# Job Application Tracker

This capstone project is a comprehensive solution that keeps track of job applications for users in Salesforce. It will demonstrate your proficiency in various aspects of Salesforce development, from basic programming concepts and classes to asynchronous Apex, LWC, and integrations. The goal is to be able to track job applications that are found on multiple job boards/sites and synthesize relevant components into one location. This system should manage the entire job application lifecycle.

One guiding principle should be: would you use this system, or would you put it into your production org?

## Project Details

The capstone project can be done on a team (3 members max) or individually.

All team members should be working in separate orgs, and all metadata necessary should be stored in the GitHub repository.

Members are expected to collaborate and share ideas with one another. This could be done through Slack and virtual meetings.

Collaboration with members outside of your team is encouraged, but all aspects of the project should be understood and explainable by each member of your team.

You are provided an empty GitHub repository.

All requirements are not well defined on purpose to allow creativity and to prepare for independence in a work setting.

Ask questions to your instructor for further clarification of the requirements!

## Planning

### Week 1

Read and review all aspects of the project carefully.

Delegate requirements to teammates. This will require a meeting and project group agreement. When doing this keep in mind everyone's availability is not the same.

Set up the GitHub repository and configure individual Salesforce orgs.

Configure Job Applications object, related objects, and additional job application functionality.

### Week 2

Focus on more straightforward requirements and programmatic solutions.

Practice branching and merging into the GitHub repository.

### Week 3

Focus on complex automation and programmatic solutions.

Begin test classes for code written so far.

### Week 4

Finalize any features being built.

Add any additional polish to the code and record pages.

Complete test classes for all code.

Review teammate's code for best practices and formatting.

Prepare and merge any final code to the main branch of your repository for review.

Practice demoing/run-through of solutions created as a team.

Review the capstone presentation expectations and rubric.

Schedule your presentation with your instructor.

All metadata(classes, triggers, objects, etc...) needed to deploy the code should included in the repo.

### Week 5 - Final Review

Deliverables

Notify your instructor via Slack you are ready for review along with a link to your repository.

Provide login credentials to a Salesforce org that has the capstone project implemented.

No more changes to the repository should be made.

Rehearse your presentation with the team and prepare for the real presentation

Developer guests who are not your instructor may join the capstone presentation to add a different perspective.

Teams will be given 1 week after the presentation to make any revisions to their codebase and submit for an additional code review.

**Notes**

You probably won’t finish all the requirements but for features on the main branch should be “production ready” (test classes, formatted, following best practices, etc…)

It is recommended to focus on a few features that you know and are designed with best practices in mind instead of having multiple half-done features.

Feel free to use project management tools to break down and organize the project.

Salesforce is a powerful system, and many of the requirements could be solved using declarative features. Use declarative solutions for basic requirements like creating objects, standard email editor, and configuring page layouts. When dealing with business logic and functionalities, opt for a programmatic solution.

It is recommended to create a new Trailhead Playground or Developer org so that existing automation does not interfere with Capstone requirements/automation.

It is recommended to create a permission set to enable the sharing and field-level security of objects and fields.

# Overview

**Job Applications**: Develop a way to store and manage individual job applications. We should capture essential job-related details. It needs to associate with contacts, track scheduled interviews, maintain notes, manage tasks, and provide functionality to send emails directly from the record.

**Application Status**: For each step in the application process, programmatic automation will be created to generate tasks.

**Paycheck Estimation**: Create a feature that calculates the estimated take-home salary based on the offered salary and taxes that will be taken out of a user's paycheck. The values should be visible yearly and monthly.

**Primary Contact**: Create automation to always have the important contacts on the job application record.

**Paycheck Breakdown**: Build a way to rapidly break down a salary from a job. It should show the salary for 6-month, monthly, bi-weekly, and weekly intervals, so I can easily understand the payment distribution at a glance. The salary should be able to be changed very quickly, and the values should be updated in real time.

**Meeting Validation**: Create validations so that interview meetings cannot overlap. This will reduce the chance of double booking and missing an interview.

**Email Notifications**: Create a feature that sends an email notification to the user a day before their scheduled interview.

**Job Board Integration**: Implement an integration with an external job board API to populate job applications the user is interested in.

**Unit Tests**: Write comprehensive test classes for all of the code so that it can be deployed to production. (Test classes are not required for Lightning Web Components)

# Requirements

## Job Application Object

Create a custom object to store job applications.

Create the following application fields:

**Status**: This picklist field will track the current status of the job application. Options should include 'Saved', 'Applying', 'Applied', 'Interviewing', Negotiating, 'Accepted', and 'Closed'.

**Company Name**: A text or lookup field to store the name of the company where the job application is being sent.

**Position/Title**: A text field to keep track of the specific job title or position being applied for.

**Application Date**: A date field to record the date when the application was submitted.

**Salary**: A numerical field to capture the expected salary for the job position.

**Description**: A long text area to hold detailed information about the job, such as responsibilities, qualifications, benefits, etc.

**URL**: A URL field to store the web link of the job posting.

**Location**: A text/picklist field to indicate the job location or if it's remote.

**Follow-up date**: A date field to specify when to follow up on the application.

**Rating**: A picklist or numerical field to rate the attractiveness of the job on a predetermined scale, for example, 1-5.

**Notes**: A text field that can hold a lot of information.

**Primary Contact**: Lookup to a Contact that represents the main point of contact for the job. This could be a recruiter or employee at the company.

Add additional fields as needed.

## Additional Job Application Functionality

**Track Interviews**: Use Salesforce standard meeting/calendar/event functionality to track interviews. This object should store details about each interview related to a specific job application, such as the interview date, time, interviewer, interview type, interview format, and any notes from the interview. Enable salesforce activity timeline for easier access to create events on the application object. https://www.getweflow.com/blog/salesforce-activity-timeline

**Send Emails Directly from Record**: Utilize the Salesforce Standard Email functionality directly on the Job Application object. This provides the ability to quickly and easily send an email related to a job application, such as a follow-up message or thank you note, without leaving the Salesforce platform. Emails sent from the platform can be logged in the activity history related to the job application.

**Manage Tasks**: Use Salesforce standard task functionality to manage activities related to the job application, such as follow-up calls or emails. Relate Tasks to the Job Application and make it visible on the layout.

**Associate Applications with Multiple Contacts**: Create functionality to associate multiple contacts with a single job application. This feature aids in keeping track of pivotal individuals associated with the company, such as hiring managers, recruiters, and other relevant employees. It's a crucial part of building and managing professional networks during the job application process.

**Application Status Automation**:Create automation to create tasks when job application status is changed. Each task should have a due date, subject, and priority, making it easier to track what needs to be done and when.

Generate the following task records based on the application status

SAVED

* Check if the job description aligns with your interests and values
* Review the highlighted skills to see if the role is a good fit
* Research the company or role and mark your excitement level

APPLYING

* Find and research someone who works at the company and add them as a contact
* Set up an informational interview to learn more about the role/company
* Identify potential referrals to help get your application on the top of the pile
* Customize your work achievements using the job description keywords
* Submit your application on the company website if possible

APPLIED

* Reach out to the hiring manager or recruiter
* Follow up on your application via email weekly
* Continue identifying and saving similar job opportunities
* Set up weekly networking calls to explore similar companies/roles

INTERVIEWING

* Prepare your blurb or “tell me about yourself” response
* Practice answering behavioral interview questions
* Research the company and your interviewers
* Set up your virtual interview space and test your tech
* Send thank you emails within 24 hours

NEGOTIATING

* Research your market value and know your numbers
* Prepare your negotiation scripts
* Evaluate your offer and decline or accept

ACCEPTED

* Plan your resignation if applicable
* Take some time to relax and recharge
* Prepare for your first day of onboarding

CLOSED

* Send a follow-up email thanking the interviewer and asking for feedback
* Review your notes and reflect on areas of improvement

**Primary Contact Automation**: Create automation to set the primary contact on the job application if the primary contact is null and there is at least one related contact. Use the first contact available in the contact-related list or the first contact related to the Company(Account).

**Take-home Pay Estimation**: Federal Income Tax: Create additional fields and automation to estimate the yearly tax liabilities. For simplicity, assume you will file with a single status and use the standard deductions.

US Federal Income Tax Brackets for 2023 and example calculations can be found on NerdWallet. NerdWallet Calculator accounts for deductions, which may throw off your calculation if you are not implementing deductions. To reduce variance, switch to itemized deduction and set the amount to $0.

**Social Security**: Create additional fields and Apex automation to estimate the yearly Social Security liability. As of 2023 Social Security rate is 6.20%, but this number could change.

**Medicare Withholding**: Create additional fields and Apex automation to estimate the yearly Medicare withholdings liability. As of 2023, the Medicare withholdings rate is 1.45%, but this number could change.

**Take-home calculations**: Create additional fields and Apex automation using the salary, federal income tax, social security tax, and Medicare withholding to get an estimate of your take-home pay. https://smartasset.com/taxes/paycheck-calculator Have breakdowns for the yearly and monthly paychecks.

**Optional** - Expand on Income Tax automation to allow other filing statuses (single, head of household, married filing separately, or married filing jointly)

**Note**: If you are not from the United States, feel free to use your country's tax liability calculations. Use online calculators to validate your automations and notate the calculations used.

**Take-home Pay Calculator**

Create a lightning web component that can be placed on the Job Application object that can quickly calculate the take-home pay without needing to save the job application record. Display relevant pay calculations like Salary, Federal Tax, Medicare, Social Security, Take Home. Show other time internals of the take-home pay like yearly, six months, monthly, and bi-weekly.

Optional - Default the salary value based on the Job Application record.

**Calendar Validation**

Create code validation to stop double-booking (scheduling two meetings simultaneously) or scheduling meetings on the weekend. The automation should be on the Salesforce standard event object so that interview events cannot be scheduled at the same time.

Easy: Check if an existing event has the exact same start date/time. Don’t check the end date/time.

Hard: Check if the events overlap at any time, including both start and end dates. - hint (Convert the StartDateTime and EndDateTime of each event into a number of minutes since a fixed point in time (e.g., the Unix Epoch).)

**Email Reminders**

Create a scheduled process that sends an email reminder the day before an interview is scheduled. Include relevant interview information in the email reminder.

**Clean Up Stale Jobs Applications**

Create an asynchronous process that checks if a job application is stale and moves the record status to closed. Update the notes field that the job application was closed by an automated process.

**Stale Criteria:**

**Status** is not Closed or Accepted

**Follow-up Dat**e 30 days old or more

Find New Jobs

Using the Jooble API, retrieve jobs for Salesforce-related jobs and create them as ‘Saved’ job applications.

Jooble API is free but has a limit of 500 total requests you can do. If you run out of requests from testing, then sign up for a new API key.

* Easy: Create the Apex class that does the callout and use Apex Anonymous to run this code.
* Medium: Create a scheduled process that does the callout automatically and creates the job applications at an interval you decide.
* Hard: Create a lightning web component that does the callout and displays all the jobs in a table that can be selected and inserted as Job Applications.

API Documentation: https://jooble.org/api/about

Video Documentation: https://vimeo.com/863756136/c18342b5a9

Endpoint: https://jooble.org/api

keywords - keywords to search for jobs by;

location - location to search for jobs in;

radius (optional) - radius for search (will be converted to km) type: string

salary (optional) - minimum salary for search type: integer

page (optional) - to get jobs on the specified page type: string

resultonpage (optional) - number of jobs to be displayed on a page;

datecreatedfrom (optional) - to get jobs created after this date. type: string format: yyyy-mm-dd

companysearch (optional): -true - to search for keywords in the company name of jobs, not in the job title or description; -false - to search for keywords in the job title or description.

## Testing

Create Apex unit tests with at least 75% code coverage so that this code can be deployed to production. Include positive and negative tests for all the Apex code written.

Unit Tests for the Lightning Web Components are not required.

Last Update: September 2023