**Salesforce Job Application Tracker**

Job searching can often become chaotic. With this project, I bring structure and efficiency into the process. Using Salesforce's versatile capabilities, I built features that keep track of Job Applications and add layers of automation and validation.

**Salesforce Capstone Project**

Demonstrates the use of Apex Triggers, Apex Classes, Test Classes, Lightning Web Components (LWC), and Data Modeling with Junction Objects in Salesforce.

**Overview**

This project automates key Tasks in the Job Application process using Salesforce. Features include:

* Automated Paycheck calculations for Job Applications.
* Automatic assignment of a Primary Contact.
* Creation of follow-up Tasks depending on Job Application Status.
* A Lightning Web Component for quick Take Home Pay calculations.
* A Junction Object to link Contacts to Job Applications.
* Ensures consistency, reduces manual work, and helps Users manage Applications efficiently.

**Components**

**Apex Class and Trigger:** Job Application Automation

**File:** JobAppAutomation.cls

**Key Methods:**

calculatePaychecks(List<Job\_Application\_\_c> jobApps)

* Runs before insert or update.
* Calculates Yearly, 6 Month, Monthly, Bi-Weekly, and Weekly Paychecks.
* Calculates estimated Taxes (Federal, Social Security, Medicare).
* Calculates Take Home Pay (Yearly & Monthly).
* Automatically assigns the Primary Contact if not set.

assignPrimaryContact(List<Job\_Application\_\_c> jobApps)

* Ensures a Primary Contact is assigned if the Job Application has linked Contacts.

**Helper Method:**

* Centralizes repeated logic for follow-up Task creation.
* Improves readability and maintainability.
* Prevents recursion issues.
* All other Methods call this Helper internally.

createFollowUpTasks(List<Job\_Application\_\_c> jobApps)

* Runs after insert or update.
* Creates Tasks automatically based on Job Application Status.
* Bulk-safe.  Multiple Job Applications handled in one operation.
* Includes recursion prevention to avoid duplicate Tasks if the same Record triggers automation multiple times.
* Debug statements included to assist with troubleshooting and understanding automation flow.

**Trigger:** JobAppTrigger.trigger

* Fires before insert, before update, and after update on Job Application.
* Calls Methods from JobAppAutomation:
* calculatePaychecks() [before save].
* createFollowUpTasks() [after save].
* This ensures automation runs every time a Job Application is created or updated.

**Purpose:** Automates updates when a Job Application Record is created or updated.

**Key Features:**

* Calculates Paycheck Fields based on Salary.
* Creates Follow-Up Task Records automatically.
* Encapsulates logic in a Handler Class for reusability.

**Test Class**

**File:** JobAppAutomationTest.cls

**Purpose:** Validates the functionality of the Trigger and automation logic.

**Key Features:**

* Inserts and updates Job Application Records.
* Verifies that Fields are correctly calculated.
* Asserts that related Task Records are created.
* Ensures automation handles all possible Job Application Status values.
* Tests recursion prevention by attempting to run createFollowUpTasks multiple times.
* Assertions ensure no duplicate Tasks are created.

**Tests:**

* testCalculatePaychecksAndPrimaryContact - Verifies Paycheck calculations and automatic Primary Contact assignment.
* testBulkJobApplications\_Minimal - Tests bulk operations to ensure automation is bulk-safe.
* testFollowUpTaskCreation - Ensures follow-up Tasks are created correctly based on Job Application Status.
* testJobApplication\_NoSalary - Ensures Job Applications with no Salary default Paychecks to 0 and don't throw errors.

***Importance:*** *Ensures compliance with Salesforce's requirement of greater than or equal to 75% test coverage and verifies business logic correctness.*

**Lightning Web Component:** Take Home Pay Calculator

**Files:**

* payCalculator.js - Handles logic & calculations.
* payCalculator.html - Template with input and results.
* payCalculator.css - Salesforce-style formatting.
* payCalculator.js-meta.xml - Exposes the component on Record, App, and Home pages.

**Purpose:** A front-end calculator that lets Users input a Salary and instantly see estimated Take Home Pay after Federal Tax, Social Security, and Medicare.

**Key Features:**

* Real-time calculation as the User types.
* Uses SLDS (Salesforce Lightning Design System) for styling.
* Demonstrates client-side interactivity with LWC.

**Custom Object:** Job Application

* Stores each Job Application record with Fields like:
* Salary – The Applicant's Salary.
* Primary Contact – Automatically assigned from related Contacts.
* Paycheck Fields – Paycheck\_Yearly\_\_c, Paycheck\_Monthly\_\_c, etc.
* Tax Fields – Federal\_Tax\_\_c, Social\_Security\_\_c, Medicare\_\_c.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Contact**

* Represents Applicants or associated Contacts.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Junction Object:** Job Application Contact

**Purpose:** Represents a many-to-many relationship between Job Applications and Contacts.

**Why is this Important?**

* Realistically, one Contact (person) can have many Job Applications (applied to different jobs).
* One Job Application might involve multiple Contacts (i.e. candidate, references, recruiter).
* This is a many-to-many relationship and Salesforce handles that via a Junction Object.
* Adding this Object makes the data model more realistic and scalable.

A screenshot of a computer

AI-generated content may be incorrect.

**Business Value**

This Project Demonstrates:

* Realistic automation of business processes - Trigger and Handler - with bulk-safe Methods to handle many Job Applications at once.
* Dynamic Task creation, which guides Applicants at each stage of the process.
* A User-friendly front-end calculator with LWC.
* Strong data modeling with Junction Objects (to allow multiple Contacts per Job Application).
* Testing best practices for reliability and compliance (test coverage, which ensures automation works reliably in all scenarios).

**Roadmap Planning**

[x] Custom Object: Job Application

[x] Junction Object: Job Application Contact

[x] Paycheck Calculation Automation

[x] Primary Contact Assignment

[x] Lightning Web Component: Take Home Pay Calculator

[ ] Interview Event Tracking

[ ] Email Notifications / Reminders (Day-Before Interview)

[ ] Automatically Close Stale Applications (After 30+ Days)

[ ] Job Board Integration (Pull Jobs into Salesforce)

[ ] Reports & Dashboards