Apex Specialist

2)Automate record creation

MaintenanceRequest.cls

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

MaintenanceRequestHelper.cls

```
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);
                }
            }
        }

        //When an existing maintenance request of type Repair or Routine Maintenance is closed,
        //create a new maintenance request for a future routine checkup.
        if (!validIds.isEmpty()){
            Map<Id,Case> closedCases = 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>();

      //calculate the maintenance request due dates by using the maintenance cycle defined on the related equipment records.
      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'));
      }

      List<Case> newCases = new List<Case>();
      for(Case cc : closedCases.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 multiple pieces of equipment are used in the maintenance request,
                //define the due date by applying the shortest maintenance cycle to today’s date.
                //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> clonedList = new List<Equipment_Maintenance_Item__c>();
            for (Case nc : newCases){
                for (Equipment_Maintenance_Item__c clonedListItem : closedCases.get(nc.ParentId).Equipment_Maintenance_Items__r){
                    Equipment_Maintenance_Item__c item = clonedListItem.clone();
                    item.Maintenance_Request__c = nc.Id;
                    clonedList.add(item);
                }
            }
            insert clonedList;
        }
    }
}
```

3)Synchronize Salesforce data with an external system

WarehouseCalloutService

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

    //Write a 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(){
        System.debug('go into runWarehouseEquipmentSync');
        Http http = new Http();
        HttpRequest request = new HttpRequest();

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

        List<Product2> product2List = new List<Product2>();
        System.debug(response.getStatusCode());
        if (response.getStatusCode() == 200){
            List<Object> jsonResponse = (List<Object>)JSON.deserializeUntyped(response.getBody());
            System.debug(response.getBody());

            //class maps the following fields:
            //warehouse SKU will be external ID for identifying
```

```
which equipment records to update within Salesforce
            for (Object jR : jsonResponse){
                Map<String,Object> mapJson = (Map<String,Object>)jR;
                Product2 product2 = new Product2();
                //replacement part (always true),
                product2.Replacement_Part__c = (Boolean) mapJson.get('replacement');
                //cost
                product2.Cost__c = (Integer) mapJson.get('cost');
                //current inventory
                product2.Current_Inventory__c = (Double) mapJson.get('quantity');
                //lifespan
                product2.Lifespan_Months__c = (Integer) mapJson.get('lifespan');
                //maintenance cycle
                product2.Maintenance_Cycle__c = (Integer) mapJson.get('maintenanceperiod');
                //warehouse SKU
                product2.Warehouse_SKU__c = (String) mapJson.get('sku');

                product2.Name = (String) mapJson.get('name');
                product2.ProductCode = (String) mapJson.get('_id');
                product2List.add(product2);
            }

            if (product2List.size() > 0){
                upsert product2List;
                System.debug('Your equipment was synced with the

            }
        }
    }

    public static void execute (QueueableContext context){
        System.debug('start runWarehouseEquipmentSync');
        runWarehouseEquipmentSync();
        System.debug('end runWarehouseEquipmentSync');
    }

}
```

4) Schedule synchronization

class;`WarehouseCalloutService`

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

    //Write a 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(){
        System.debug('go into runWarehouseEquipmentSync');
        Http http = new Http();
        HttpRequest request = new HttpRequest();

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

        List<Product2> product2List = new List<Product2>();
        System.debug(response.getStatusCode());
        if (response.getStatusCode() == 200){
            List<Object> jsonResponse = (List<Object>)JSON.deserializeUntyped(response.getBody());
            System.debug(response.getBody());

            //class maps the following fields:
```

```
            //warehouse SKU will be external ID for identifying which equipment records to update within Salesforce
            for (Object jR : jsonResponse){
                Map<String,Object> mapJson = (Map<String,Object>)jR;
                Product2 product2 = new Product2();
                //replacement part (always true),
                product2.Replacement_Part__c = (Boolean) mapJson.get('replacement');
                //cost
                product2.Cost__c = (Integer) mapJson.get('cost');
                //current inventory
                product2.Current_Inventory__c = (Double) mapJson.get('quantity');
                //lifespan
                product2.Lifespan_Months__c = (Integer) mapJson.get('lifespan');
                //maintenance cycle
                product2.Maintenance_Cycle__c = (Integer) mapJson.get('maintenanceperiod');
                //warehouse SKU
                product2.Warehouse_SKU__c = (String) mapJson.get('sku');

                product2.Name = (String) mapJson.get('name');
                product2.ProductCode = (String) mapJson.get('_id');
                product2List.add(product2);
            }

            if (product2List.size() > 0){
                upsert product2List;
                System.debug('Your equipment was synced
            }
        }
    }

    public static void execute (QueueableContext context){
        System.debug('start runWarehouseEquipmentSync');
        runWarehouseEquipmentSync();
        System.debug('end runWarehouseEquipmentSync');
    }

}
```

Schedule synchronization.cls

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

5) Test automation logic

MaintenanceRequest.cls

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

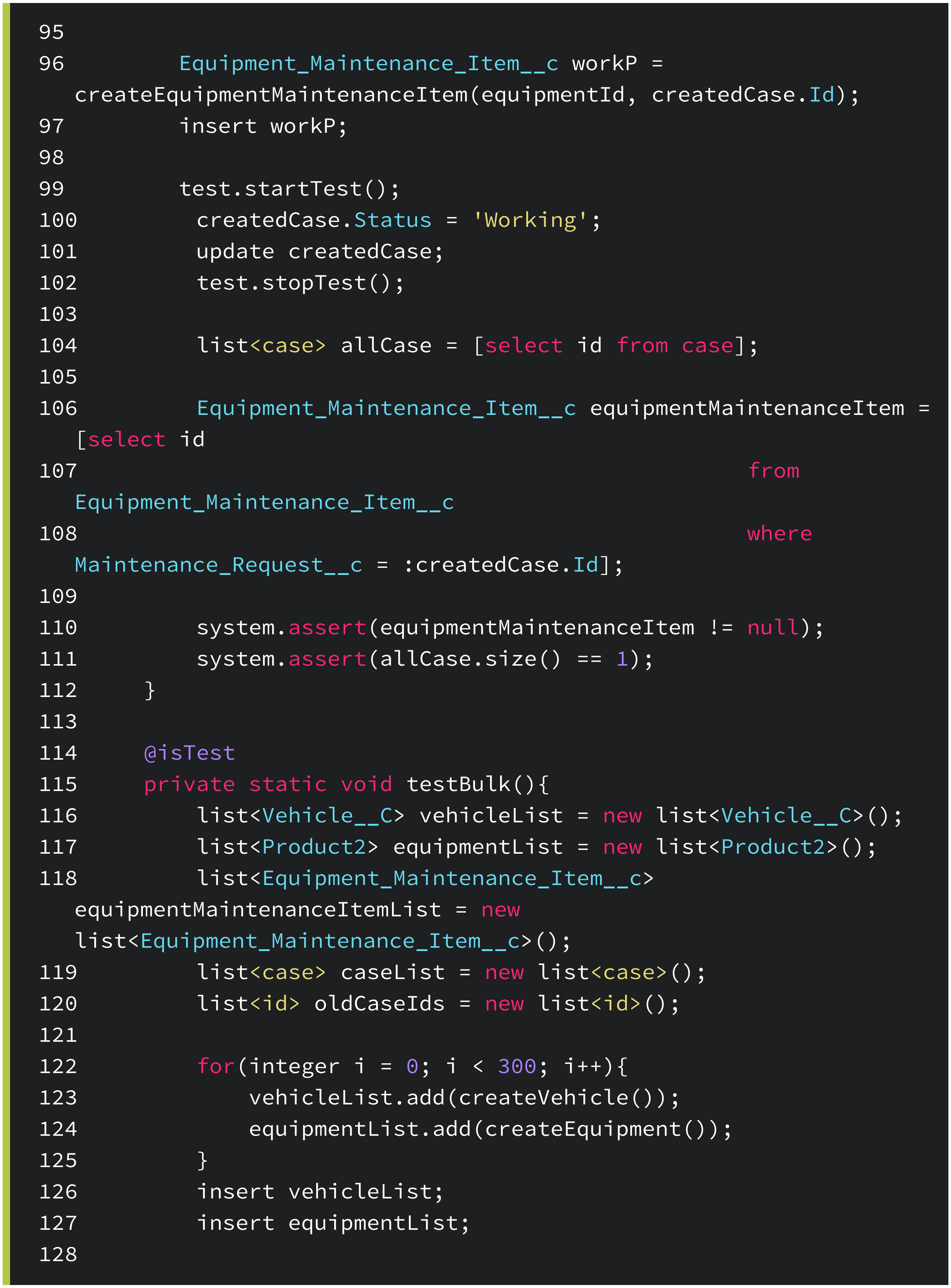
MaintenanceRequestHelper.cls

```
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

          validIds.add(c.Id);
        }
      }
    }

    //When an existing maintenance request of type Repair or Routine Maintenance is closed,
    //create a new maintenance request for a future routine checkup.
    if (!validIds.isEmpty()){
      Map<Id,Case> closedCases = 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>();

      //calculate the maintenance request due dates by using the maintenance cycle defined on the related equipment records.
      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'));
            }

            List<Case> newCases = new List<Case>();
            for(Case cc : closedCases.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 multiple pieces of equipment are used in the maintenance request,
                //define the due date by applying the shortest maintenance cycle to today’s date.
                //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> clonedList = new List<Equipment_Maintenance_Item__c>();
            for (Case nc : newCases){
                for (Equipment_Maintenance_Item__c clonedListItem : closedCases.get(nc.ParentId).Equipment_Maintenance_Items__r){
                    Equipment_Maintenance_Item__c item = clonedListItem.clone();
                    item.Maintenance_Request__c = nc.Id;
                    clonedList.add(item);
                }
            }
            insert clonedList;
        }
    }
}
```

MaintenanceRequestHelperTest

```
@isTest
public with sharing class MaintenanceRequestHelperTest {

    // createVehicle
    private static Vehicle__c createVehicle(){
        Vehicle__c vehicle = new Vehicle__C(name = 'Testing
        return vehicle;
    }

    // createEquipment
    private static Product2 createEquipment(){
        product2 equipment = new product2(name = 'Testing
                                          lifespan_months__c = 10,
                                          maintenance_cycle__c = 10,
                                          replacement_part__c = true);
        return equipment;
    }

    // createMaintenanceRequest
    private static Case createMaintenanceRequest(id vehicleId, id equipmentId){
        case cse = new case(Type='Repair',
                            Status='New',
                            Origin='Web',
                            Subject='Testing subject',
                            Equipment__c=equipmentId,
                            Vehicle__c=vehicleId);
        return cse;
    }

    // createEquipmentMaintenanceItem
    private static Equipment_Maintenance_Item__c createEquipmentMaintenanceItem(id equipmentId,id requestId){
        Equipment_Maintenance_Item__c equipmentMaintenanceItem = new Equipment_Maintenance_Item__c(
            Equipment__c = equipmentId,
            Maintenance_Request__c = requestId);
        return equipmentMaintenanceItem;
    }

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

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

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

        Equipment_Maintenance_Item__c equipmentMaintenanceItem = createEquipmentMaintenanceItem(equipmentId,createdCase.id);
        insert equipmentMaintenanceItem;

        test.startTest();
        createdCase.status = 'Closed';
        update createdCase;
        test.stopTest();

        Case newCase = [Select id,
                        subject,
                        type,
                        Equipment__c,
                        Date_Reported__c,
                        Vehicle__c,
                        Date_Due__c
                       from case
                       where status ='New'];

        Equipment_Maintenance_Item__c workPart = [select id
                                                  from Equipment_Maintenance_Item__c
                                                  where Maintenance_Request__c =:newCase.Id];
        list<case> allCase = [select id from case];
        system.assert(allCase.size() == 2);

        system.assert(newCase != null);
        system.assert(newCase.Subject != null);
        system.assertEquals(newCase.Type, 'Routine Maintenance');
        SYSTEM.assertEquals(newCase.Equipment__c, equipmentId);
        SYSTEM.assertEquals(newCase.Vehicle__c, vehicleId);
        SYSTEM.assertEquals(newCase.Date_Reported__c, system.today());
    }

    @isTest
    private static void testNegative(){
        Vehicle__C vehicle = createVehicle();
        insert vehicle;
        id vehicleId = vehicle.Id;

        product2 equipment = createEquipment();
        insert equipment;
        id equipmentId = equipment.Id;

        case createdCase = createMaintenanceRequest(vehicleId,equipmentId);
        insert createdCase;
```



```

        Equipment_Maintenance_Item__c workP = createEquipmentMaintenanceItem(equipmentId, createdCase.Id);
        insert workP;

        test.startTest();
        createdCase.Status = 'Working';
        update createdCase;
        test.stopTest();

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

        Equipment_Maintenance_Item__c equipmentMaintenanceItem = [select id
                                                      from Equipment_Maintenance_Item__c
                                                      where Maintenance_Request__c = :createdCase.Id];

        system.assert(equipmentMaintenanceItem != null);
        system.assert(allCase.size() == 1);
    }

    @isTest
    private static void testBulk(){
        list<Vehicle__C> vehicleList = new list<Vehicle__C>();
        list<Product2> equipmentList = new list<Product2>();
        list<Equipment_Maintenance_Item__c> equipmentMaintenanceItemList = new list<Equipment_Maintenance_Item__c>();
        list<case> caseList = new list<case>();
        list<id> oldCaseIds = new list<id>();

        for(integer i = 0; i < 300; i++){
            vehicleList.add(createVehicle());
            equipmentList.add(createEquipment());
        }
        insert vehicleList;
        insert equipmentList;

        for(integer i = 0; i < 300; i++){
            caseList.add(createMaintenanceRequest(vehicleList.get(i).id, equipmentList.get(i).id));
        }
        insert caseList;

        for(integer i = 0; i < 300; i++){
            equipmentMaintenanceItemList.add(createEquipmentMaintenanceItem(e
        }
        insert equipmentMaintenanceItemList;

        test.startTest();
        for(case cs : caseList){
            cs.Status = 'Closed';
            oldCaseIds.add(cs.Id);
        }
        update caseList;
        test.stopTest();

        list<case> newCase = [select id
                                  from case
                                  where status ='New'];



        list<Equipment_Maintenance_Item__c> workParts = [select id
                                                         from Equipment_Maintenance_Item__c
                                                         where Maintenance_Request__c in: oldCaseIds];

        system.assert(newCase.size() == 300);

        list<case> allCase = [select id from case];
        system.assert(allCase.size() == 600);
    }
```

```
}
```

6) TEST callout Logic

WarehouseCalloutService.cls

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

    //Write a 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(){
        System.debug('go into runWarehouseEquipmentSync');
        Http http = new Http();
        HttpRequest request = new HttpRequest();

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

        List<Product2> product2List = new List<Product2>();
        System.debug(response.getStatusCode());
        if (response.getStatusCode() == 200){
            List<Object> jsonResponse = (List<Object>)JSON.deserializeUntyped(response.getBody());
            System.debug(response.getBody());

            //class maps the following fields:
            //warehouse SKU will be external ID for identifying which equipment records to update within Salesforce
            for (Object jR : jsonResponse){
                Map<String,Object> mapJson = (Map<String,Object>)jR;
                Product2 product2 = new Product2();
                //replacement part (always true),
                product2.Replacement_Part__c = (Boolean) mapJson.get('replacement');
                //cost
                product2.Cost__c = (Integer) mapJson.get('cost');
                //current inventory
                product2.Current_Inventory__c = (Double) mapJson.get('quantity');
                //lifespan
                product2.Lifespan_Months__c = (Integer) mapJson.get('lifespan');
                //maintenance cycle
                product2.Maintenance_Cycle__c = (Integer) mapJson.get('maintenanceperiod');
                //warehouse SKU
                product2.Warehouse_SKU__c = (String) mapJson.get('sku');

                product2.Name = (String) mapJson.get('name');
                product2.ProductCode = (String) mapJson.get('_id');
                product2List.add(product2);
            }

            if (product2List.size() > 0){
                upsert product2List;
                System.debug('Your equipment was synced with the

            }
        }
    }

    public static void execute (QueueableContext context){
        System.debug('start runWarehouseEquipmentSync');
        runWarehouseEquipmentSync();
        System.debug('end runWarehouseEquipmentSync');
    }

}
```

WarehouseCalloutServiceMock.cls

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

        HttpResponse response = new HttpResponse();
        response.setHeader('Content-Type', 'application/json');

response.setBody('[{"_id":"55d66226726b611100aaf741","replacement









        response.setStatusCode(200);

        return response;
    }
}
```

WarehouseCalloutServiceTest.cls

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

        HttpResponse response = new HttpResponse();
        response.setHeader('Content-Type', 'application/json');

response.setBody('[{"_id":"55d66226726b611100aaf741","replacement
        response.setStatusCode(200);

        return response;
    }
}
```

7)

WarehouseSyncSchedule

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

WarehouseCalloutServiceMock.cls

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

        HttpResponse response = new HttpResponse();
        response.setHeader('Content-Type', 'application/json');
        response.setBody('[{"_id":"55d66226726b611100aaf741","replacement
        response.setStatusCode(200);

        return response;
    }
}
```

WarehouseSyncScheduleTest.cls

```
@isTest
public with sharing class WarehouseSyncScheduleTest {
    // implement scheduled code here
    //
    @isTest static void test() {
        String scheduleTime = '00 00 00 * * ? *';
        Test.startTest();
        Test.setMock(HttpCalloutMock.class, new WarehouseCalloutServiceMock());
        String jobId = System.schedule('Warehouse Time to ());
        CronTrigger c = [SELECT State FROM CronTrigger WHERE Id =: jobId];
        System.assertEquals('WAITING', String.valueOf(c.State), 'JobId does not match');

        Test.stopTest();
    }
}
```

Process Automation Specialist

Automate Leads

1)Install package ID 04t46000001Zch4
2)lead- validation rule -new-name- formula=OR(

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)),

LEN(State) <> 2,

NOT(OR(Country ="US",Country ="USA",Country ="United States", ISBLANK(Country)))

)

3)queuse -new-label - RainbowSales-select -object lead and add it-save
4)queuse -new-label -Assembly System Sales -add lead object and -save
5)home -search box-lead assignment rule-new-rule name- RainbowSale-active-save
edit-sort order=1-critirea -lead source -equals-web
6)home -search box-lead assignment rule-new-rule name- Assembly System Sales-active-save
edit-sort order=1-critirea -lead source -NOT equals-web

Automate Accounts

1)create four rollup summary in account object
Feild- Lable-number of deals
summary type-count
summarized object-opportunity
filter -none

Feild- Lable-number ofwon deals

summary type-count
summarized object-opportunity
filter -satge equals closed won

Feild- Lable-last won deal date
summary type-max
field to aggregate ;opportunity=close date
summarized object-opportunity
filter -satge equals closed won

Feild- Lable-amount of won deals
summary type-sum
field to aggregate ;opportunity=amount
summarized object-opportunity
filter -satge equals closed won

2)2 formula feild in account object

lable ;deal win percent
return type; percent
decimal place ; 2
formula; (Number_of_won_deals__c/Number_of_deals__c)

lable ;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_dea
```

4) validation rule
rule name =us address
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:M
A: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:M
A: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))))
```

```
rule name= name
formula = ISCHANGED( Name ) && ( OR( ISPICKVAL( Type ,'Customer - Direct')
,ISPICKVAL( Type ,'Customer - Channel') ))
```

Create Robot Setup Object

1) create robot setup object
with auto number
2) robot steup object craete a master detail relationship with opportaunity.
3)create 3 field in robot setup object

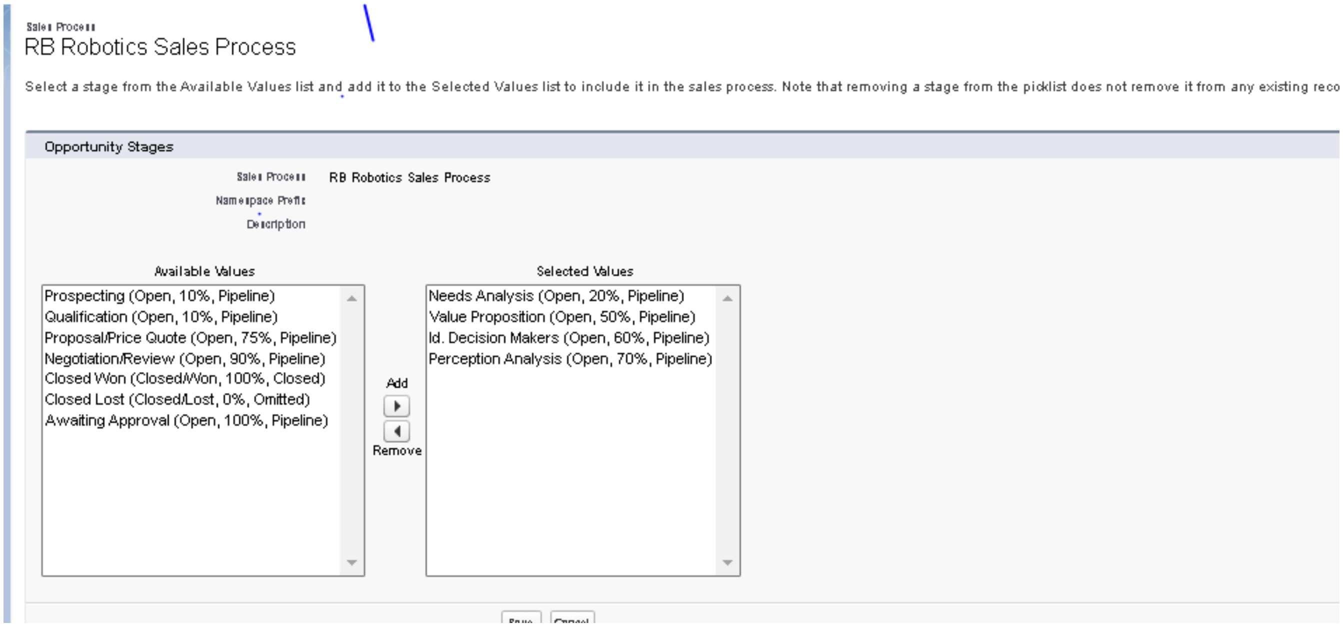
field name- date =datatype=date
field name -notes=datatype=text area

field name=day of week -data type- formula =

```
CASE(weekday (Date__c),
1,"Sunday",
2,"Monday",
3,"Tuesday",
4,"Wednesday",
5,"Thursday",
6,"Friday",
7,"Saturday",
Text(weekday (Date__c))
)
```

Create Sales Process and Validate Opportunities

1) opportaunity object - record type - new -create sales process-



Sales Process

RB Robotics Sales Process

Select a stage from the Available Values list and add it to the Selected Values list to include it in the sales process. Note that removing a stage from the picklist does not remove it from any existing reco

Opportunity Stages

Sales Process RB Robotics Sales Process

Namespace Prefix

Description

Available Values

Prospecting (Open, 10%, Pipeline)
Qualification (Open, 10%, Pipeline)
Proposal/Price Quote (Open, 75%, Pipeline)
Negotiation/Review (Open, 90%, Pipeline)
Closed Won (Closed/Won, 100%, Closed)
Closed Lost (Closed/Lost, 0%, Omitted)
Awaiting Approval (Open, 100%, Pipeline)

Selected Values

Needs Analysis (Open, 20%, Pipeline)
Value Proposition (Open, 50%, Pipeline)
Id. Decision Makers (Open, 60%, Pipeline)
Perception Analysis (Open, 70%, Pipeline)

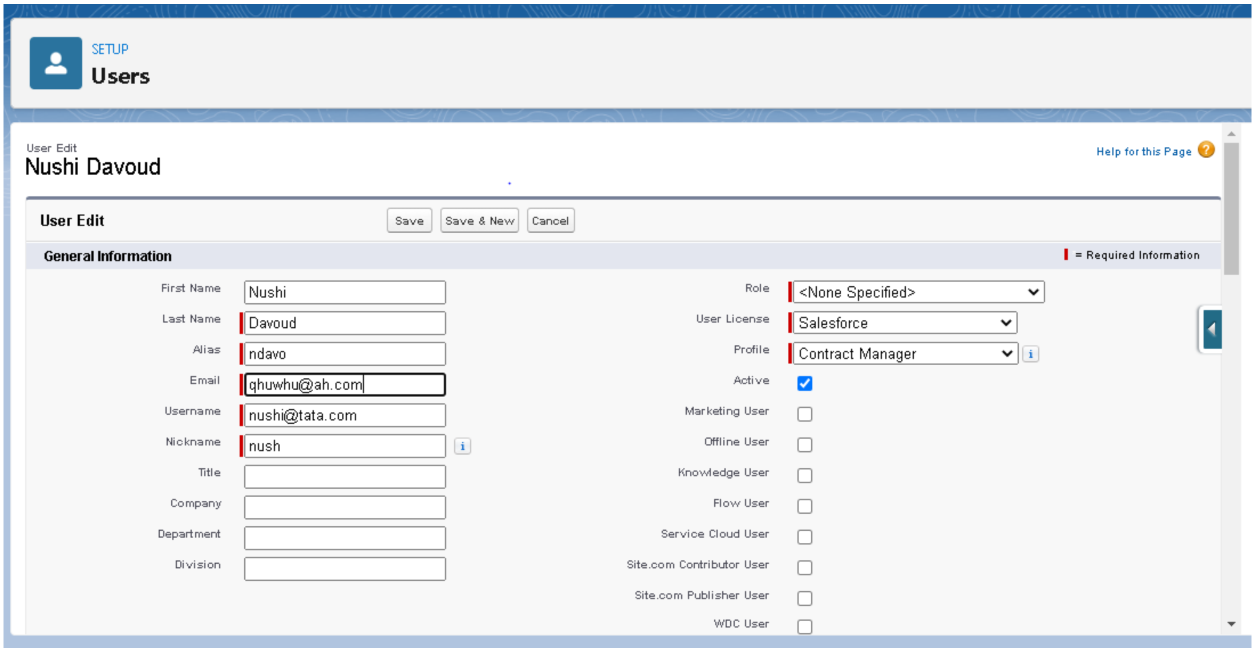
2) create new field in opportaunity -lable name = approverd -data type -checkbox-save

3)create vaildation rule in opportaunity object- rule name=Opportunity_Validation_Rule_1=

formula= IF((Amount > 100000 && Approved__c <> True && ISPICKVAL(StageName,'Closed Won')
),True,False)

Automate Opportunities

1) create user name Nushi Davoud



User Edit

Nushi Davoud

User Edit

General Information

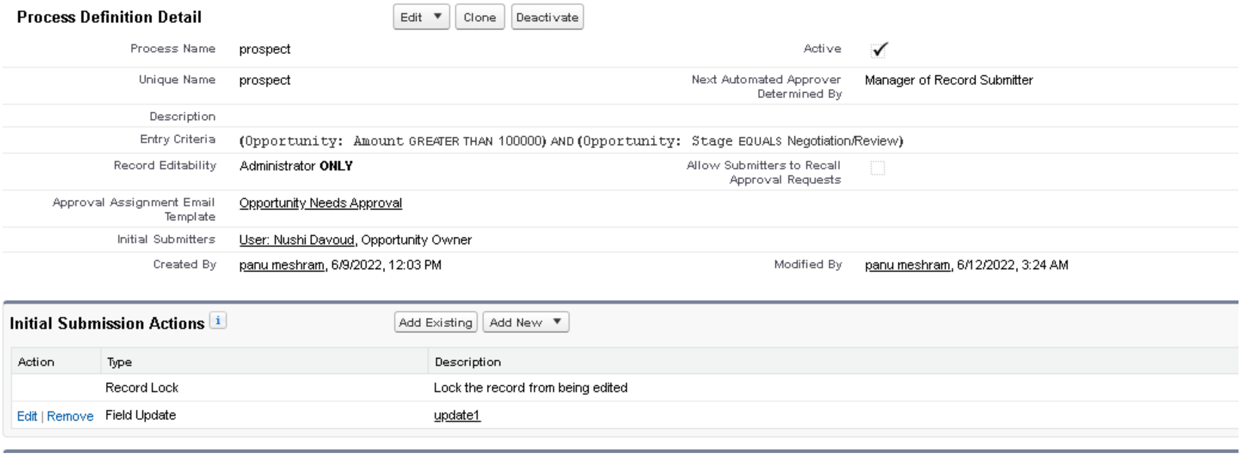
= Required Information

First Name	Nushi	Role	<None Specified>
Last Name	Davoud	User License	Salesforce
Alias	ndavo	Profile	Contract Manager
Email	qhuwhu@ah.com	Active	☑
Username	nushi@tata.com	Marketing User	□
Nickname	nush	Offline User	□
Title		Knowledge User	□
Company		Flow User	□
Department		Service Cloud User	□
Division		Site.com Contributor User	□
		Site.com Publisher User	□
		WDC User	□

2) create 3 email alert with name

fianance :account creation
sales ; opportunity needs approval
sales; opportunity approval status

3) create approval process
opportunity object

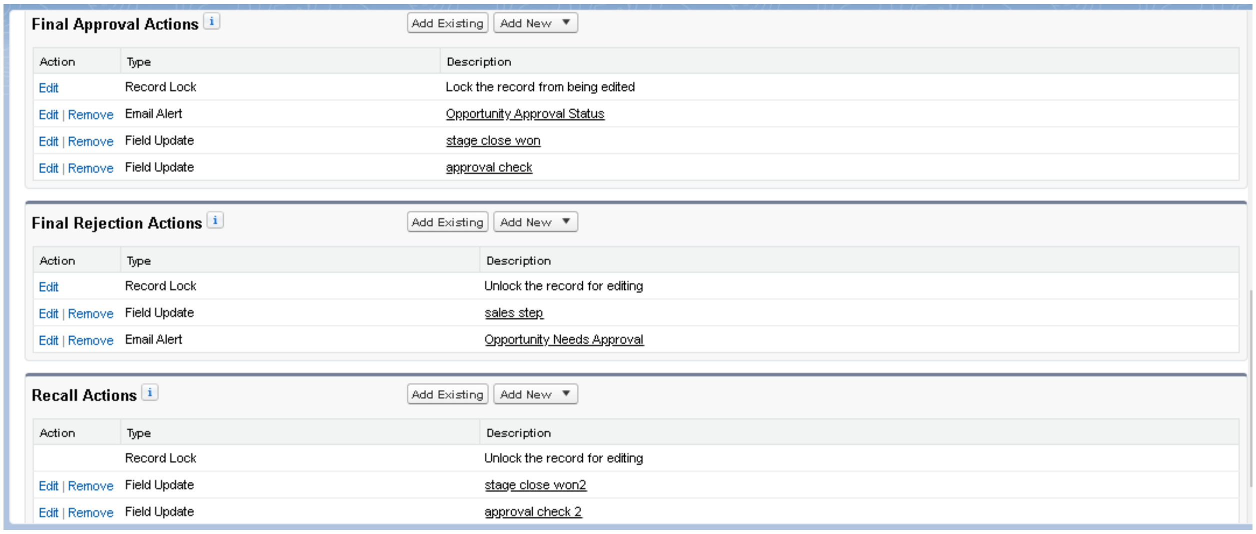


Process Definition Detail

Process Name	prospect	Active	✓
Unique Name	prospect	Next Automated Approver Determined By	Manager of Record Submitter
Description			
Entry Criteria	(Opportunity: Amount GREATER THAN 100000) AND (Opportunity: Stage EQUALS Negotiation/Review)		
Record Editability	Administrator **ONLY**	Allow Submitters to Recall Approval Requests	□
Approval Assignment Email Template	Opportunity Needs Approval		
Initial Submitters	User: Nushi Davoud, Opportunity Owner		
Created By	panu meshram, 6/9/2022, 12:03 PM	Modified By	panu meshram, 6/12/2022, 3:24 AM

Initial Submission Actions

Action	Type	Description
	Record Lock	Lock the record from being edited
Edit | Remove	Field Update	update1



Final Approval Actions

Action	Type	Description
Edit	Record Lock	Lock the record from being edited
Edit | Remove	Email Alert	Opportunity Approval Status
Edit | Remove	Field Update	stage close won
Edit | Remove	Field Update	approval check

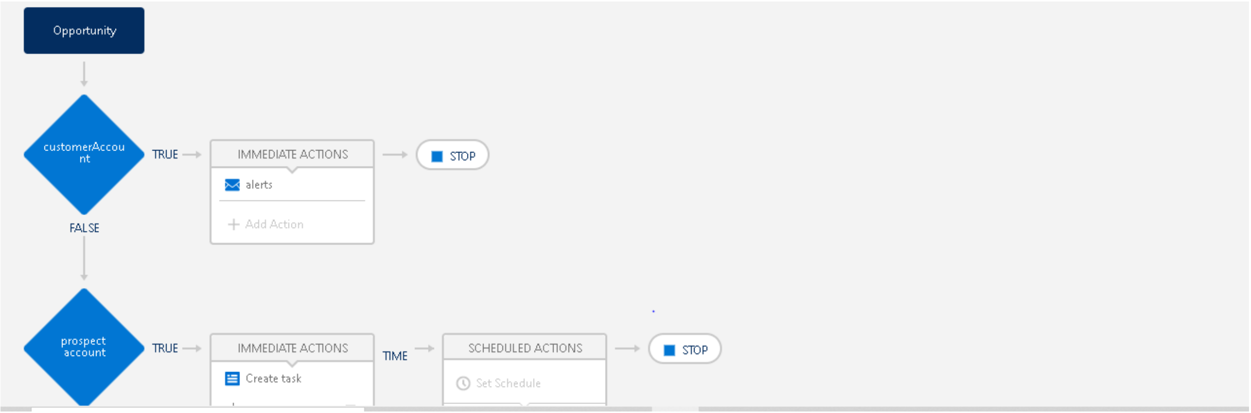
Final Rejection Actions

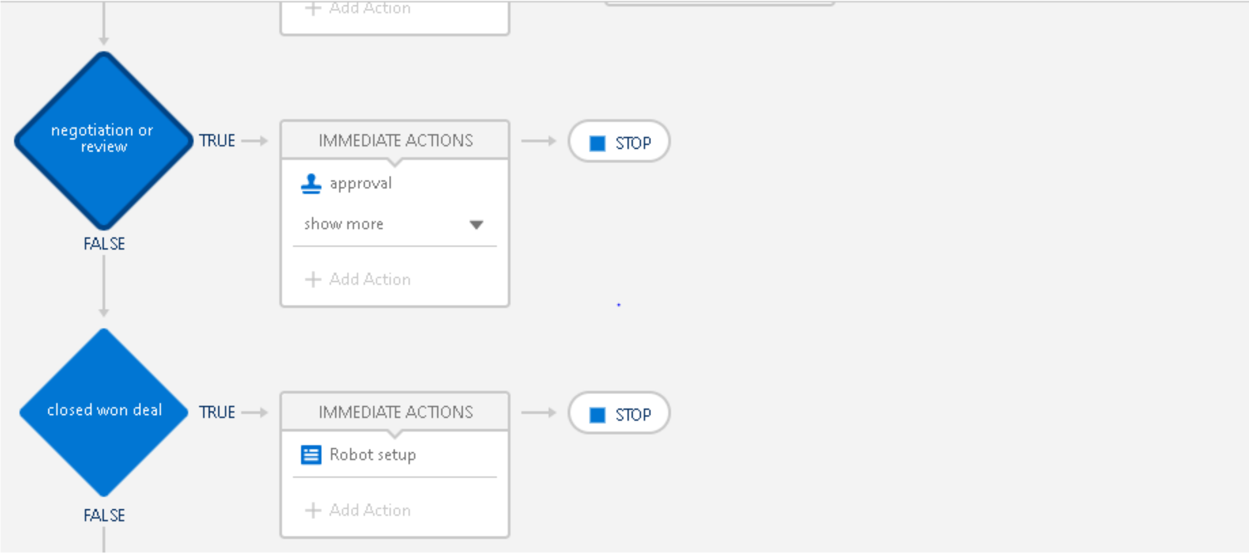
Action	Type	Description
Edit	Record Lock	Unlock the record for editing
Edit | Remove	Field Update	sales step
Edit | Remove	Email Alert	Opportunity Needs Approval

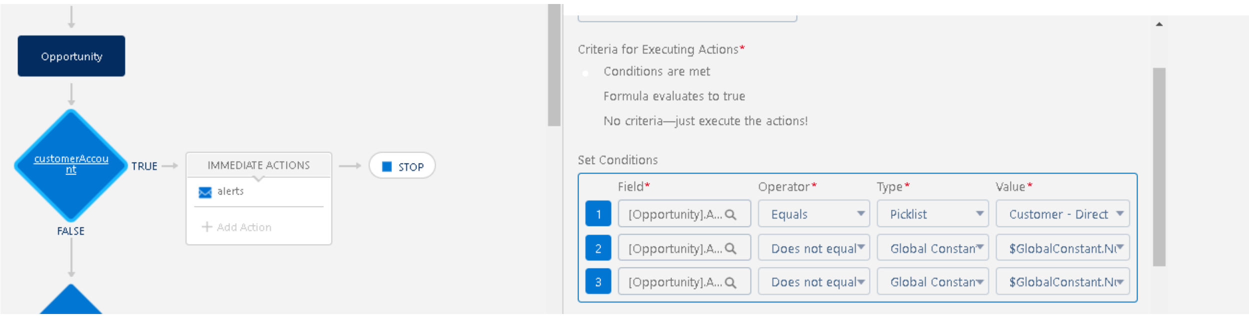
Recall Actions

Action	Type	Description
	Record Lock	Unlock the record for editing
Edit | Remove	Field Update	stage close won2
Edit | Remove	Field Update	approval check 2

4) proccess builder







Criteria for Executing Actions*

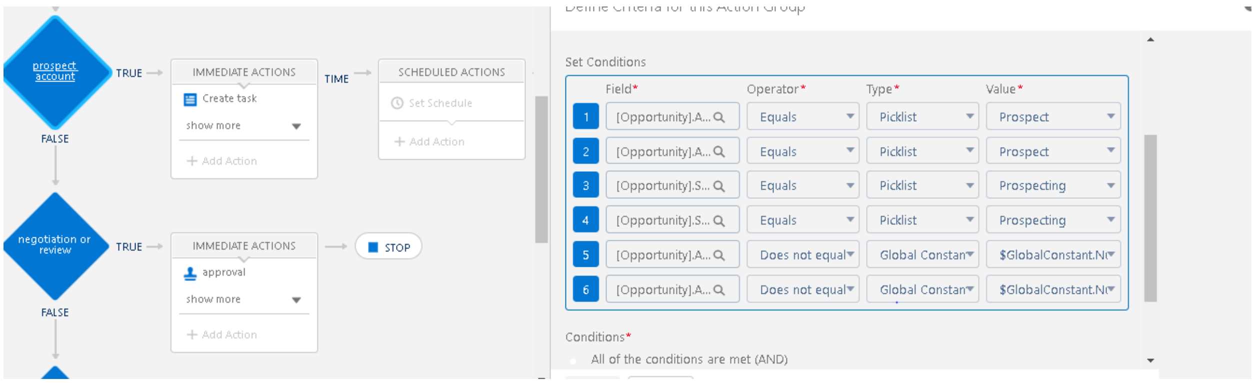
○ Conditions are met

Formula evaluates to true

No criteria—just execute the actions!

Set Conditions

	Field*	Operator*	Type*	Value*
1	[Opportunity].A...	Equals	Picklist	Customer - Direct
2	[Opportunity].A...	Does not equal	Global Constan	$GlobalConstant.Nι
3	[Opportunity].A...	Does not equal	Global Constan	$GlobalConstant.Nι



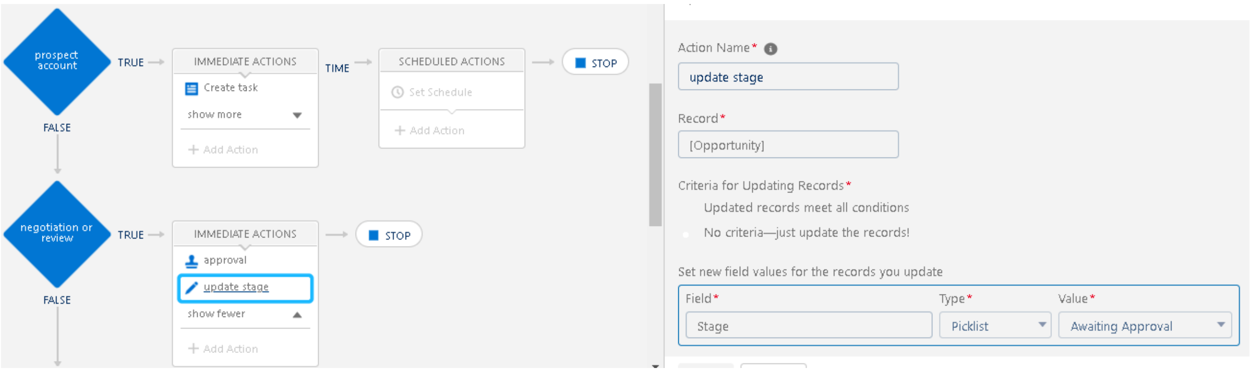
Define Criteria for this Action Group

Set Conditions

	Field*	Operator*	Type*	Value*
1	[Opportunity].A...	Equals	Picklist	Prospect
2	[Opportunity].A...	Equals	Picklist	Prospect
3	[Opportunity].S...	Equals	Picklist	Prospecting
4	[Opportunity].S...	Equals	Picklist	Prospecting
5	[Opportunity].A...	Does not equal	Global Constan	$GlobalConstant.Nι
6	[Opportunity].A...	Does not equal	Global Constan	$GlobalConstant.Nι

Conditions*

○ All of the conditions are met (AND)



Action Name* update stage

Record* [Opportunity]

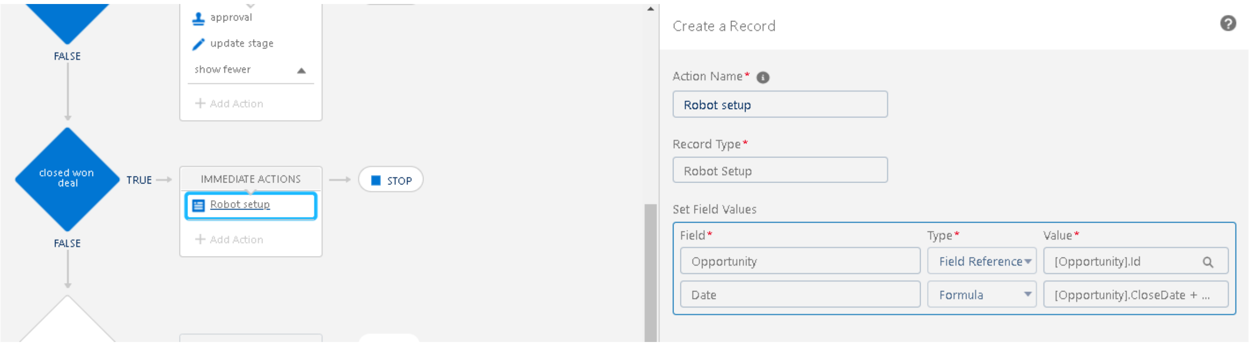
Criteria for Updating Records*

Updated records meet all conditions

No criteria—just update the records!

Set new field values for the records you update

Field*	Type*	Value*
Stage	Picklist	Awaiting Approval



Create a Record

Action Name* Robot setup

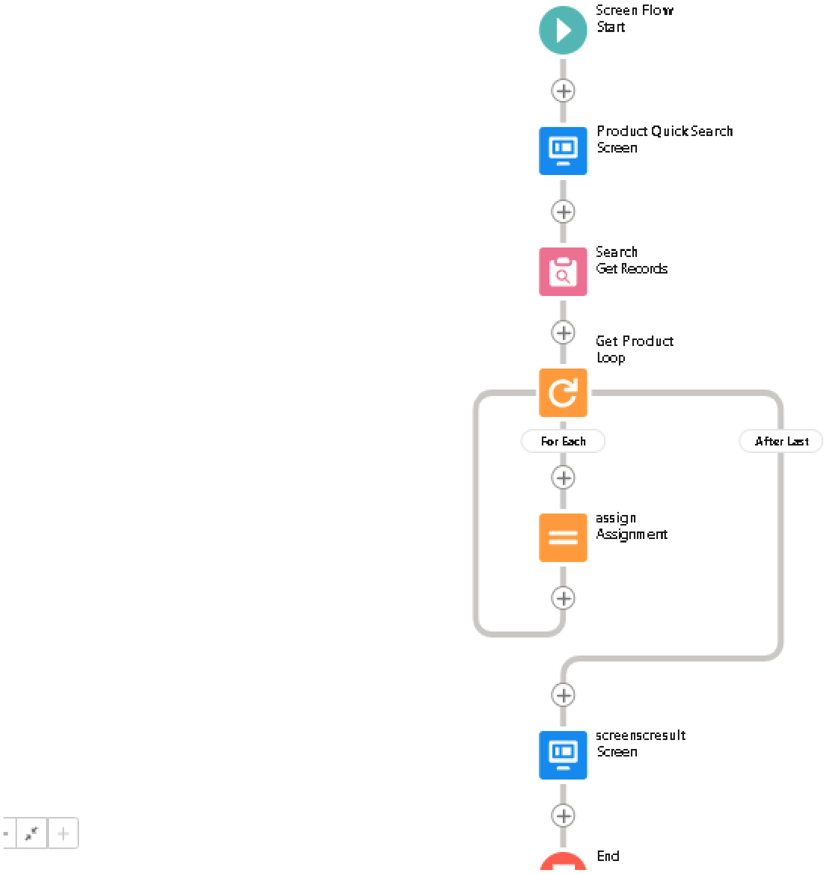
Record Type* Robot Setup

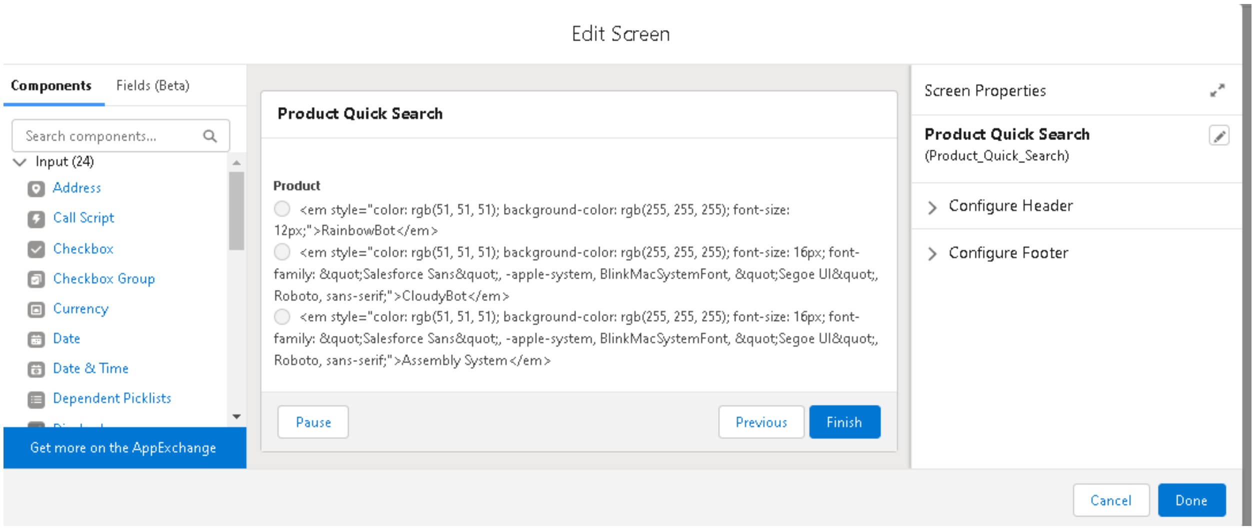
Set Field Values

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

Create Flow for Opportunities

1) create flow for opportunity




Edit Screen

Product Quick Search

Product

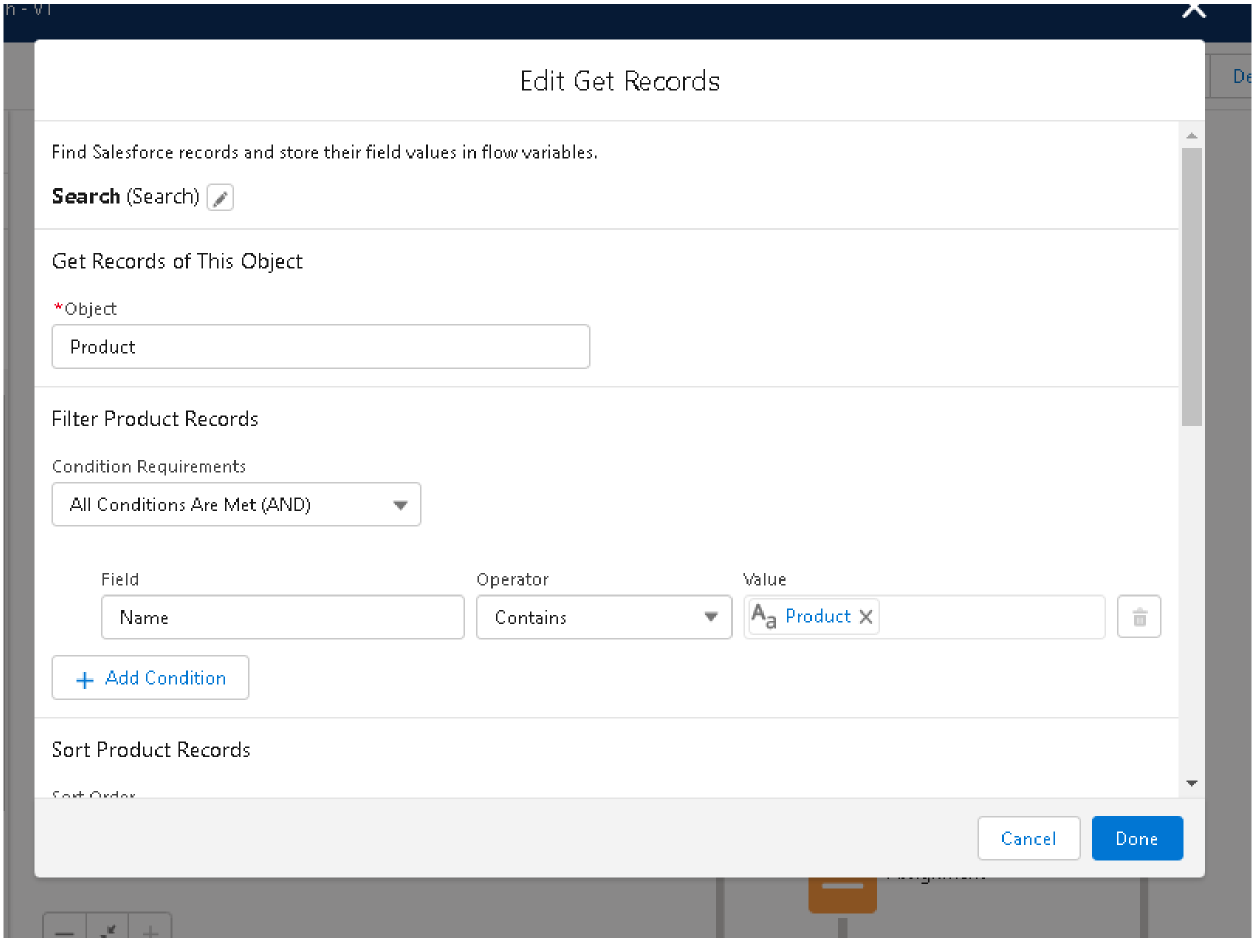
○ <em style="color: rgb(51, 51, 51); background-color: rgb(255, 255, 255); font-size: 12px;">RainbowBot</em>

○ <em style="color: rgb(51, 51, 51); background-color: rgb(255, 255, 255); font-size: 16px; font-family: &quot;Salesforce Sans&quot;, -apple-system, BlinkMacSystemFont, &quot;Segoe UI&quot;, Roboto, sans-serif;">CloudyBot</em>

○ <em style="color: rgb(51, 51, 51); background-color: rgb(255, 255, 255); font-size: 16px; font-family: &quot;Salesforce Sans&quot;, -apple-system, BlinkMacSystemFont, &quot;Segoe UI&quot;, Roboto, sans-serif;">Assembly System</em>

Screen Properties

Product Quick Search
(Product_Quick_Search)



Edit Get Records

Find Salesforce records and store their field values in flow variables.

Search (Search)

Get Records of This Object

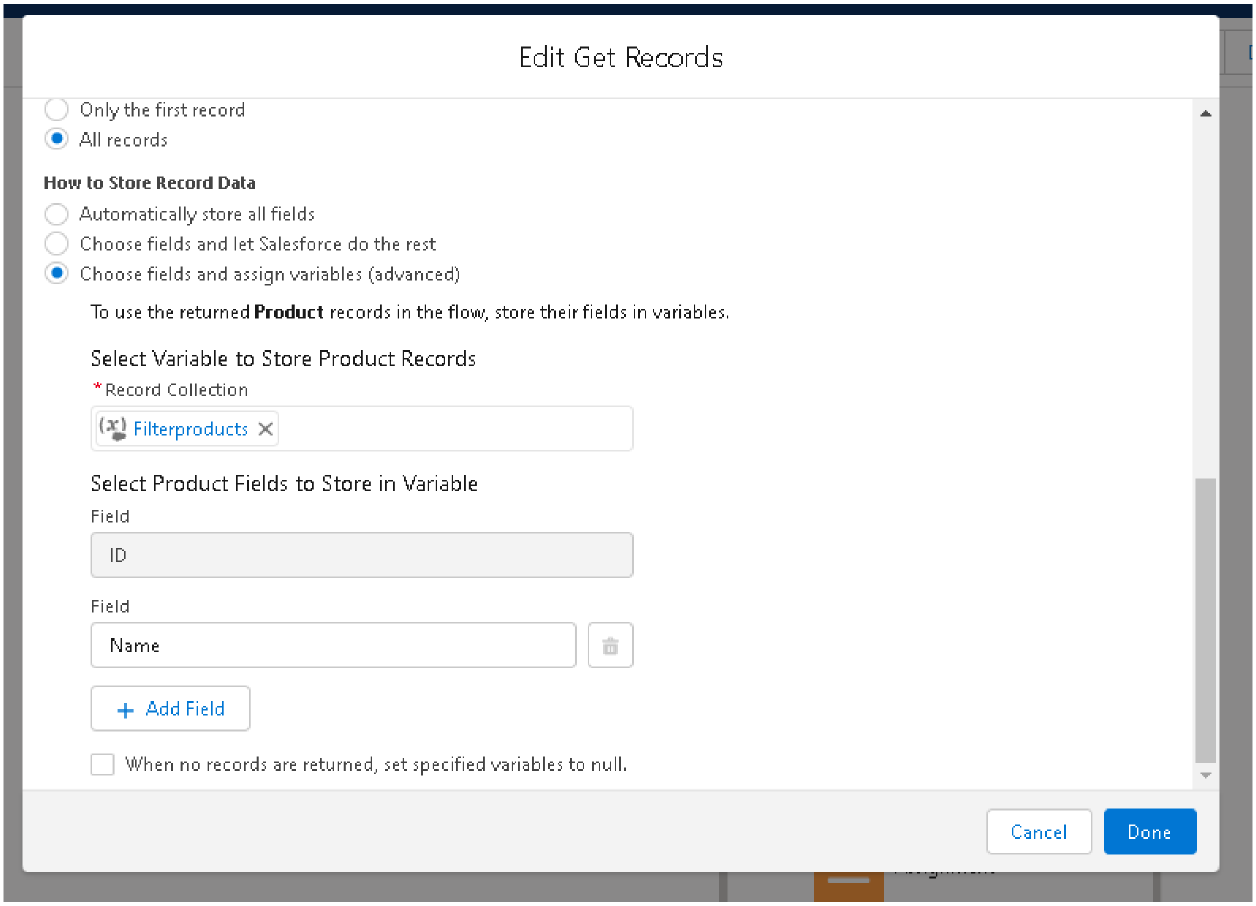
*Object: Product

Filter Product Records

Condition Requirements: All Conditions Are Met (AND)

Field	Operator	Value
Name	Contains	Product

Sort Product Records



Edit Get Records

○ Only the first record
● All records

How to Store Record Data

○ Automatically store all fields
○ Choose fields and let Salesforce do the rest
● Choose fields and assign variables (advanced)

To use the returned **Product** records in the flow, store their fields in variables.

Select Variable to Store Product Records

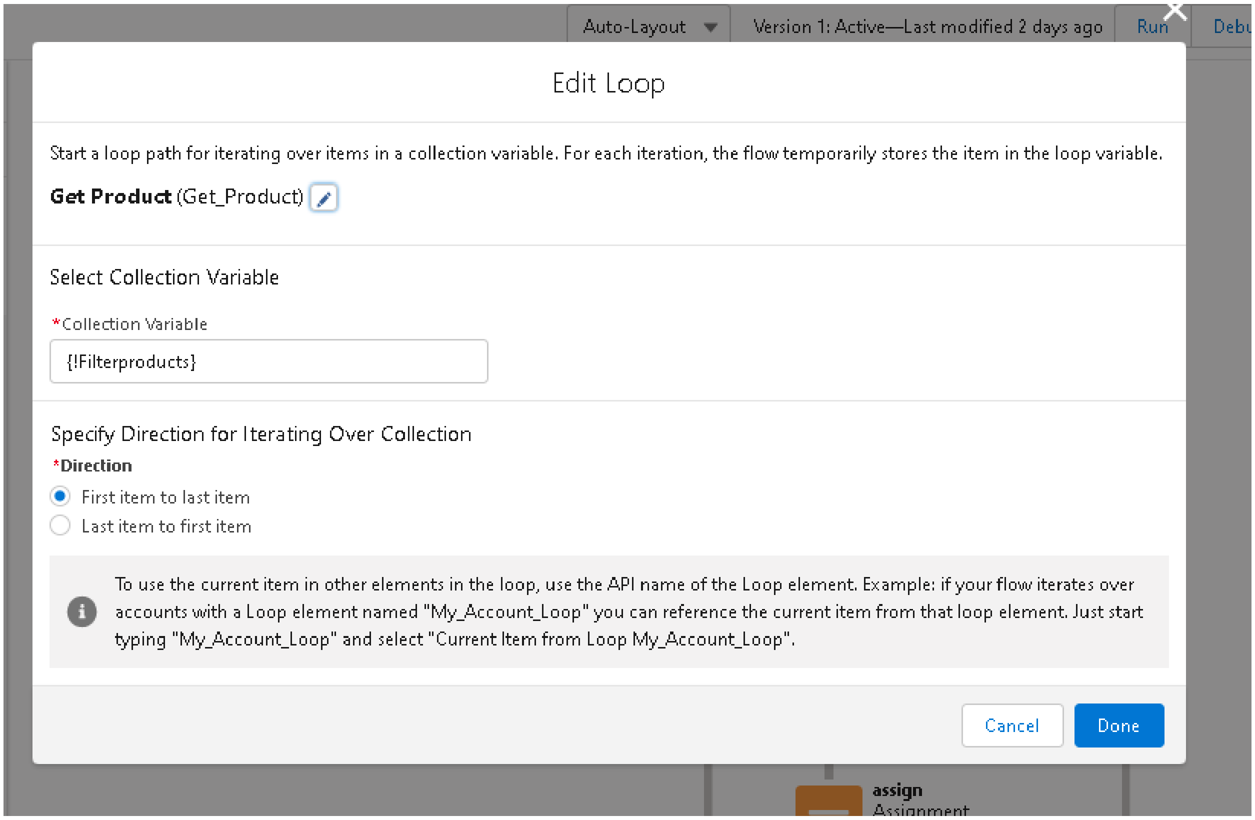
*Record Collection: Filterproducts

Select Product Fields to Store in Variable

Field: ID

Field: Name

□ When no records are returned, set specified variables to null.



Edit Loop

Start a loop path for iterating over items in a collection variable. For each iteration, the flow temporarily stores the item in the loop variable.

Get Product (Get_Product)

Select Collection Variable

*Collection Variable

{!Filterproducts}

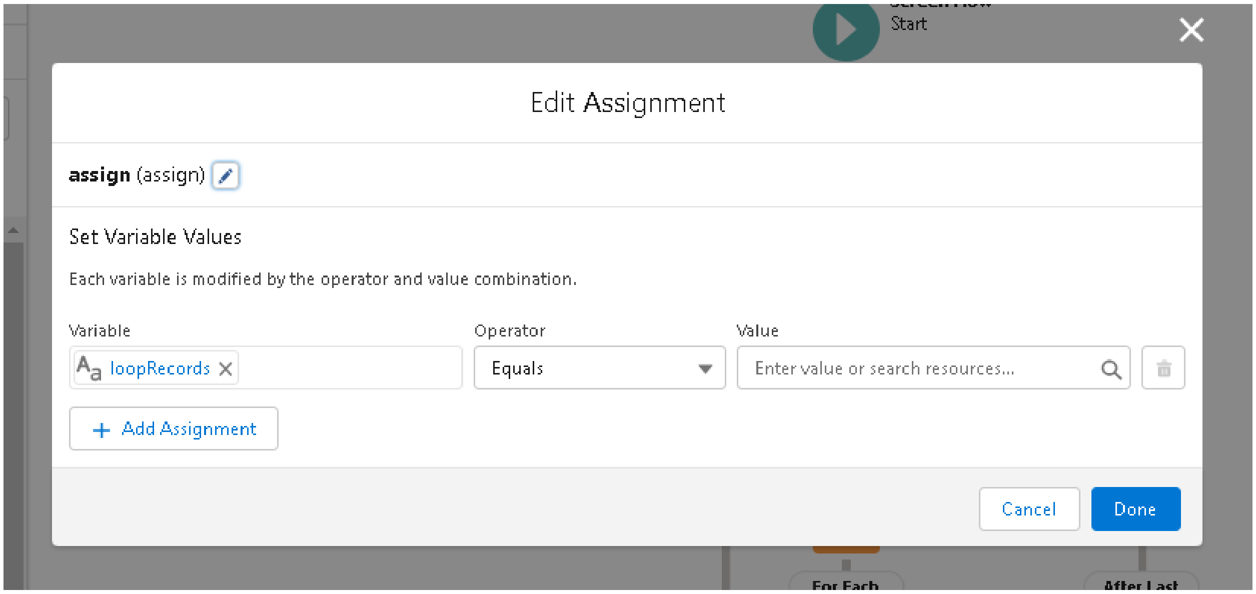
Specify Direction for Iterating Over Collection

***Direction**

● First item to last item

○ Last item to first item

To use the current item in other elements in the loop, use the API name of the Loop element. Example: if your flow iterates over accounts with a Loop element named "My_Account_Loop" you can reference the current item from that loop element. Just start typing "My_Account_Loop" and select "Current Item from Loop My_Account_Loop".



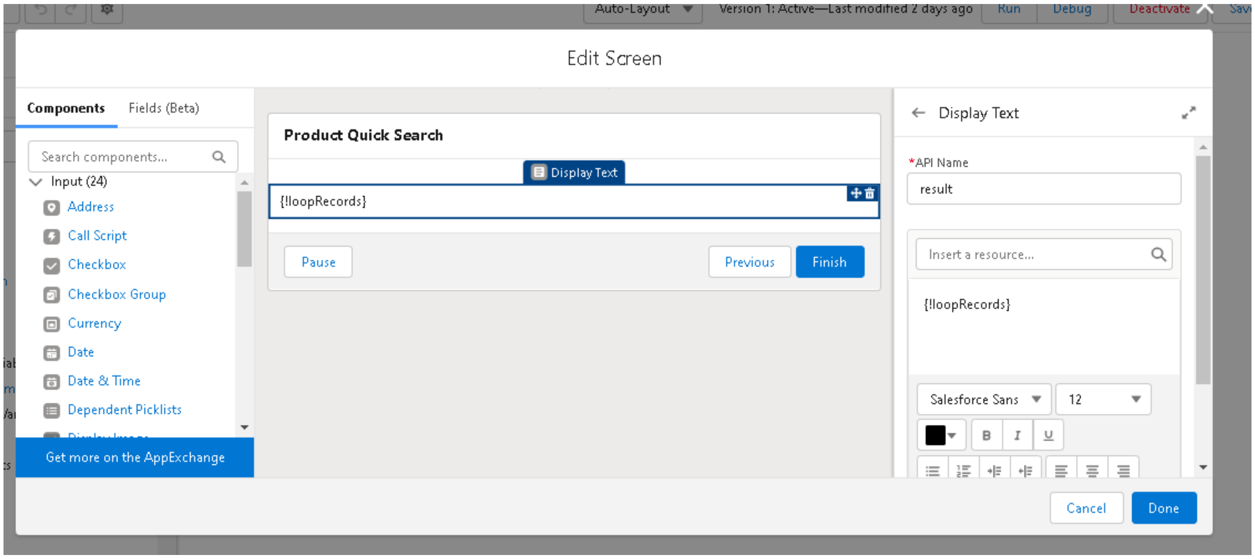
Edit Assignment

assign (assign)

Set Variable Values

Each variable is modified by the operator and value combination.

Variable	Operator	Value
loopRecords	Equals	Enter value or search resources...


Auto-Layout · Version 1: Active—Last modified 2 days ago · Run · Debug · Deactivate · Save

Edit Screen

Components · Fields (Beta)

Search components...

Input (24)
- Address
- Call Script
- Checkbox
- Checkbox Group
- Currency
- Date
- Date & Time
- Dependent Picklists

Get more on the AppExchange

Product Quick Search

Display Text

{!loopRecords}

Pause · Previous · Finish

Display Text

*API Name: result

Insert a resource...

{!loopRecords}

Salesforce Sans · 12

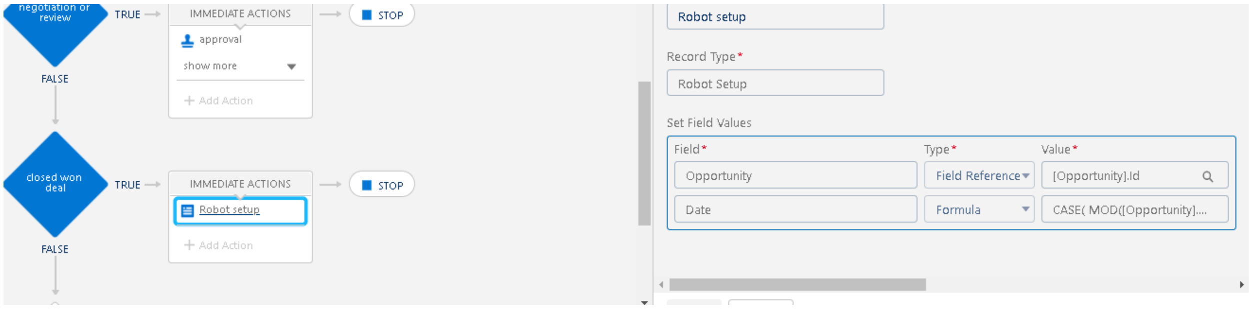
Cancel · Done

active it save

Automate Setups

1) just follw the step which Automate Opportunities where we create process builder

just add formula in



Robot setup

Record Type*: Robot Setup

Set Field Values

Field*	Type*	Value*
Opportunity	Field Reference	[Opportunity].Id
Date	Formula	CASE(MOD([Opportunity]....

closed won deal - action -create record name =Robot setup
change date critera =date-formula = CASE(MOD([Opportunity].CloseDate + 180 - DATE(1900, 1, 7),7), 0, [Opportunity].CloseDate + 181, 6, [Opportunity].CloseDate + 182, [Opportunity].CloseDate + 180)