

Project Document

NAME : Lakshay Garg

EMAIL: lakshay.511garg@gmail.com

Trailhead URL : <https://trailblazer.me/id/lgarg14>

(Apex Specialist Superbadge & Process Automation Specialist Superbadge)

Apex Specialist:

Automate record creation ->

2.1 MaintenanceRequestHelper

```
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;
    }
}
}

```

2.2 MaintenanceRequest

```

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

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

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

    }

}

```

Synchronize Salesforce data with an external system ->

3.1 WarehouseCalloutService

```

public with sharing class WarehouseCalloutService implements Queueable {
    private static final String WAREHOUSE_URL = 'https://th-superbadge-
apex.herokuapp.com/equipment';

```

//class that makes a REST callout to an external warehouse system to get a list of equipment that needs to be updated.

//The callout's JSON response returns the equipment records that you upsert in Salesforce.

```

    @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());

        //class maps the following fields: replacement part (always true), cost, current inventory,
        lifespan, maintenance cycle, and warehouse SKU
        //warehouse SKU will be external ID for identifying which equipment records to update
        within Salesforce
        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 one");
        }
    }
}

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

```

After saving the code open execute anonymous window (CTRL+E) and run this method ,

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

Schedule synchronization ->

4.1 WarehouseSyncShedule

```

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

```

```
}
```

Test automation logic ->

5.1 MaintenanceRequestHelperTest

@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);
}
}

```

5.2MaintenanceRequestHelper

```

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;
}
}
}
}

```

5.3 MaintenanceRequest

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

Test callout logic ->

6.1 WarehouseCalloutService

```
public with sharing class WarehouseCalloutService {
```

```
    private static final String WAREHOUSE_URL = 'https://th-superbadge-  
    apex.herokuapp.com/equipment';
```

```
    //@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 one');  
            System.debug(warehouseEq);
```

```

    }
}
}
}

```

6.2 WarehouseCalloutServiceTest

@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]);
    }
}

```

6.3 WarehouseCalloutServiceMock

@isTest

```

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

```

```

        System.assertEquals('https://th-superbadge-apex.herokuapp.com/equipment',
request.getEndpoint());
        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,"quantity":5,"name":"Ge
nerator 1000 kW","maintenanceperiod":365,"lifespan":120,"cost":5000,"sku":"100003"}');
        response.setStatusCode(200);
        return response;
    }
}

```

Test scheduling logic ->

7.1 WarehouseSyncSchedule

```
global class WarehouseSyncSchedule implements Schedulable {  
    global void execute(SchedulableContext ctx) {  
  
        WarehouseCalloutService.runWarehouseEquipmentSync();  
    }  
}
```

7.2 WarehouseSyncScheduleTest

```
@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();  
        //Contains schedule information for a scheduled job. CronTrigger is similar to a cron job on  
UNIX systems.  
        // This object is available in API version 17.0 and later.  
        CronTrigger a=[SELECT Id FROM CronTrigger where NextFireTime > today];  
        System.assertEquals(jobID, a.Id,'Schedule ');  
  
    }  
}
```

Process Automation Specialist:

Automate Leads ->

Validation rule on Lead

Searched for Validation rule and created a new under Leads

Error Condition Formula:

OR(AND(LEN(State) > 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", State)) , NOT(OR(Country ="US",Country ="USA",Country ="United States",
ISBLANK(Country))))Copy

Created two Queues:

Searched in quick box and select lead as object and create the below queues.

Queue Name: Rainbow Sales ; **AND** Assembly System Sales

Assignment Rule:

Search from quick box and create a new.

Rule Entry Edit
Trailhead

Enter the rule entry
Save
Save & New
Cancel

Step 1: Set the order in which this rule entry will be processed

Sort Order
2

Step 2: Select the criteria for this rule entry

Run this rule if the criteria are met :

Field	Operator	Value	
Lead: Lead Source	not equal to	Web	AND
--None--	--None--		AND
--None--	--None--		AND
--None--	--None--		AND
--None--	--None--		

Add Filter Logic...

Step 3: Select the user or queue to assign the Lead to

Queue
Assembly System Sales
Email Template

☐ Do Not Reassign Owner

Save
Save & New
Cancel

Automate Accounts ->

Created 4 Roll Up Summary fields as below:

Field 1: Label: **Number of deals**

Summary Type: **COUNT**

Summarized Object: **Opportunity**

Filter Criteria: **None**

Field 2: Label: **Number of won deals**

Summary Type: **COUNT**

Summarized Object: **Opportunity**

Filter Criteria: **Stage EQUALS Closed Won**

Field 3: Label: **Last won deal date**

Summary Type: **MAX**

Field to Aggregate: **Opportunity: Close Date**

Summarized Object: **Opportunity**

Filter Criteria: **Stage EQUALS Closed Won**

Field 4: Label: **Amount of won deals**

Summary Type: **SUM**

Field to Aggregate: **Opportunity: Amount**

Summarized Object: **Opportunity**

Filter Criteria: **Stage EQUALS Closed Won**

And 2 Formula Fields with below:

Field 5:Label: **Deal win percent**

Return Type: **Percent**

Decimal Places: 2

Formula: **(Number_of_won_deals__c / Number_of_deals__c)**

Field 6:Label: **Call for Service**

Return Type: **Text**

Formula: **IF(DATE(YEAR>Last_won_deal_date__c)+2 ,
MONTH>Last_won_deal_date__c),DAY>Last_won_deal_date__c) <=
TODAY(), "Yes", "No")**

Create 2 validation rules as below

Validation Rule 1 : Rule Name : US_Address (*Anything*)

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))))Copy

Error Message : You can not 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.

Error Location : Top Of Page

VALIDATION RULE 2 : Rule Name : Name Change

Error Condition Formula :

ISCHANGED(Name) && (OR(ISPICKVAL(Type , 'Customer - Direct') , ISPICKVAL(Type , 'Customer - Channel'))) Copy

Error Message : You can't change the Account name for "Customer – Direct" or "Customer – Channel"

Error Location : Account Name

Create Robot Setup Object ->

Created a custom object **Robot Setup** with a Master-Detail relationship to the **opportunity** include Autonumber the record name, starting with 0 using name format: ROBOT SETUP-{0000}.
Use the following field names.

Date, Date__c : Date type

Notes, Notes__c : Text type

Day of the Week, Day_of_the_Week__c : Number

Create Sales Process and Validate Opportunities ->

Started by adding a field to Opportunity

Approval: Checkbox type

Ideally, the sales reps shouldn't be able to check that box and only system administrators like and sales managers should be able to check it. Though it doesn't throw an error for that condition.

Also, Click on the Opportunity field **STAGE** and add a picklist value as "**Awaiting Approval**"

Next, **created a sales process** under opportunities by searching the sales process in the Search box. Add the desired fields as below

Opportunity Stages

Sales Process: RB Robotics Sales Process

Namespace Prefix:
 Description:

Available Values		Selected Values
Needs Analysis (Open, 20%, Pipeline)	Add ▶	Prospecting (Open, 10%, Pipeline)
Value Proposition (Open, 50%, Pipeline)		Qualification (Open, 10%, Pipeline)
Id. Decision Makers (Open, 60%, Pipeline)	Remove ◀	Proposal/Price Quote (Open, 75%, Pipeline)
Perception Analysis (Open, 70%, Pipeline)		Negotiation/Review (Open, 90%, Pipeline)
		Closed Won (Closed/Won, 100%, Closed)
		Closed Lost (Closed/Lost, 0%, Omitted)
		Awaiting Approval (Open, 95%, Best Case)

Save Cancel

Next add the **Opportunity Validation Rule** with error formula as below
 IF([Amount > 100000 && Approved__c <> True && ISPICKVAL(StageName,'Closed Won')
),True,False)

Automate Opportunities ->

Created Three Email Templates:

Finance: Account Creation,

SALES: Opportunity Needs Approval,

Sales: Opportunity Approval Status

Create related Email Alert from search box for the templates above.

Created an approval process:

Search for the approval process and select an **opportunity** object.

Process Definition Detail [Edit](#) [Clone](#) [Deactivate](#)

Process Name: prospect Active ☒

Unique Name: prospect Next Automated Approver Determined By

Description

Entry Criteria: (Opportunity: Stage EQUALS Negotiation/Review) AND (Opportunity: Amount GREATER THAN 100000)

Record Editability: Administrator **ONLY** Allow Submitters to Resubmit Approval Requests ☐

Approval Assignment Email Template: [SALES: Opportunity Needs Approval](#)

Initial Submitter: Opportunity Owner

Created By: [Nushi Davoud](#) 2020-06-04, 9:42 a.m. Modified By: [Nushi Davoud](#) 2020-06-05, 3:36 a.m.

Initial Submission Actions [Add Existing](#) [Add New](#)

Action	Type	Description
Edit Remove	Record Lock	Lock the record from being edited
Edit Remove	Field Update	approval update

Approval Steps [Add Existing](#) [Add New](#)

Action	Step Number	Name	Description	Criteria	Assigned Approver	Reject Behavior
Show Actions Edit	1	Step 1			User Nushi Davoud	Final Rejection

Final Approval Actions [Add Existing](#) [Add New](#)

Action	Type	Description
Edit Remove	Record Lock	Lock the record from being edited
Edit Remove	Field Update	user close

Final Rejection Actions [Add Existing](#) [Add New](#)

Action	Type	Description
Edit Remove	Record Lock	Unlock the record for editing
Edit Remove	Field Update	close

Recall Actions [Add Existing](#) [Add New](#)

Action	Type	Description
Edit Remove	Record Lock	Unlock the record for editing

Criteria :

(Opportunity: Stage EQUALS Negotiation/Review) AND (Opportunity: Amount GREATER THAN 100000)

SALES: Opportunity Needs Approval——>Template. Make sure to populate your manager as **Nushi Davoud** in **Manage Users**.

Create a process with the process builder

Opportunity object with option created and updated.

Node 1 Criteria.: Opportunity.Account Type = customer and Opportunity.account id not equal to null

Node 2 Criteria.: Opportunity.Account Type = Prospect, Opportunity stage = prospecting and Opportunity.account id not equal to null

Node 3 Criteria.: Opportunity Stage = Negotiation/Review and Opportunity Amount > 100,000

Node 4 Criteria.: Opportunity Stage = Closed Won



Action for **Node 1 Email Alert** to mail notifies account creation: Finance: Account Creation.

Email Alerts

Action Name * ⓘ

Email Alert *

Action for Node 2 :

Email Alert to mail notifies account creation : Finance: Account Creation.
Create a Record: Task with any name but mandatory subject line 'Send Marketing Materials'.

Make sure the string has no full stop or comma to it.

Assigned to the Account owner

The screenshot shows a Salesforce Flow Builder interface. On the left, a flow diagram has a decision diamond labeled 'prospect acct'. The 'TRUE' path leads to an 'IMMEDIATE ACTIONS' block containing 'create task' and 'alert'. The 'FALSE' path leads to another decision diamond labeled 'Nego'. Below this, the 'TRUE' path leads to another 'IMMEDIATE ACTIONS' block. On the right, the 'Create a Record' dialog is open for the 'Task' object. The 'Set Field Values' section is populated as follows:

Field *	Type *	Value *
Due Date Only	Formula	TODAY() + 7
Assigned To ID	Field Reference	[Opportunity].Account...
Priority	Picklist	High
Status	Picklist	In Progress
Subject	String	Send Marketing Materials
Related To ID	Field Reference	[Opportunity].Id

Action for Node 3: Approvals

Choose the one we created for the opportunity [here](#). And it takes care of the process thereby.

The screenshot shows a Salesforce Flow Builder interface. On the left, a flow diagram has a decision diamond labeled 'Nego'. The 'TRUE' path leads to an 'IMMEDIATE ACTIONS' block containing 'approval'. The 'FALSE' path leads to another decision diamond labeled 'closed deal'. Below this, the 'TRUE' path leads to an 'IMMEDIATE ACTIONS' block containing 'set robo' and 'STOP'. On the right, the 'Submit for Approval' dialog is open. The configuration is as follows:

- Object: Opportunity
- Approval Process: Specific approval process (selected), prospect - prospect
- Skip the entry criteria for this process?: ☐ Yes
- Submitter: Current User
- Submission Comments: (empty)

Action for Node 4: Record for Robot Setup

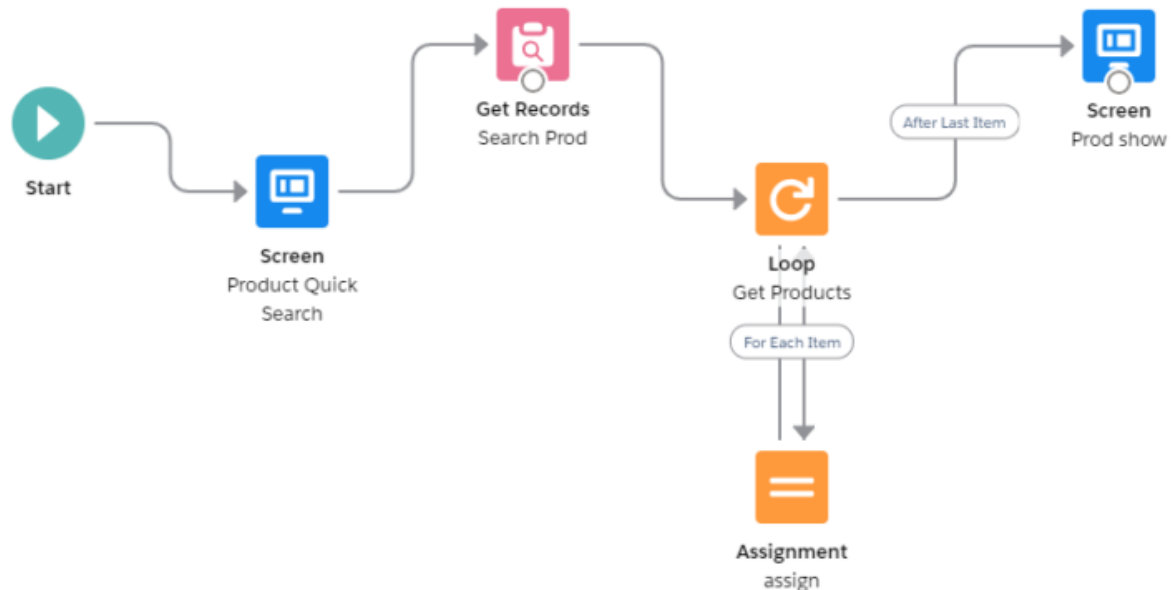
Set fields as below and Date formula being (closed date +180)

The screenshot shows a Salesforce Flow Builder interface. On the left, a flow diagram has a decision diamond labeled 'closed deal'. The 'TRUE' path leads to an 'IMMEDIATE ACTIONS' block containing 'set robo' and 'STOP'. The 'FALSE' path leads to another decision diamond labeled '+ Add Criteria'. Below this, the 'TRUE' path leads to an 'IMMEDIATE ACTIONS' block. On the right, the 'Create a Record' dialog is open for the 'Robot Setup' object. The configuration is as follows:

- Action Name: set robo
- Record Type: Robot Setup
- Set Field Values:

Field *	Type *	Value *
Opportunity	Field Reference	[Opportunity].Id
Date	Formula	[Opportunity].CloseDate + ...

Create Flow for Opportunities ->



Automate Setups ->

Search for the field “Day of the Week” on robot object and change the field type from Number to formula field of return type: text and use the below formula.

If you don't find the formula field in the edit option of the field, you can delete and recreate the field with the same name as well.

Formula being :

Case (WEEKDAY(Date__c),

1,"Sunday",

2,"Monday",

3,"Tuesday",

4,"Wednesday",

5,"Thursday",

6,"Friday",

7,"Saturday",

Text(WEEKDAY(Date__c)))Copy

Go to the Process we created in step 5. Clone this Process. Go to action on the last node where we set up robo record. Change formula of date field

from **[Opportunity].CloseDate + 180..to..** below formula.

The image shows two parts of a Salesforce interface. On the left is a flow diagram with two decision diamonds. The first diamond is labeled 'closed deal' and has a 'TRUE' path leading to an 'IMMEDIATE ACTIONS' box containing 'set robo' and 'show more', which then leads to a 'STOP' node. The 'FALSE' path leads to a second decision diamond labeled '+ Add Criteria'. This second diamond has a 'TRUE' path leading to another 'IMMEDIATE ACTIONS' box with an '+ Add Action' button, which then leads to a 'STOP' node. The 'FALSE' path of the second diamond is unlabeled. On the right is the 'Create a Record' formula editor. It shows a table with columns 'Field', 'Type', and 'Value'. The first row has 'Opportunity' in the Field column, 'Field Reference' in the Type column, and '[Opportunity].Id' in the Value column. The second row has 'Date' in the Field column, 'Formula' in the Type column, and a complex CASE formula in the Value column. The formula is: `CASE(MOD([Opportunity].CloseDate + 180 - DATE(1900, 1, 7), 7), 0, [Opportunity].CloseDate + 181, 6, [Opportunity].CloseDate + 182, [Opportunity].CloseDate + 180)`. Below the table are buttons for 'Use this Formula' and 'Cancel'.

CASE(MOD([Opportunity].CloseDate + 180 - DATE(1900, 1, 7), 7), 0, [Opportunity].CloseDate + 181, 6, [Opportunity].CloseDate + 182, [Opportunity].CloseDate + 180)