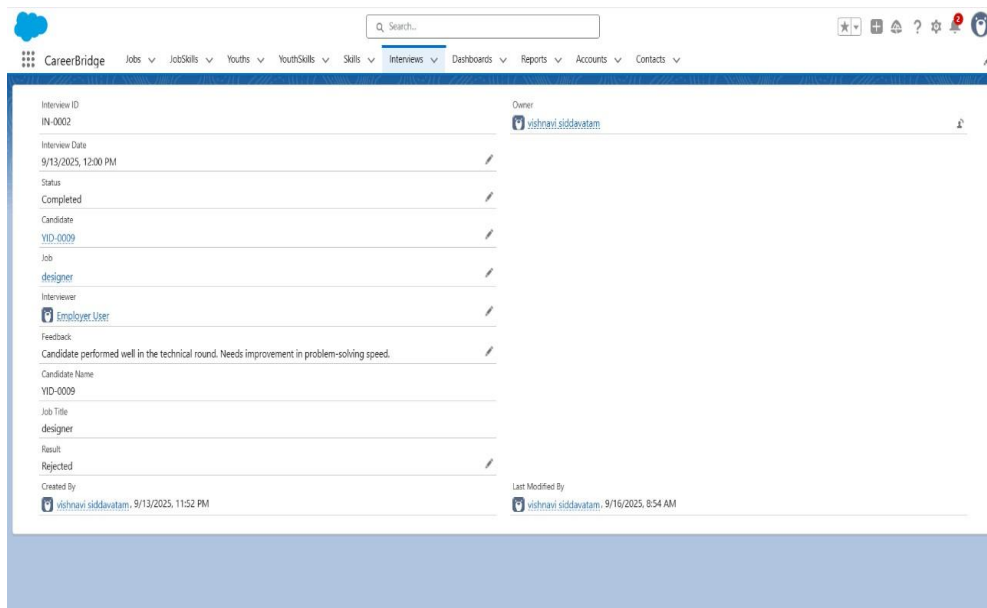
- Custom layouts for specific object records (**Youth, Job, Interview, Skill**).
- You can decide what **fields, related lists, and components** appear.



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



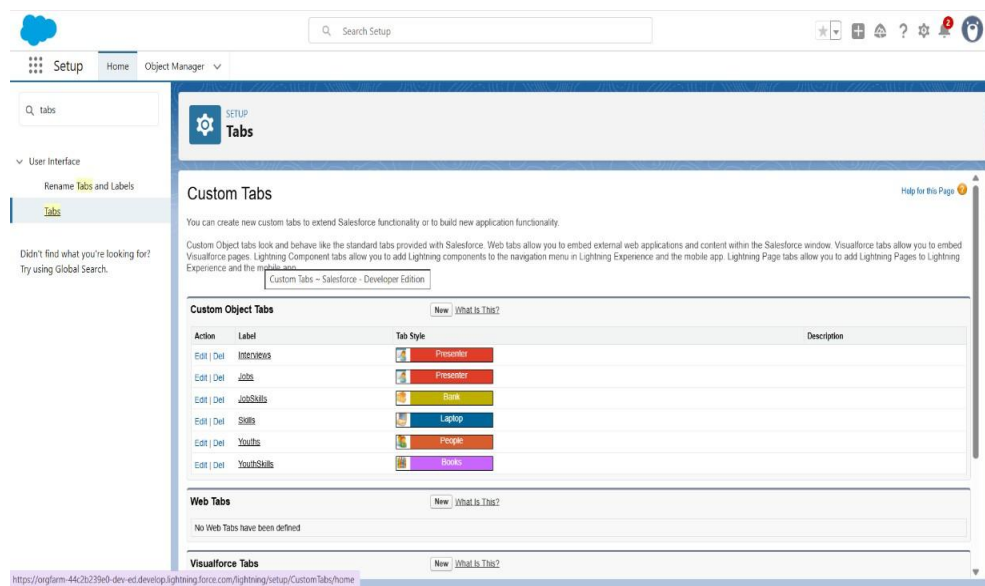
Job Record Page: Show job details, required skills, and linked inter views



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

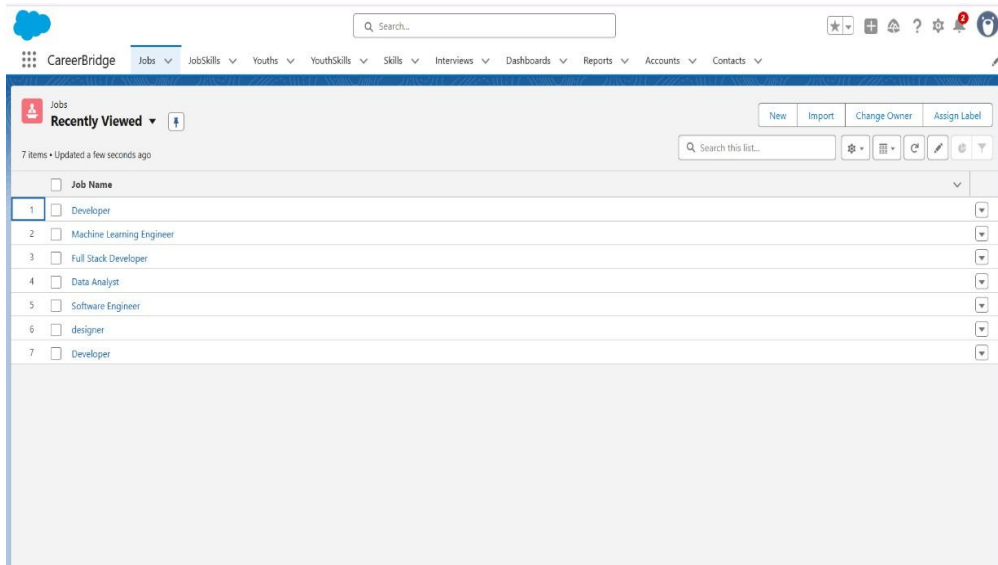
Step 3: Tabs

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



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.



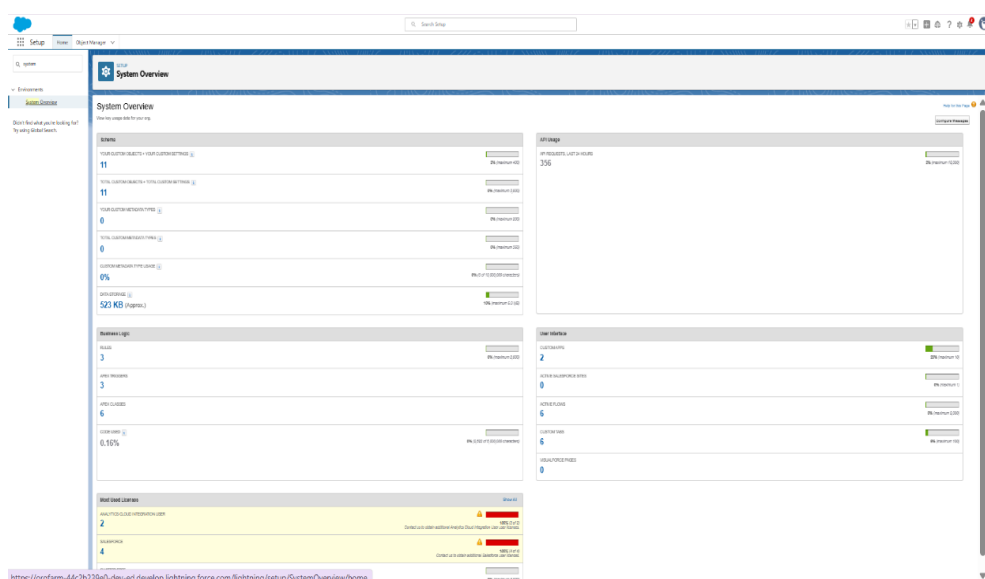
Phase 7: Integration & External Access

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.



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**.

The screenshot shows the Salesforce Setup interface for creating a new Matching Rule. The page is titled 'Matching Rules' and the breadcrumb trail is 'Setup > Matching Rules'. The main content area is titled 'Edit Rule Youth Email Match'. It contains a 'Rule Details' section with fields for 'Object' (Youth), 'Rule Name' (Youth Email Match), 'Unique Name' (Youth_Email_Match), and 'Description'. Below this is the 'Matching Criteria' section, which has a table with columns 'Field', 'Matching Method', and 'Match Blank Fields'. The table has four rows, all with 'Email' in the 'Field' column and 'Exact' in the 'Matching Method' column. The 'Match Blank Fields' column has checkboxes, with the first one checked. To the right of the table are 'AND' and 'OR' buttons. At the bottom of the page are 'Save' and 'Cancel' buttons.

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.

Autosave OFF

Job_c.csv - Read Only - Saved to this PC

Search

FileHomeInsertDrawPage LayoutFormulasDataReviewViewHelp

Calibri11A A

B I U

Font

Wrap Text

Alignment

Number

Styles

Conditional Formatting

Format as Table

Cell Styles

Insert

Delete

Format

Cells

AutoSum

Fill

Clear

Editing

Comments

Sort & Filter

Find & Select

Add-ins

Share

POSSIBLE DATA LOSS

Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To preserve these features, save it in an Excel file format.

Don't show again

Save As...

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	Id	OwnerId	IsDeleted	Name	CreatedOn	CreatedBy	LastModified	LastModified	SystemMo	Location	Description	Required	Status	c									
2	a09gl.0000.005gl.0000	0	Java Devel	Developer	005gl.0000.005gl.0000	005gl.0000.005gl.0000	Hyderabad	Developer				2	Open										
3	a09gl.0000.005gl.0000	0	Developer	Developer	005gl.0000.005gl.0000	005gl.0000.005gl.0000	Hyderabad	Required E				2	Open										
4	a09gl.0000.005gl.0000	0	designer	designer	005gl.0000.005gl.0000	005gl.0000.005gl.0000	Hyderabad					1	Open										
5	a09gl.0000.005gl.0000	0	Software I	Software I	005gl.0000.005gl.0000	005gl.0000.005gl.0000	Bangalore					0	Open										
6	a09gl.0000.005gl.0000	0	Data Anal	Data Anal	005gl.0000.005gl.0000	005gl.0000.005gl.0000	Chennai					2	Open										
7	a09gl.0000.005gl.0000	0	Full Stack	Full Stack	005gl.0000.005gl.0000	005gl.0000.005gl.0000	Pune					1	Open										
8	a09gl.0000.005gl.0000	0	Machine L	Machine L	005gl.0000.005gl.0000	005gl.0000.005gl.0000	Delhi					1	Open										

Job_c

1

AutoSave OFF

Interview_c.csv - Read-... - Saved to this PC

Search

FileHomeInsertDrawPage LayoutFormulasDataReviewViewHelp

CutCopyPasteFormat PainterClipboard

Calibri11A⁺A⁻BBIU⁺⁻Font

Wrap TextMerge & CenterAlignmentNumber

GeneralConditional FormattingStylesFormat as TableCell StylesInsertDeleteFormat

AutosumFillClearEditing

CommentsSort & FilterFind & SelectAdd-ins

POSSIBLE DATA LOSS

Some features might be lost if you save this workbook in the comma delimited (.csv) format. To preserve these features, save it in an Excel file format.

Don't show again

Save As...

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	Id	OwnerId	IsDeleted	Name	CreatedOn	CreatedBy	LastModified	SystemMo	Interview	Status	Candidate	Job_c	Interview	Feedback	Result	c							
2	a07gl.0000.005gl.0000	0	IN-0001		005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	Completed	a09gl.0000.005gl.0000	a09gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	
3	a07gl.0000.005gl.0000	0	IN-0002		005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	Completed	a09gl.0000.005gl.0000	a09gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	
4	a07gl.0000.005gl.0000	0	IN-0003		005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	Scheduled	a09gl.0000.005gl.0000	a09gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	
5	a07gl.0000.005gl.0000	0	IN-0004		005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	Scheduled	a09gl.0000.005gl.0000	a09gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	
6	a07gl.0000.005gl.0000	0	IN-0005		005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	Completed	a09gl.0000.005gl.0000	a09gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	
7	a07gl.0000.005gl.0000	0	IN-0005		005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	Completed	a09gl.0000.005gl.0000	a09gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	005gl.0000.005gl.0000	

<>

Interview_c

+

ReadyAccessibility: Unavailable

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)

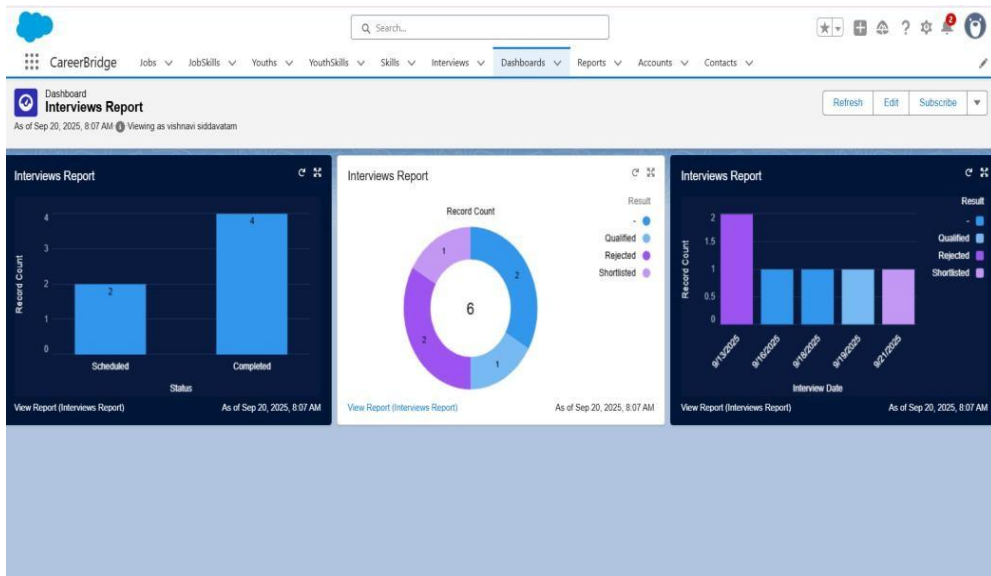
Status	Result	Interview Date	Candidate Name	Job
Scheduled (2)	(2)	9/16/2025 (1)	YID-0009	Developer
	Subtotal			
		9/18/2025 (1)	YID-0001	Java Developer
	Subtotal			
Subtotal				
Completed (4)	Qualified (1)	9/19/2025 (1)	YID-0009	Developer
	Subtotal			
		9/19/2025 (2)	YID-0009	designer
	Projected (2)		YID-0009	Developer
	Subtotal			
		9/21/2025 (1)	YID-0009	designer
	Shortlisted (1)			
	Subtotal			
Subtotal				
Total (6)				

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”

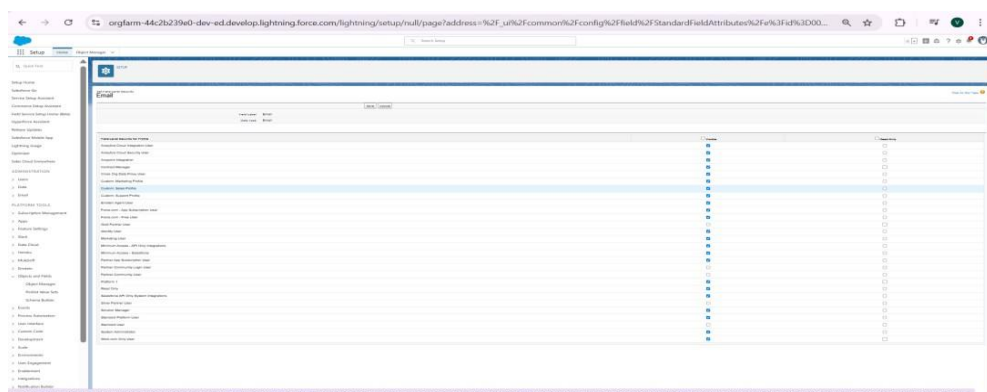


Step 3: Security

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

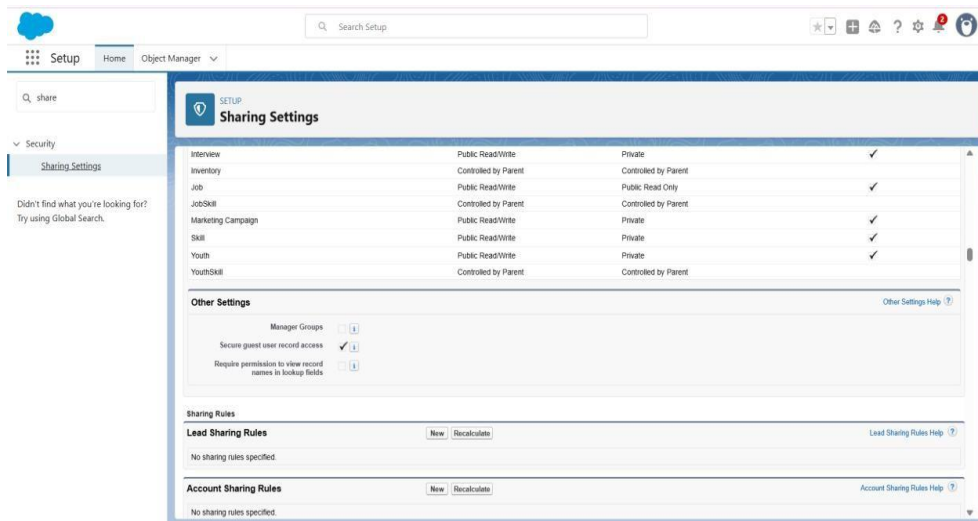
A. Field-Level Security (FLS)

1. Setup → Object Manager → Youth__c → Fields → Email__c → Set Field-Level Security
2. Hide fields from standard users or users without permission



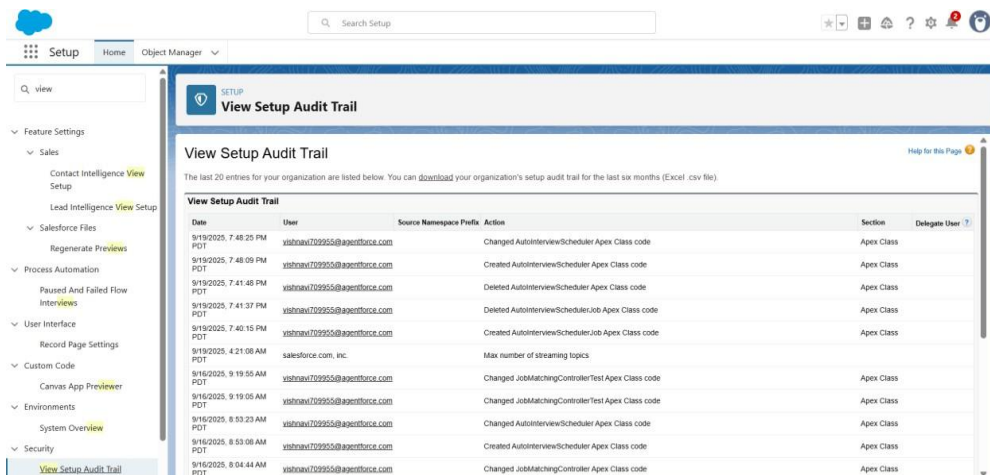
B. Sharing & Org-Wide Defaults (OWD)

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

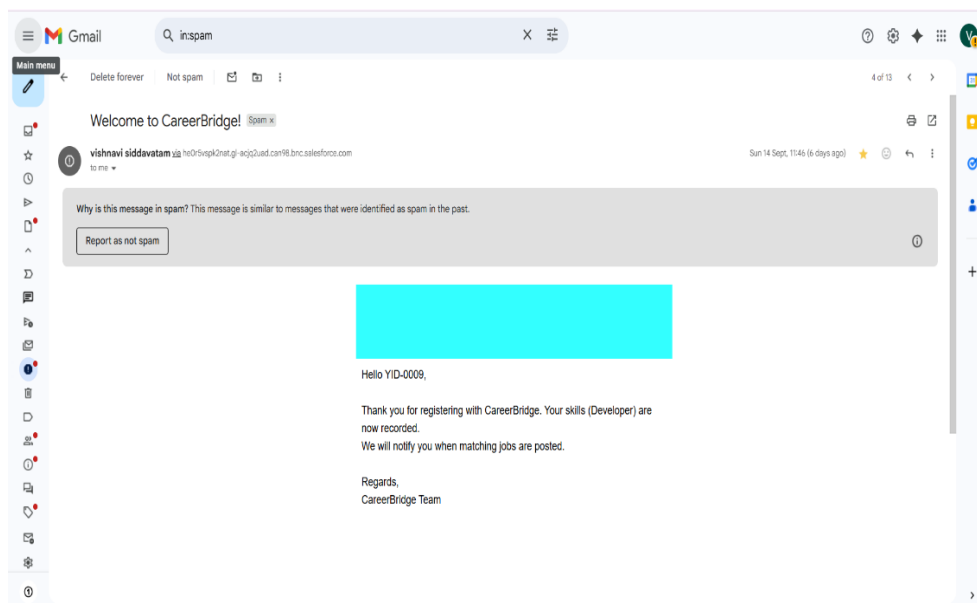
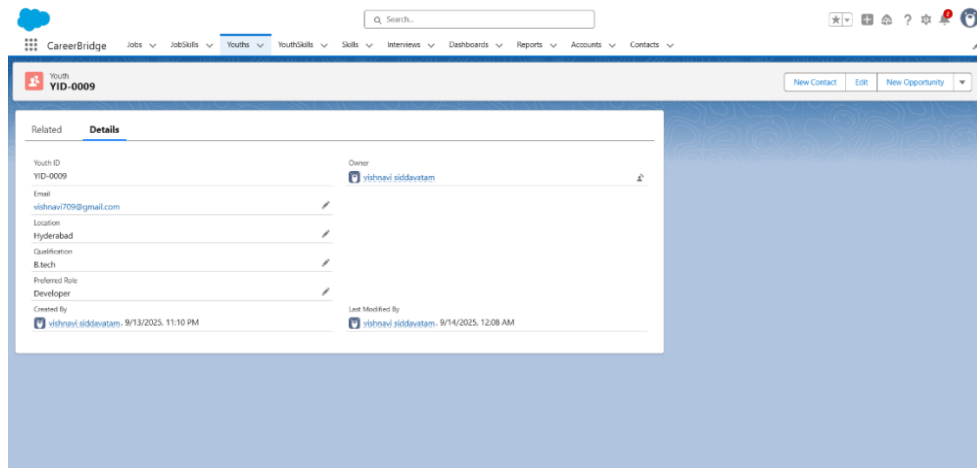


Phase 10: Quality Assurance Testing

Test Case 1 – Youth Registration → Welcome Email

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

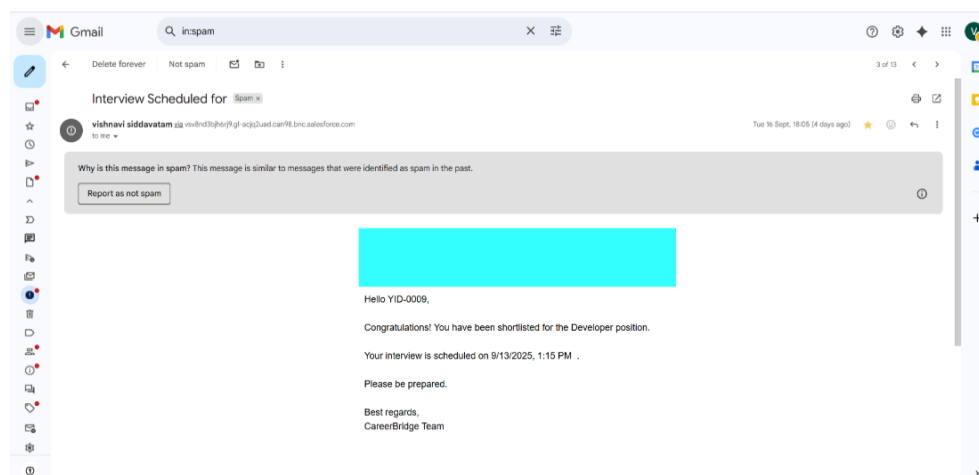
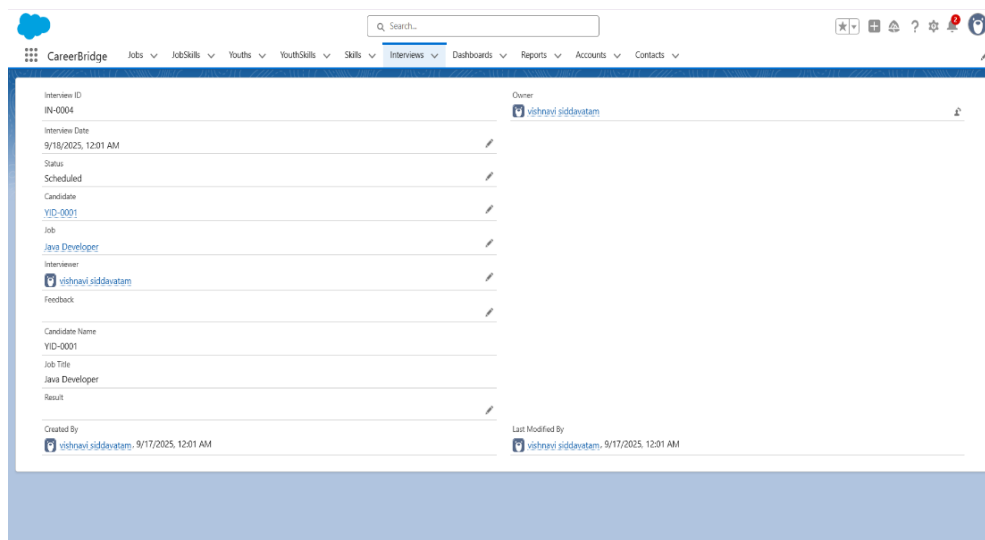
- Fill Youth form → Name = Vishnavi, Email = vishnavi@gmail.com, Skills = Java, Python.
- **Expected Result:**
 - Youth record saved in **Youth__c** object.
 - **Welcome Email** triggered:
“Hello , thank you for registering. Your skills (Python) are now recorded.”



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

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

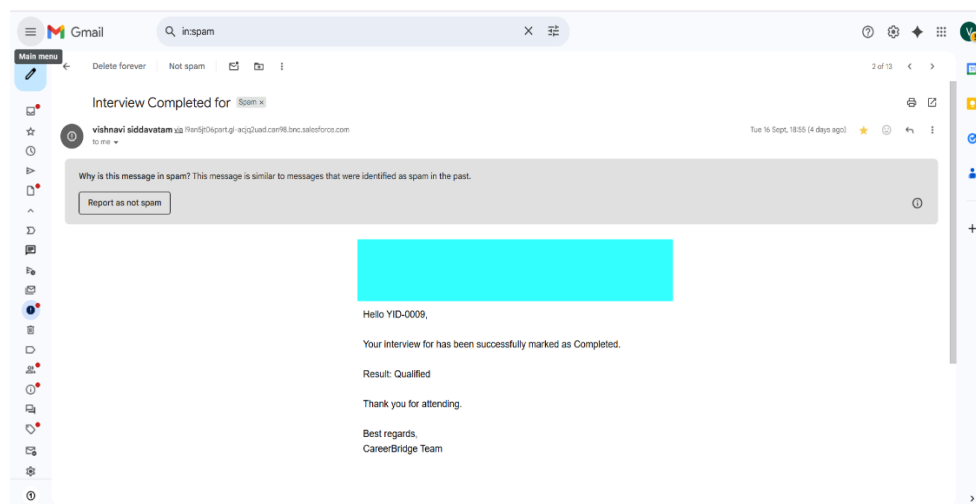
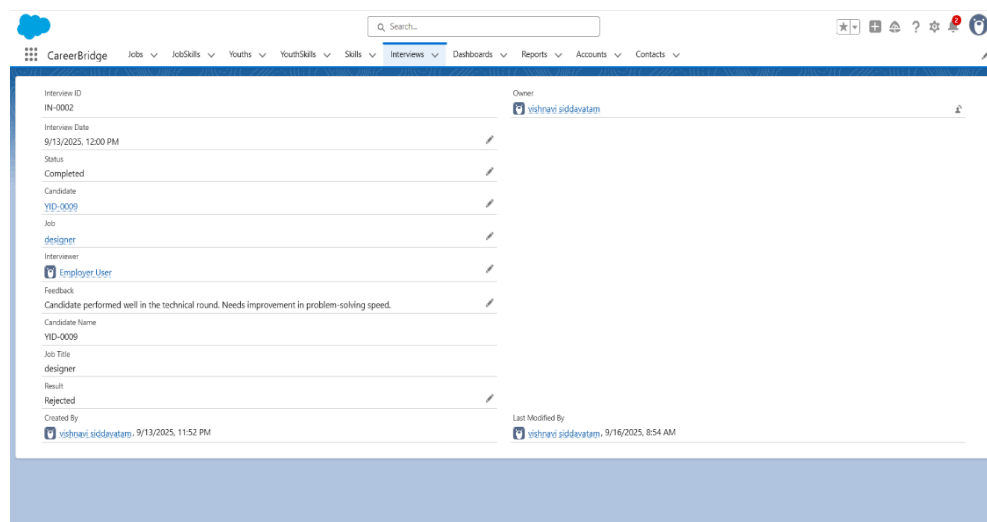
- Job Title = Java Developer, Location = Bangalore, Required Skill = Java.
- **Expected Result:**
 - Job record created in **Job__c** object.
 - System checks youth skills → Match found with Vishnavi.
 - **Interview__c** record auto-created with Status = Scheduled.
 - **Interview Scheduled Email** triggered:
“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.”

The screenshot shows the 'Youth YID-0009' form in the CareerBridge system. The form fields are: Youth ID (YID-0009), Owner (vishnavi siddavatam), Email (empty), Location (Hyderabad), Qualification (B.tech), Preferred Role (Developer), Created By (vishnavi siddavatam, 9/13/2025, 11:10 PM), and By (siddavatam, 9/14/2025, 12:08 AM). A red error message box is displayed over the form, stating: "We hit a snag. Review the errors on this page. • Email is required." Below the error message are 'Cancel' and 'Save' buttons.

The screenshot shows the 'Youth YID-0009' form in the CareerBridge system. The form fields are: Youth ID (YID-0009), Owner (vishnavi siddavatam), Email (vishnavi79@gmail.com), Location (empty), Qualification (B.tech), Preferred Role (Developer), Created By (vishnavi siddavatam, 9/13/2025, 11:10 PM), and By (siddavatam, 9/14/2025, 12:08 AM). A red error message box is displayed over the form, stating: "We hit a snag. Review the errors on this page. • Location is required." Below the error message are 'Cancel' and 'Save' buttons.

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