Apex Specialist

(Superbadge)

What You'll Be Doing to Earn This Superbadge

1. Automate record creation using Apex triggers
2. Synchronize Salesforce data with an external system using asynchronous REST callouts
3. Schedule synchronization using Apex code
4. Test automation logic to confirm Apex trigger side effects
5. Test integration logic using callout mocks
6. Test scheduling logic to confirm action gets queued

Concepts Tested in This Superbadge

- Apex Triggers
- Asynchronous Apex
- Apex Integration
- Apex Testing

Task 1 : Automated Record Creation

Create a Maintainance Request For each new Equipment automatically.

MaintenanceRequestHelper.apxc

```
public with sharing class MaintenanceRequestHelper {
    public static void updateworkOrders(List<Case> updWorkOrders, Map<Id,Case> nonUpdCaseMap) {
        Set<Id> validIds = new Set<Id>();


        For (Case c : updWorkOrders){
            if (nonUpdCaseMap.get(c.Id).Status != 'Closed' && c.Status == 'Closed'){
                if (c.Type == 'Repair' || c.Type == 'Routine Maintenance'){
                    validIds.add(c.Id);


                }
            }
```

```
        }

        if (!validIds.isEmpty()){
            List<Case> newCases = new List<Case>();
            Map<Id,Case> closedCasesM = new Map<Id,Case>([SELECT Id, Vehicle__c, Equipment__c, Equipment__r.Maintenance_Cycle__c,(SELECT Id,Equipment__c,Quantity__c FROM Equipment_Maintenance_Items__r)
                                                                FROM Case WHERE Id IN :validIds]);
            Map<Id,Decimal> maintenanceCycles = new Map<ID,Decimal>();
            AggregateResult[] results = [SELECT Maintenance_Request__c, MIN(Equipment__r.Maintenance_Cycle__c)cycle FROM Equipment_Maintenance_Item__c WHERE Maintenance_Request__c IN :ValidIds GROUP BY Maintenance_Request__c];

        for (AggregateResult ar : results){
            maintenanceCycles.put((Id) ar.get('Maintenance_Request__c'), (Decimal) ar.get('cycle'));
        }

            for(Case cc : closedCasesM.values()){
                Case nc = new Case (
                    ParentId = cc.Id,
                Status = 'New',
                    Subject = 'Routine Maintenance',
                    Type = 'Routine Maintenance',
                    Vehicle__c = cc.Vehicle__c,
                    Equipment__c =cc.Equipment__c,
                    Origin = 'Web',
                    Date_Reported__c = Date.Today()

                );

                If (maintenanceCycles.containskey(cc.Id)){
                    nc.Date_Due__c = Date.today().addDays((Integer) maintenanceCycles.get(cc.Id));
                } else {
                    nc.Date_Due__c = Date.today().addDays((Integer) cc.Equipment__r.maintenance_Cycle__c);
                }

                newCases.add(nc);
```

```
            }

            insert newCases;

            List<Equipment_Maintenance_Item__c> clonedWPs = new List<Equipment_Maintenance_Item__c>();
            for (Case nc : newCases){
                for (Equipment_Maintenance_Item__c wp : closedCasesM.get(nc.ParentId).Equipment_Maintenance_Items__r){
                    Equipment_Maintenance_Item__c wpClone = wp.clone();
                    wpClone.Maintenance_Request__c = nc.Id;
                    ClonedWPs.add(wpClone);

                }
            }
            insert ClonedWPs;
        }
    }
}
```

MaitenanceRequest.apxt

```
 trigger MaintenanceRequest on Case (before update, after update) {

    if(Trigger.isUpdate && Trigger.isAfter){

        MaintenanceRequestHelper.updateWorkOrders(Trigger.New, Trigger.OldMap);

    }

}
```

Task 2: Synchronize Salesforce data with an external system

1. Make a REST callout to an external warehouse system to get a list of equipment that needs to be updated.
2. The callout's JSON response returns the equipment records that you upsert in Salesforce.

WarehouseCalloutService.apxc :-

```
public with sharing class WarehouseCalloutService implements Queueable {
    private static final String WAREHOUSE_URL = 'https://th-superbadge-

    @future(callout=true)
    public static void runWarehouseEquipmentSync(){
        Http http = new Http();
        HttpRequest request = new HttpRequest();

        request.setEndpoint(WAREHOUSE_URL);
        request.setMethod('GET');
        HttpResponse response = http.send(request);

        List<Product2> warehouseEq = new List<Product2>();

        if (response.getStatusCode() == 200){
            List<Object> jsonResponse = (List<Object>)JSON.deserializeUntyped(response.getBody());
            System.debug(response.getBody());


            for (Object eq : jsonResponse){
                Map<String,Object> mapJson = (Map<String,Object>)eq;
                Product2 myEq = new Product2();
                myEq.Replacement_Part__c = (Boolean) mapJson.get('replacement');
                myEq.Name = (String) mapJson.get('name');
                myEq.Maintenance_Cycle__c = (Integer) mapJson.get('maintenanceperiod');
                myEq.Lifespan_Months__c = (Integer) mapJson.get('lifespan');
                myEq.Cost__c = (Integer) mapJson.get('cost');
                myEq.Warehouse_SKU__c = (String) mapJson.get('sku');
                myEq.Current_Inventory__c = (Double) mapJson.get('quantity');
                myEq.ProductCode = (String) mapJson.get('_id');
                warehouseEq.add(myEq);
```

```
            }

            if (warehouseEq.size() > 0){
                upsert warehouseEq;
                System.debug('Your equipment was synced with the warehouse

            }
        }
    }

    public static void execute (QueueableContext context){
        runWarehouseEquipmentSync();
    }

}
```

open execute anonymous window (CTRl+E) and run this method

```
System.enqueueJob(new WarehouseCalloutService());
```

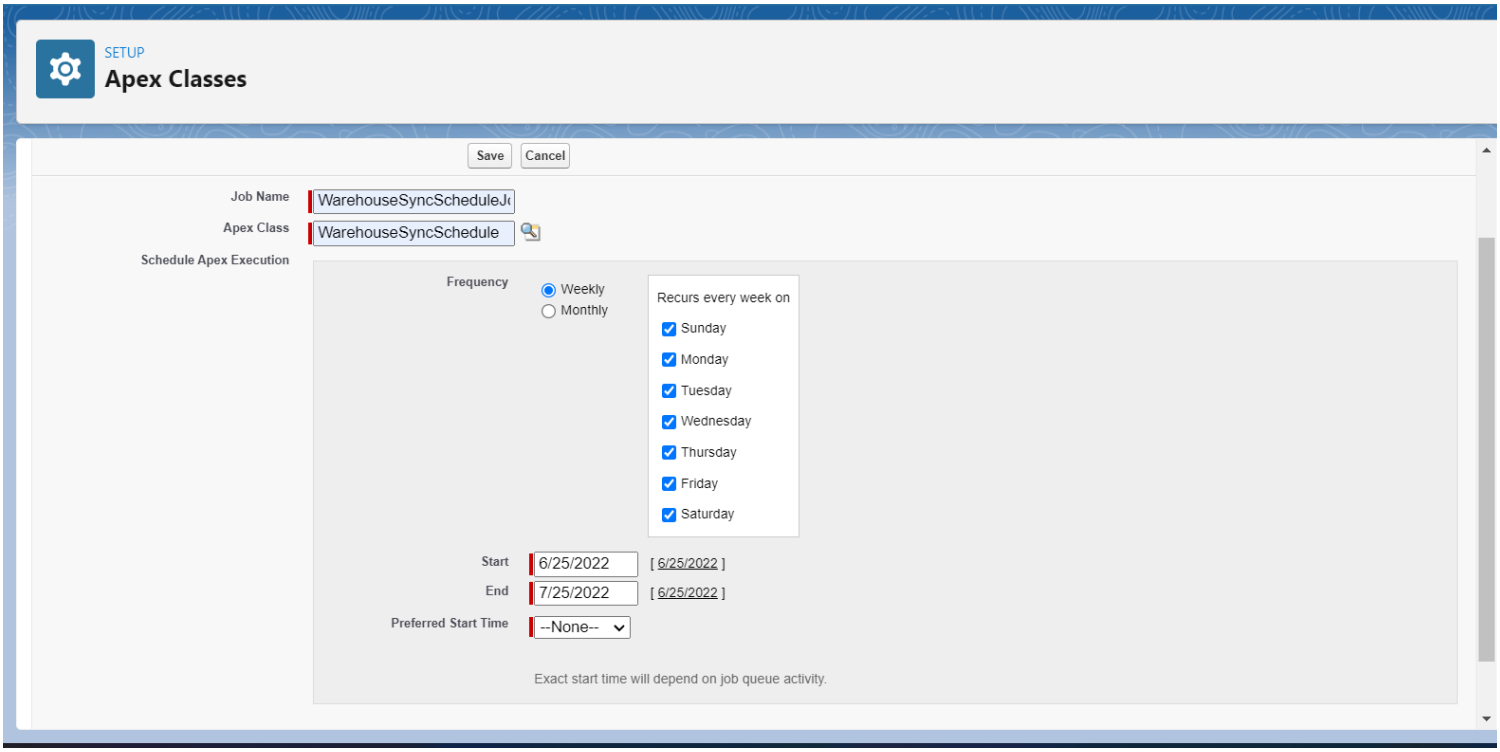
Task 3 : Schedule synchronization using Apex code

WarehouseSyncShedule.apxc :-

```
global with sharing class WarehouseSyncSchedule implements Schedulable{
    global void execute(SchedulableContext ctx){
        System.enqueueJob(new WarehouseCalloutService());
    }
}
```

Create A job schedule

Job Name = WarehouseSyncScheduleJob
Apex Class = WarehouseSyncSchedule



Task 4 : Test automation logic

Testing the generated automation logic and achieveing 100% code coverage.

MaintenanceRequestHelperTest.apxc :

```
@istest
public with sharing class MaintenanceRequestHelperTest {

    private static final string STATUS_NEW = 'New';
    private static final string WORKING = 'Working';
    private static final string CLOSED = 'Closed';
    private static final string REPAIR = 'Repair';
    private static final string REQUEST_ORIGIN = 'Web';
    private static final string REQUEST_TYPE = 'Routine Maintenance';
    private static final string REQUEST_SUBJECT = 'Testing subject';

    PRIVATE STATIC Vehicle__c createVehicle(){
        Vehicle__c Vehicle = new Vehicle__C(name = 'SuperTruck');
        return Vehicle;
    }

    PRIVATE STATIC Product2 createEq(){
        product2 equipment = new product2(name = 'SuperEquipment',
                                          lifespan_months__C = 10,
                                          maintenance_cycle__C = 10,
                                          replacement_part__c = true);
        return equipment;
    }

    PRIVATE STATIC Case createMaintenanceRequest(id vehicleId, id equipmentId){
        case cs = new case(Type=REPAIR,
                           Status=STATUS_NEW,
                           Origin=REQUEST_ORIGIN,
                           Subject=REQUEST_SUBJECT,
                           Equipment__c=equipmentId,
                           Vehicle__c=vehicleId);
        return cs;
    }

    PRIVATE STATIC Equipment_Maintenance_Item__c createWorkPart(id
```

```
equipmentId,id requestId){
        Equipment_Maintenance_Item__c wp = new Equipment_Maintenance_Item__c(Equipment__c = equipmentId,
Maintenance_Request__c = requestId);
        return wp;
    }


    @istest
    private static void testMaintenanceRequestPositive(){
        Vehicle__c vehicle = createVehicle();
        insert vehicle;
        id vehicleId = vehicle.Id;

        Product2 equipment = createEq();
        insert equipment;
        id equipmentId = equipment.Id;

        case somethingToUpdate = createMaintenanceRequest(vehicleId,equipmentId);
        insert somethingToUpdate;

        Equipment_Maintenance_Item__c workP = createWorkPart(equipmentId,somethingToUpdate.id);
        insert workP;

        test.startTest();
        somethingToUpdate.status = CLOSED;
        update somethingToUpdate;
        test.stopTest();

        Case newReq = [Select id, subject, type, Equipment__c, Date_Reported__c, Vehicle__c, Date_Due__c
                      from case
                      where status =:STATUS_NEW];

        Equipment_Maintenance_Item__c workPart = [select id
                                                  from Equipment_Maintenance_Item__c
                                                  where Maintenance_Request__c =:newReq.Id];

```

```
        system.assert(workPart != null);
        system.assert(newReq.Subject != null);
        system.assertEquals(newReq.Type, REQUEST_TYPE);
        SYSTEM.assertEquals(newReq.Equipment__c, equipmentId);
        SYSTEM.assertEquals(newReq.Vehicle__c, vehicleId);
        SYSTEM.assertEquals(newReq.Date_Reported__c, system.today());
    }

    @istest
    private static void testMaintenanceRequestNegative(){
        Vehicle__C vehicle = createVehicle();
        insert vehicle;
        id vehicleId = vehicle.Id;
        product2 equipment = createEq();
        insert equipment;
        id equipmentId = equipment.Id;

        case emptyReq = createMaintenanceRequest(vehicleId,equipmentId);
        insert emptyReq;

        Equipment_Maintenance_Item__c workP = createWorkPart(equipmentId, emptyReq.Id);
        insert workP;
        test.startTest();
        emptyReq.Status = WORKING;
        update emptyReq;
        test.stopTest();

        list<case> allRequest = [select id
                                 from case];

                  Equipment_Maintenance_Item__c workPart = [select id
                                                            from Equipment_Maintenance_Item__c
                                                            where Maintenance_Request__c = :emptyReq.Id];

                  system.assert(workPart != null);
                  system.assert(allRequest.size() == 1);
              }
              @istest
              private static void testMaintenanceRequestBulk(){
```

```
                    list<Vehicle__C> vehicleList = new list<Vehicle__C>();
                    list<Product2> equipmentList = new list<Product2>();
                    list<Equipment_Maintenance_Item__c> workPartList = new list<Equipment_Maintenance_Item__c>();
                    list<case> requestList = new list<case>();
                    list<id> oldRequestIds = new list<id>();
                    for(integer i = 0; i < 300; i++){
                       vehicleList.add(createVehicle());
                        equipmentList.add(createEq());
                    }
                    insert vehicleList;
                    insert equipmentList;
                    for(integer i = 0; i < 300; i++){
                        requestList.add(createMaintenanceRequest(vehicleList.get(i).id, equipmentList.get(i).id));
                    }
                    insert requestList;

                    for(integer i = 0; i < 300; i++){
                        workPartList.add(createWorkPart(equipmentList.get(i).id, requestList.get(i).id));
                    }
                    insert workPartList;

                    test.startTest();
                    for(case req : requestList){
                        req.Status = CLOSED;
                        oldRequestIds.add(req.Id);
                    }
                    update requestList;
                    test.stopTest();

                    list<case> allRequests = [select id
                                              from case
                                              where status =: STATUS_NEW];

                    list<Equipment_Maintenance_Item__c> workParts = [select id
                                                                     from Equipment_Maintenance_Item__c
                                                                     where
```

```
Maintenance_Request__c in: oldRequestIds];
                    system.assert(allRequests.size() == 300);
                }
            }
```

MaintenanceRequestHelper.apxc :

```
public with sharing class MaintenanceRequestHelper {
    public static void updateworkOrders(List<Case> updWorkOrders, Map<Id,Case> nonUpdCaseMap) {
        Set<Id> validIds = new Set<Id>();

        For (Case c : updWorkOrders){
            if (nonUpdCaseMap.get(c.Id).Status != 'Closed' && c.Status == 'Closed'){
                if (c.Type == 'Repair' || c.Type == 'Routine Maintenance'){
                    validIds.add(c.Id);
                }
            }
        }
        if (!validIds.isEmpty()){
            List<Case> newCases = new List<Case>();
            Map<Id,Case> closedCasesM = new Map<Id,Case>([SELECT Id, Vehicle__c, Equipment__c, Equipment__r.Maintenance_Cycle__c,(SELECT Id,Equipment__c,Quantity__c FROM Equipment_Maintenance_Items__r)
                                                                FROM Case WHERE Id IN :validIds]);
            Map<Id,Decimal> maintenanceCycles = new Map<ID,Decimal>();
            AggregateResult[] results = [SELECT Maintenance_Request__c, MIN(Equipment__r.Maintenance_Cycle__c)cycle FROM Equipment_Maintenance_Item__c WHERE Maintenance_Request__c IN :ValidIds GROUP BY Maintenance_Request__c];
        for (AggregateResult ar : results){
            maintenanceCycles.put((Id) ar.get('Maintenance_Request__c'), (Decimal) ar.get('cycle'));
        }
            for(Case cc : closedCasesM.values()){
                Case nc = new Case (
                    ParentId = cc.Id,
                Status = 'New',
                    Subject = 'Routine Maintenance',
                    Type = 'Routine Maintenance',
```

```
                    Vehicle__c = cc.Vehicle__c,
                    Equipment__c =cc.Equipment__c,
                    Origin = 'Web',
                    Date_Reported__c = Date.Today()
                );
                If (maintenanceCycles.containskey(cc.Id)){
                    nc.Date_Due__c = Date.today().addDays((Integer) maintenanceCycles.get(cc.Id));
                }
                newCases.add(nc);
            }
            insert newCases;
            List<Equipment_Maintenance_Item__c> clonedWPs = new List<Equipment_Maintenance_Item__c>();
            for (Case nc : newCases){
                for (Equipment_Maintenance_Item__c wp : closedCasesM.get(nc.ParentId).Equipment_Maintenance_Items__r){
                    Equipment_Maintenance_Item__c wpClone = wp.clone();
                    wpClone.Maintenance_Request__c = nc.Id;
                    ClonedWPs.add(wpClone);
                }
            }
            insert ClonedWPs;
        }
    }
}
```

MaintenanceRequest.apxt :

```
trigger MaintenanceRequest on Case (before update, after update) {
    if(Trigger.isUpdate && Trigger.isAfter){
        MaintenanceRequestHelper.updateWorkOrders(Trigger.New, Trigger.OldMap);
    }
}
```

Task 5 : Test callout logic

Test the external callout logic and achieveing 100% code coverage.

WarehouseCalloutService.apxc :

```
public with sharing class WarehouseCalloutService {
    private static final String WAREHOUSE_URL = 'https://th-superbadge-

    //@future(callout=true)
    public static void runWarehouseEquipmentSync(){

        Http http = new Http();
        HttpRequest request = new HttpRequest();
        request.setEndpoint(WAREHOUSE_URL);
        request.setMethod('GET');
        HttpResponse response = http.send(request);

        List<Product2> warehouseEq = new List<Product2>();

        if (response.getStatusCode() == 200){
            List<Object> jsonResponse = (List<Object>)JSON.deserializeUntyped(response.getBody());
            System.debug(response.getBody());

            for (Object eq : jsonResponse){
                Map<String,Object> mapJson = (Map<String,Object>)eq;
                Product2 myEq = new Product2();
                myEq.Replacement_Part__c = (Boolean) mapJson.get('replacement');
                myEq.Name = (String) mapJson.get('name');
                myEq.Maintenance_Cycle__c = (Integer) mapJson.get('maintenanceperiod');
                myEq.Lifespan_Months__c = (Integer) mapJson.get('lifespan');
                myEq.Cost__c = (Decimal) mapJson.get('lifespan');
                myEq.Warehouse_SKU__c = (String) mapJson.get('sku');
                myEq.Current_Inventory__c = (Double) mapJson.get('quantity');
                warehouseEq.add(myEq);
            }

            if (warehouseEq.size() > 0){
                upsert warehouseEq;
```

```
                System.debug('Your equipment was synced with the warehouse

                System.debug(warehouseEq);
            }

        }
    }
}
```

WarehouseCalloutServiceTest.apxc :

```
@isTest

private class WarehouseCalloutServiceTest {
    @isTest
    static void testWareHouseCallout(){
        Test.startTest();
        // implement mock callout test here
        Test.setMock(HTTPCalloutMock.class, new WarehouseCalloutServiceMock());
        WarehouseCalloutService.runWarehouseEquipmentSync();
        Test.stopTest();
        System.assertEquals(1, [SELECT count() FROM Product2]);
    }     }
```

WarehouseCalloutServiceMock.apxc :

```
@isTest
global class WarehouseCalloutServiceMock implements HttpCalloutMock {
    // implement http mock callout
    global static HttpResponse respond(HttpRequest request){

        System.assertEquals('https://th-superbadge-
));
        System.assertEquals('GET', request.getMethod());

```

```
        // Create a fake response
        HttpResponse response = new HttpResponse();
        response.setHeader('Content-Type', 'application/json');
        response.setBody('[{"_id":"55d66226726b611100aaf741","replacement":false,"qu
        response.setStatusCode(200);
        return response;
    }
}
```

Task 6 : Test scheduling logic

Test Scheduling logic and 100% code coverage.

WarehouseSyncSchedule.apxc :

```
global class WarehouseSyncSchedule implements Schedulable {
    global void execute(SchedulableContext ctx) {

        WarehouseCalloutService.runWarehouseEquipmentSync();
    }
}
```

WarehouseSyncScheduleTest.apxc :-

```
@isTest
public class WarehouseSyncScheduleTest {

    @isTest static void WarehousescheduleTest(){
        String scheduleTime = '00 00 01 * * ?';
        Test.startTest();
        Test.setMock(HttpCalloutMock.class, new WarehouseCalloutServiceMock());
        String jobID=System.schedule('Warehouse Time To Schedule to Test', scheduleTime, new WarehouseSyncSchedule());
        Test.stopTest();
        CronTrigger a=[SELECT Id FROM CronTrigger where NextFireTime > today];
        System.assertEquals(jobID, a.Id,'Schedule ');


    }
}
```

Process Automation Specialist

(*Superbadge*)

What You'll Be Doing to Earn This Superbadge

1. Automate lead ownership using assignment rules
2. Enforce data integrity with formula fields and validation rules
3. Create a custom object in a master-detail relationship to a standard object
4. Define an opportunity sales process using stages, record types, and validation rules
5. Automate business processes to send emails, create related records, and submit opportunities for approval
6. Create a flow to display dynamic information on a Lightning record page
7. Create a process to evaluate and update records

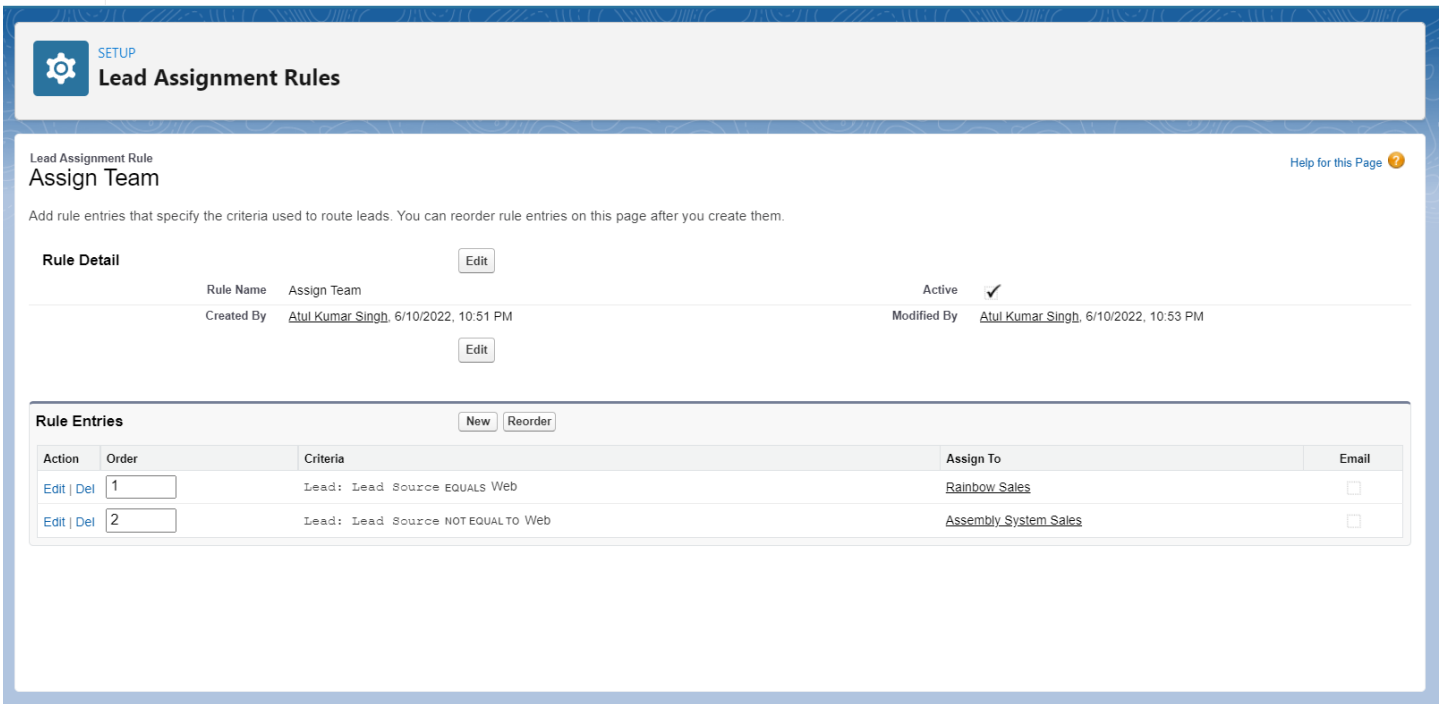
Concepts Tested in This Superbadge

- Validations and Formulas
- Sales Process
- Process Builder
- Flow

Task 1 : Automate Leads

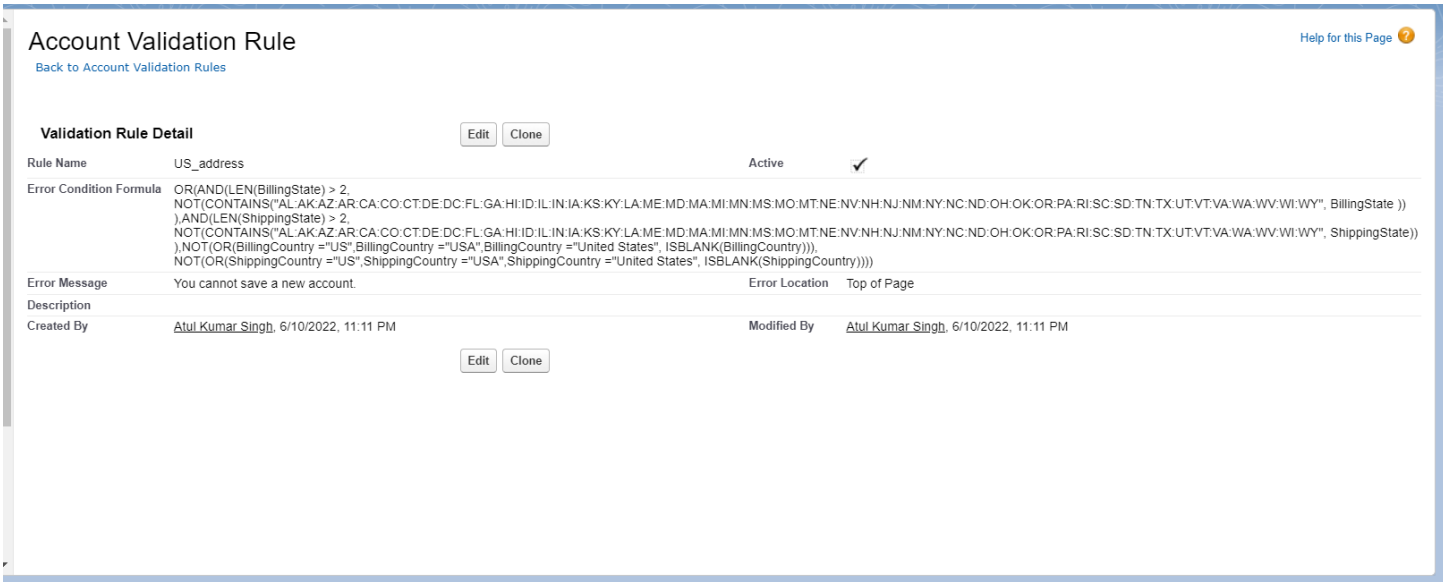
Make sure that all leads have the standard 2-character US state abbreviation in the State/Province field and either US, USA, United States, or nothing in the Country field.

The Rainbow sales team should be handling all the leads that come from the web. A bunch of other leads come from our partners and from lists that we buy--all those leads are potential buyers of our assembly systems. So the Assembly System team should get those leads.



Task 2 : Automate Accounts

Make sure that nobody can save a new account unless the shipping and billing state fields are valid US state abbreviations, and the country field is either blank or US, USA, or United States.



Account Validation Rule

Back to Account Validation Rules

Help for this Page

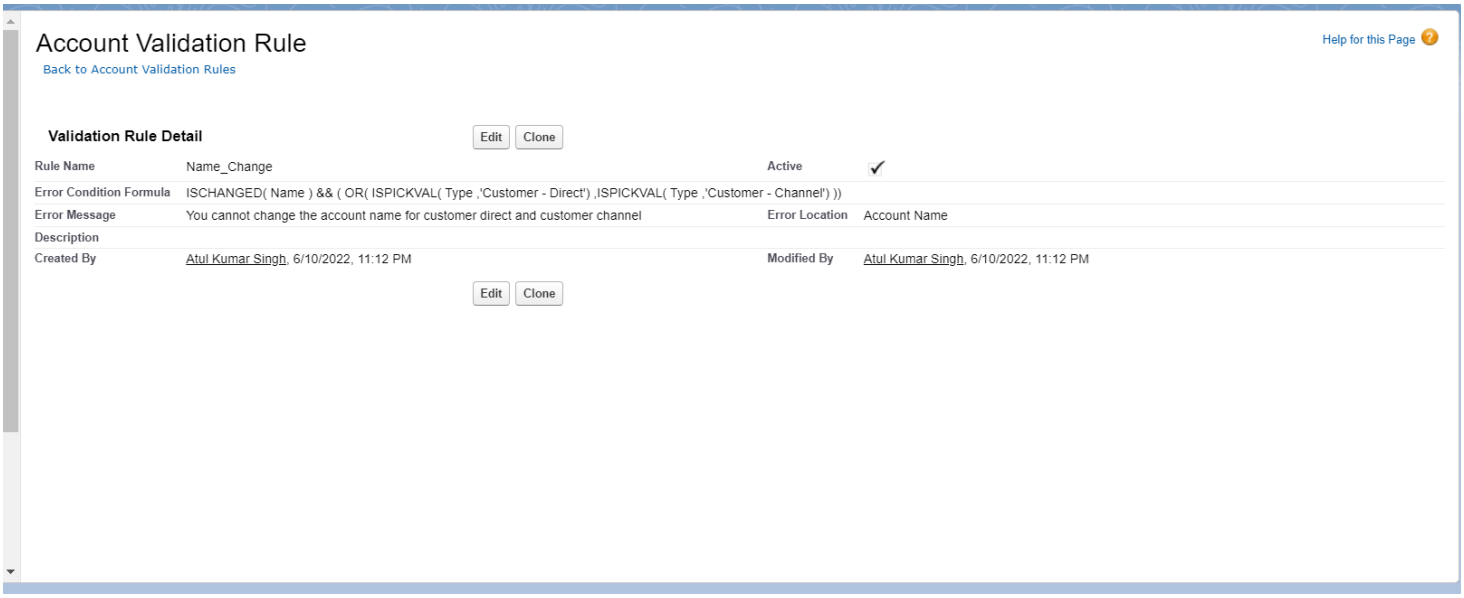
Validation Rule Detail

Edit Clone

Rule Name	US_address	Active	✓
Error Condition Formula	OR(AND(LEN(BillingState) > 2, NOT(CONTAINS("AL:AK:AZ:AR:CA:CO:CT:DE:DC:FL:GA:HI:ID:IL:IN:IA:KS:KY:LA:ME:MD:MA:MI:MN:MS:MO:MT:NE:NV:NH:NJ:NM:NY:NC:ND:OH:OK:OR:PA:RI:SC:SD:TN:TX:UT:VT:VA:WA:WV:WI:WY", BillingState))),AND(LEN(ShippingState) > 2, NOT(CONTAINS("AL:AK:AZ:AR:CA:CO:CT:DE:DC:FL:GA:HI:ID:IL:IN:IA:KS:KY:LA:ME:MD:MA:MI:MN:MS:MO:MT:NE:NV:NH:NJ:NM:NY:NC:ND:OH:OK:OR:PA:RI:SC:SD:TN:TX:UT:VT:VA:WA:WV:WI:WY", ShippingState))),NOT(OR(BillingCountry ="US",BillingCountry ="USA",BillingCountry ="United States", ISBLANK(BillingCountry))), NOT(OR(ShippingCountry ="US",ShippingCountry ="USA",ShippingCountry ="United States", ISBLANK(ShippingCountry))))		
Error Message	You cannot save a new account.	Error Location	Top of Page
Description			
Created By	Atul Kumar Singh, 6/10/2022, 11:11 PM	Modified By	Atul Kumar Singh, 6/10/2022, 11:11 PM

Edit Clone

If the account type is either “Customer - Direct” or “Customer - Channel” we don’t want anybody to change the name.



Account Validation Rule

Back to Account Validation Rules

Help for this Page

Validation Rule Detail

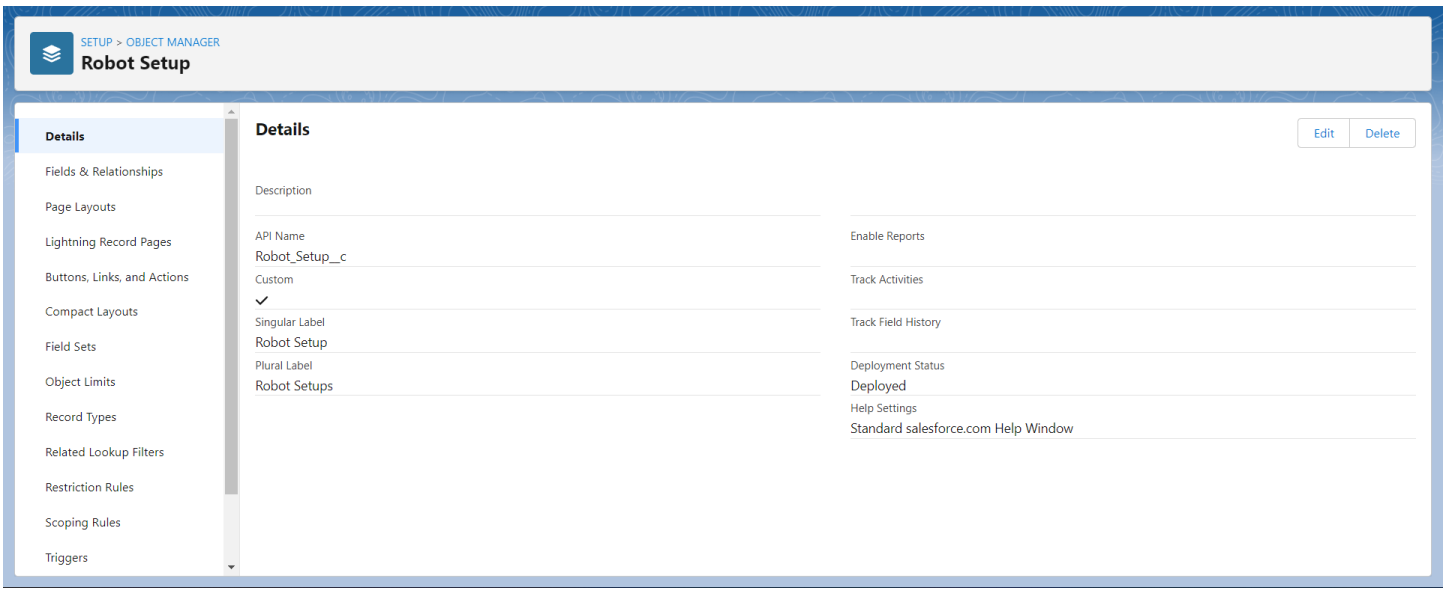
Edit Clone

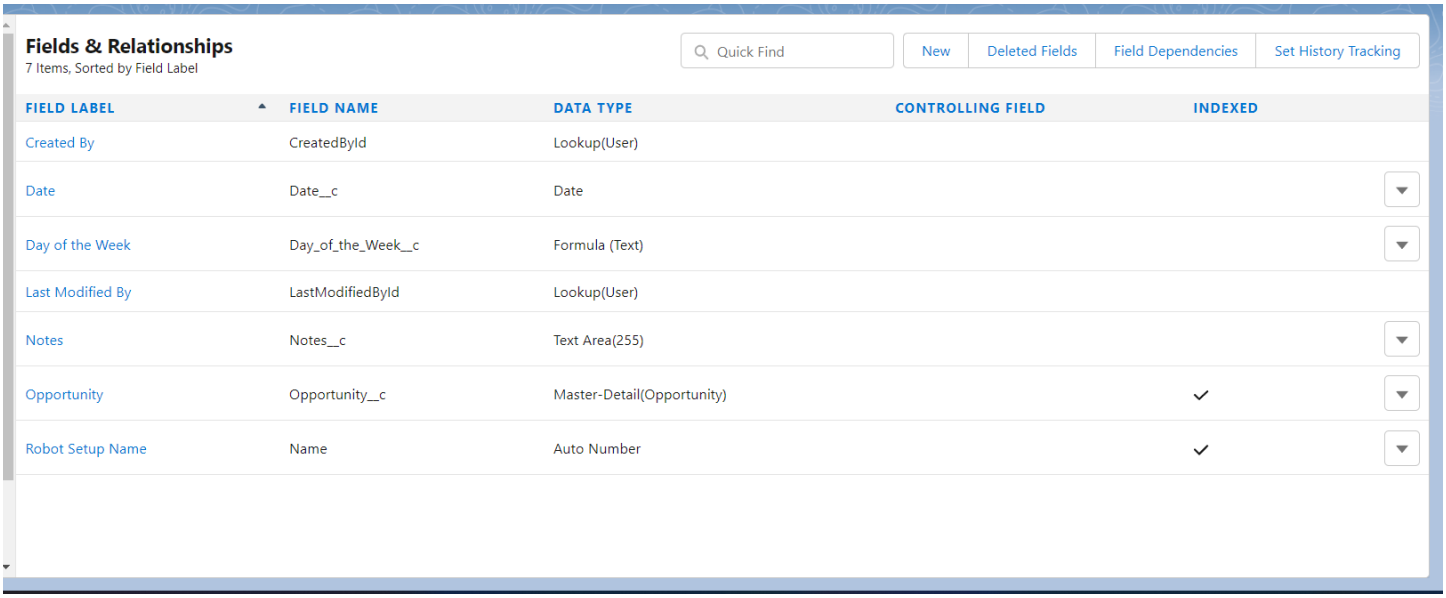
Rule Name	Name_Change	Active	✓
Error Condition Formula	ISCHANGED(Name) && (OR(ISPICKVAL(Type ,'Customer - Direct') ,ISPICKVAL(Type ,'Customer - Channel')))		
Error Message	You cannot change the account name for customer direct and customer channel	Error Location	Account Name
Description			
Created By	Atul Kumar Singh, 6/10/2022, 11:12 PM	Modified By	Atul Kumar Singh, 6/10/2022, 11:12 PM

Edit Clone

Task 3 : Create Robot Setup Object

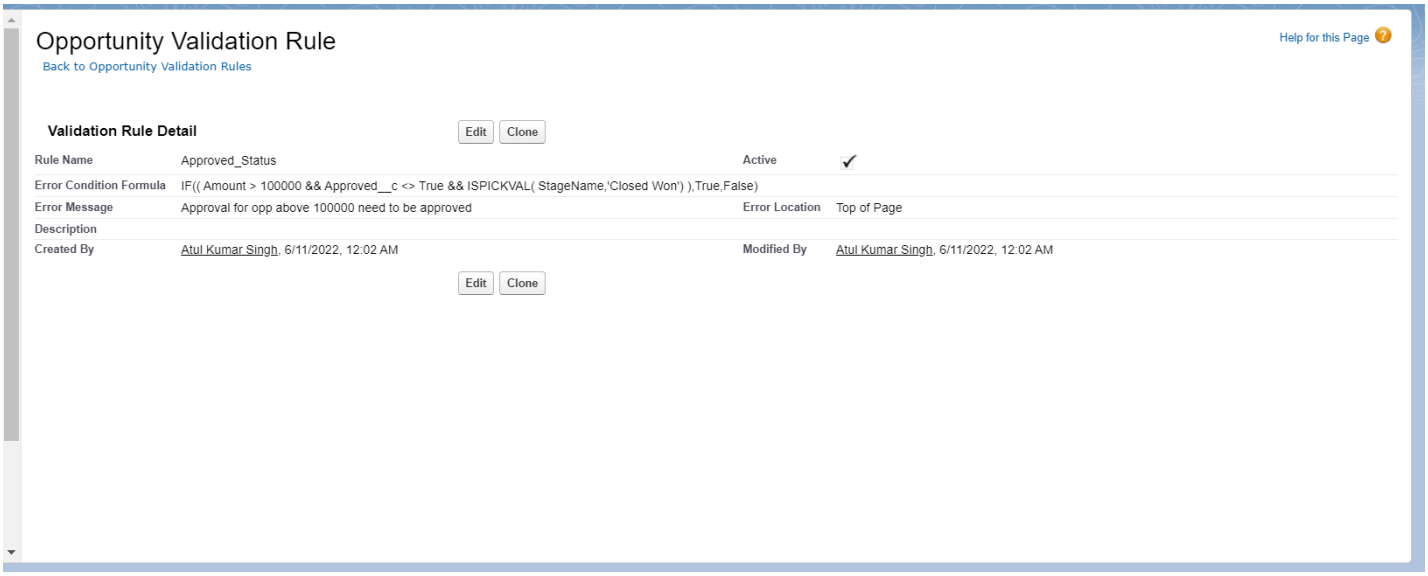
- *For every sale we close and win, we need a setup record created that we can use for scheduling. If we delete a deal for any reason the setup record should also be deleted. We need to see the setup date and we also need a spot for the technician or the rep to make notes about the setup.*
- *The records can just be auto-numbered--they don't need the name of the account or anything.*





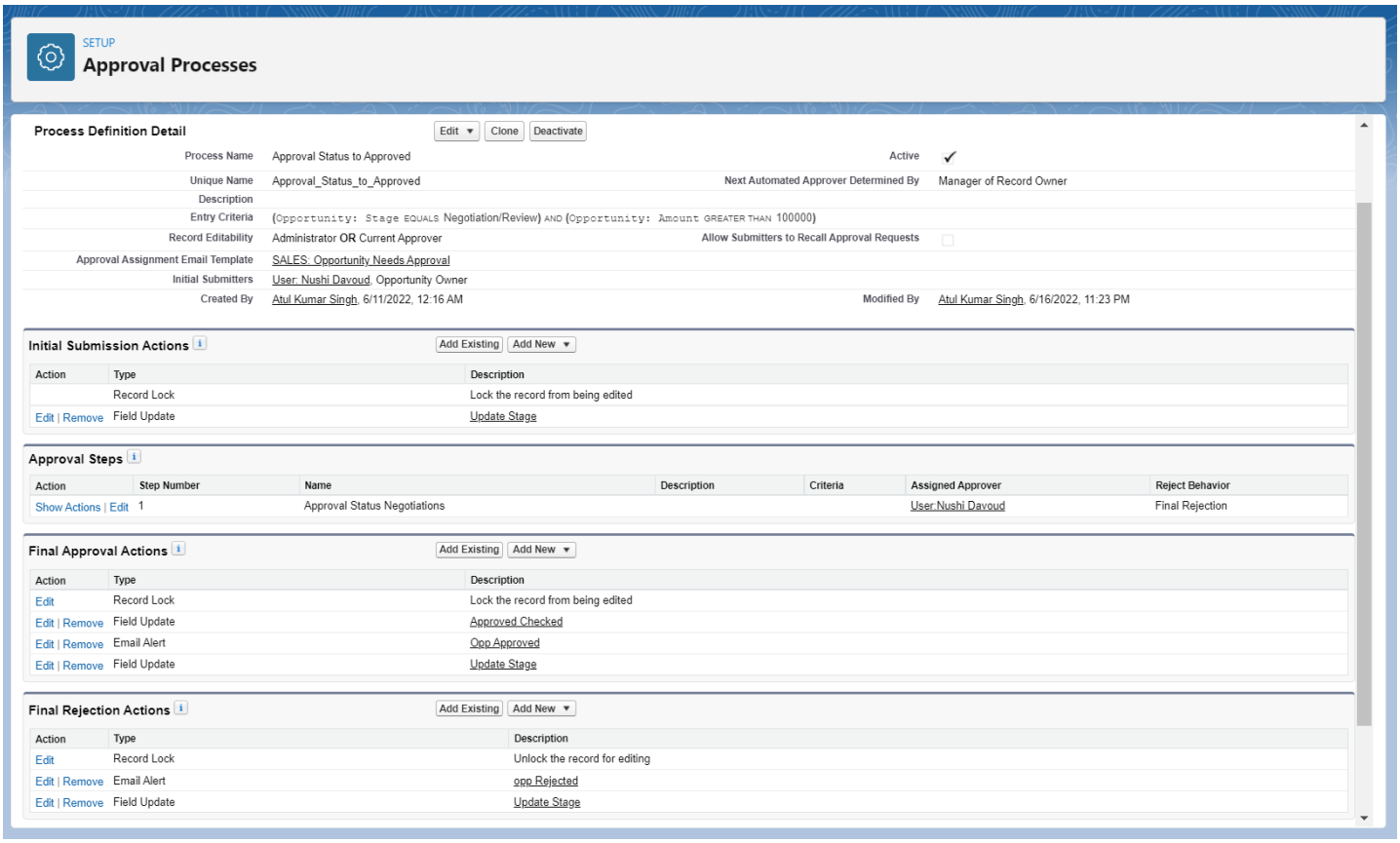
Task 4 : Create Sales Process and Validate Opportunities

The biggest deals--anything over $100K--have to be approved before they can close. We should have a way to show on the record that the deal is approved--maybe you can add an "Approved" checkbox to the opportunity. Only system administrators like you and sales managers should be able to check it.

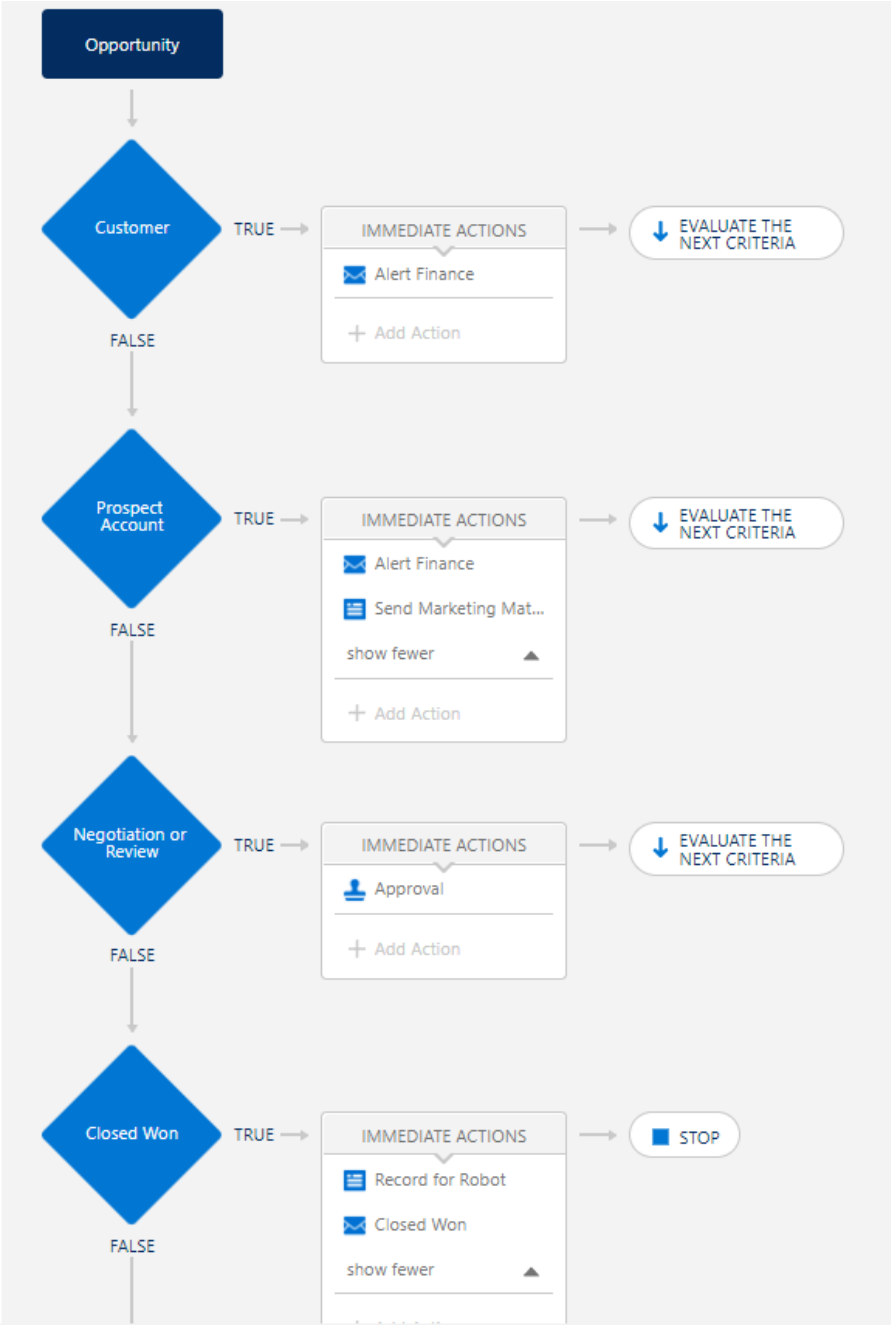


Task 5 : Automate Opportunities

Send an email to the finance group whenever an opportunity is created for a prospect or customer account, and at the same time to create a task for the account owner, but only if the account is a prospect. And when we win a deal, we want another email sent to the finance group.

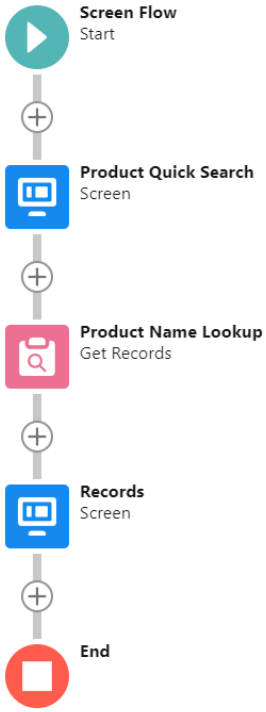


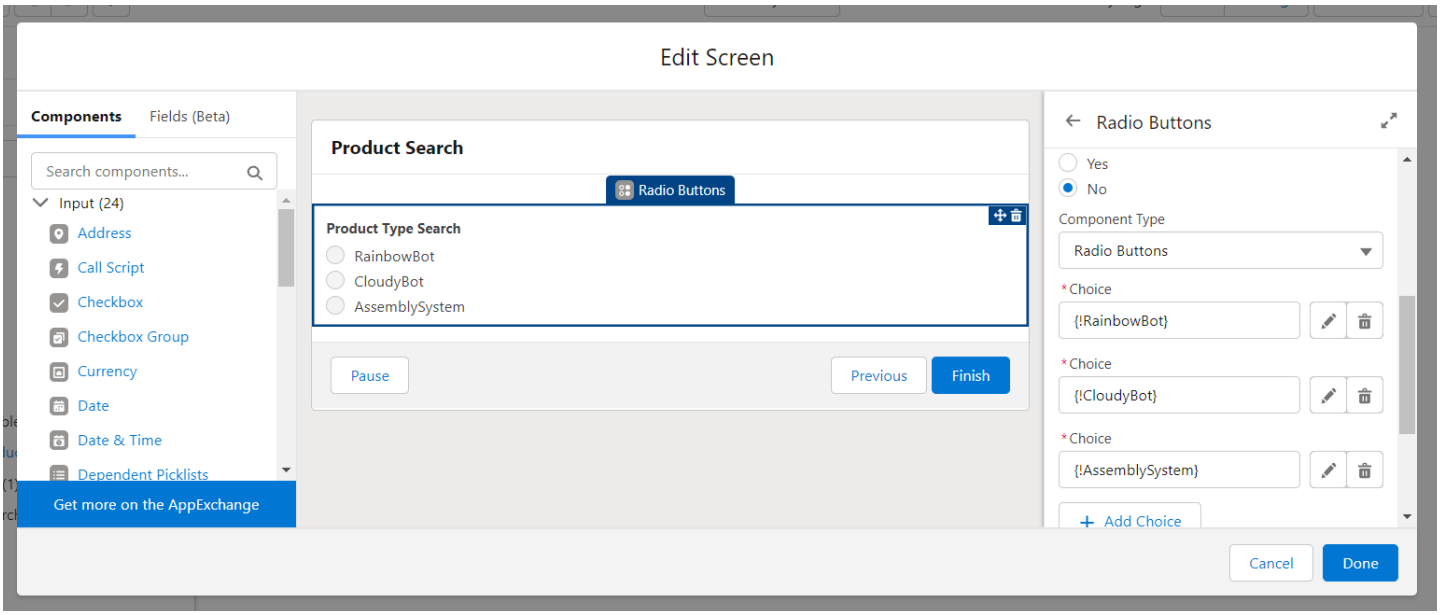
The opportunity owner’s manager has to approve all deals over $100,000. At this point, it should be in the Awaiting Approval stage.If the manager approves, the opportunity goes to the Closed/Won stage.If the manager doesn’t approve, the opportunity should go back to Negotiation/Review stage. Changes to the record are OK at that point.


Opportunity
Customer
TRUE
IMMEDIATE ACTIONS
Alert Finance
Add Action
EVALUATE THE NEXT CRITERIA
FALSE
Prospect Account
TRUE
IMMEDIATE ACTIONS
Alert Finance
Send Marketing Mat...
show fewer
Add Action
EVALUATE THE NEXT CRITERIA
FALSE
Negotiation or Review
TRUE
IMMEDIATE ACTIONS
Approval
Add Action
EVALUATE THE NEXT CRITERIA
FALSE
Closed Won
TRUE
IMMEDIATE ACTIONS
Record for Robot
Closed Won
show fewer
STOP
FALSE

Task 6 : Create Flow for Opportunities

We need some kind of product quick search thingy or widget on the page that lets reps filter products on the type of robot they want to buy and see a list right on the opportunity record.





Task 7 : Automate Setups

We need to make sure that any robot setup date that would fall on Saturday or Sunday is set to the following Monday instead.