Process Automation Specialist Superbadge

Set Up Development Org :-

- Create a new Trailhead Playground for this superbadge. Your new org will have all the special data you need. (Be sure to create a Trailhead Playground, and not a regular Developer Edition org. Only Trailhead Playgrounds have the correct data for these challenges.) Using this org for any other reason might create problems when validating the challenges.
- Use Lightning Experience.
- Install the Package(package ID 04t46000001Zch4). If you have trouble installing a managed or unmanaged package or app from AppExchange, follow the steps in this article.

Challenge 1: Automate Leads

Validation Rule

- Check the function for Length.
- Remember to check the NULL Values in Validation rule.

Queue Creation

- This is straightforward normal Queue creation
- Create Names with related to appropriate sales team.

Assignment Rule

- Create new Assignment rule for this scenario(Do not use the standard rule).
- Make sure that you rule is Active before you validate this step.

Tip: Create 2 public groups (Sales Team) and assign each one queue.

Challenge 2: Automate Accounts

Field Creations on Account Object

- **Number of deals** Field should be a Roll-Up Summary take count of COUNT Opportunities
- **Number of won deals** Field should be a Roll-Up Summary (COUNT Opportunity) with filter criteria of Closed Won
- **Amount of won deals** Field should be a Roll-Up Summary (SUM Opportunity) with filter criteria of Closed Won
- **Last won deal date** Field should be a Roll-Up Summary (MAX Opportunity)
- **Deal win percent** Field should be a Formula(Percentage field) IF Number_of_deals__c greater than 0 the , Number_of_won_deals__c /Number_of_deals__c otherwise Zero
- **Call for Service** Field should be a Formula (Date) *IF(OR(TODAY() – 730 > Last_won_deal_date__c , TODAY() + 730 < Last_won_deal_date__c), 'Yes','No')*

Validation Rules on Account Object

- For Customer – Channel

ISCHANGED(Name) && ISPICKVAL(Type, "Customer – Channel")

- For Customer – Direct

ISCHANGED(Name) && ISPICKVAL(Type, "Customer – Direct")

- For Billing State/Province

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))

- For Billing Country

BillingCountry <> “US” && BillingCountry <> “USA” && BillingCountry <> “United States” && NOT(ISBLANK(BillingCountry))

- For Shipping State/Province and Shipping Country

Don’t forget replicate For Shipping State/Province and Shipping Country same as Billing State/Province and Billing Country validation which I have mentioned above.

Challenge 3: Create Robot Setup Object

It can be done easily:

- Create a object and make sure the object name should be ***Robot_Setup__c***
- Edit the Robot name(Standard field) switch the data type from Text to AutoNumber and make sure the display format should be ***ROBOT SETUP-{0000}***
- Create following fields with correct data type:

Date————––>Date__c————––––>DATE

Notes————––> Notes__c————–––>TEXT

Day of the Week——>Day_of_the_Week__c——->TEXT

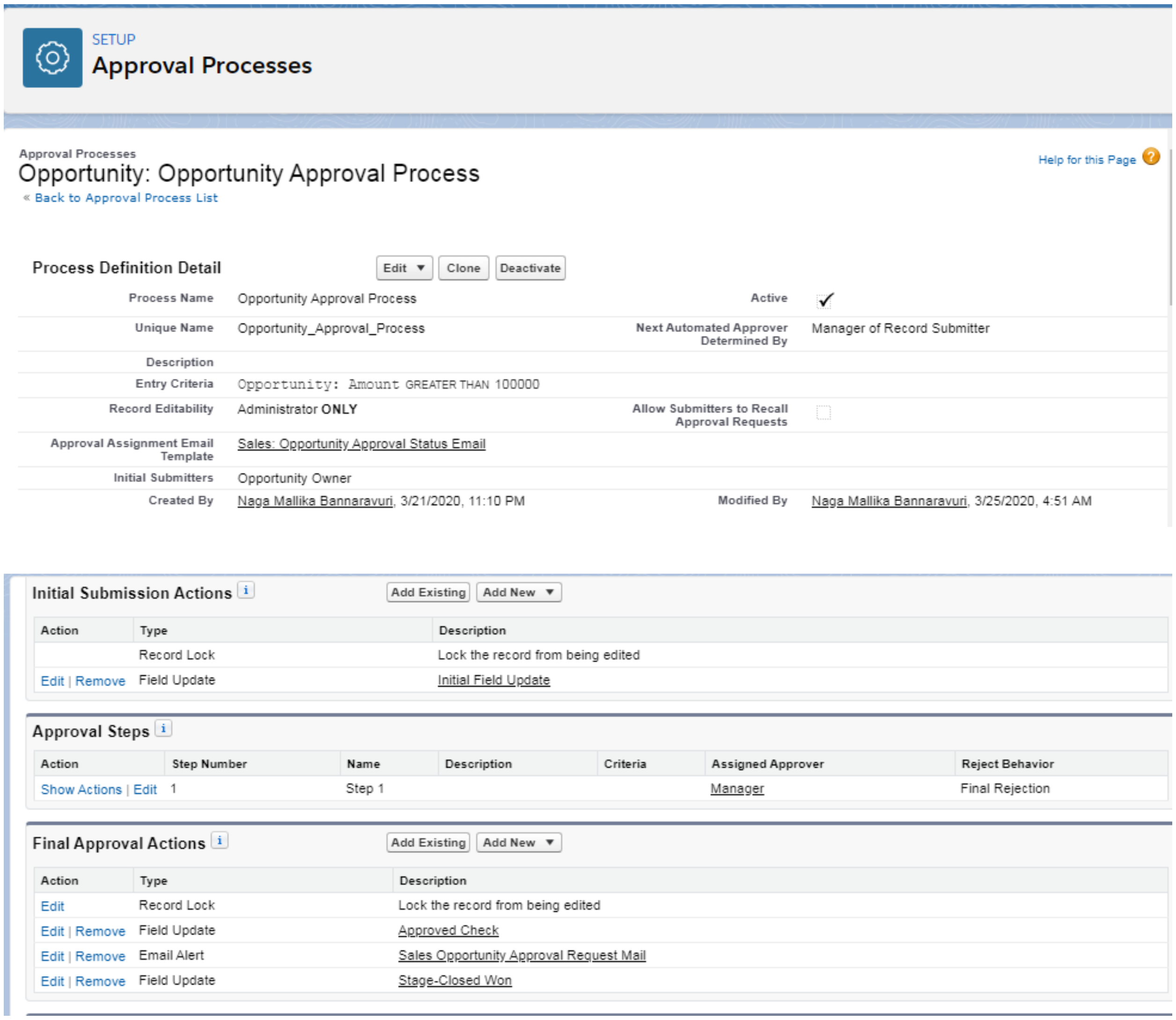
Challenge 4: Create Sales Process and Validate Opportunities

- Create Sales Process in Opportunity; the name should be ***RB Robotics Sales Process***.
- Create a record type; the name should be ***RB Robotics Process RT***.
- Add ***Awaiting Approval*** value in opportunity Stage don’t forget to add RB Robotics Process RT record type.
- Create a Checkbox field and Name it ***Approved.***
- Write a validation rule as below:

AND(Amount > 100000, Approved__c = False)

Challenge 5: Automate Opportunities

Approval Process Definition Detail: See the screenshot below for details


SETUP

Approval Processes

Approval Processes

Opportunity: Opportunity Approval Process

« Back to Approval Process List

Help for this Page

Process Definition Detail

Process Name	Opportunity Approval Process	Active	✓
Unique Name	Opportunity_Approval_Process	Next Automated Approver Determined By	Manager of Record Submitter
Description			
Entry Criteria	Opportunity: Amount GREATER THAN 100000		
Record Editability	Administrator ONLY	Allow Submitters to Recall Approval Requests	☐
Approval Assignment Email Template	Sales: Opportunity Approval Status Email		
Initial Submitters	Opportunity Owner		
Created By	Naga Mallika Bannaravuri, 3/21/2020, 11:10 PM	Modified By	Naga Mallika Bannaravuri, 3/25/2020, 4:51 AM

Initial Submission Actions

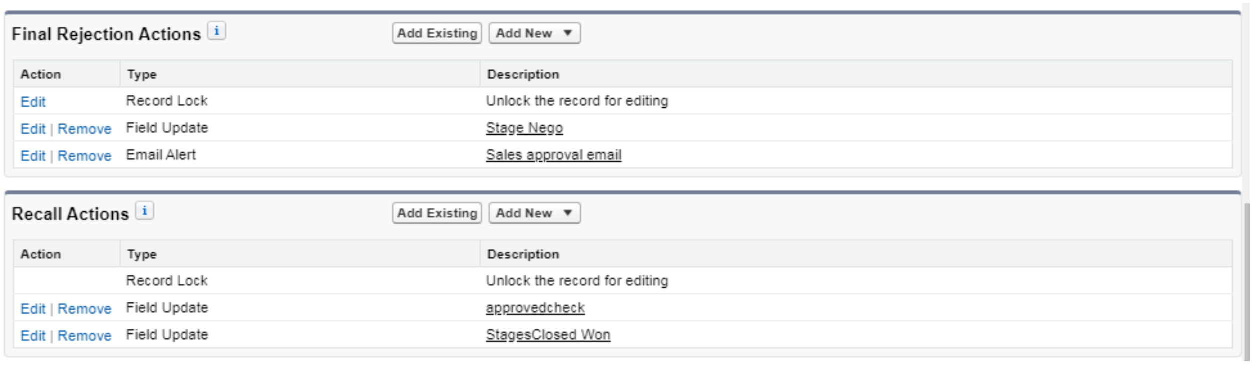
Action	Type	Description
	Record Lock	Lock the record from being edited
Edit | Remove	Field Update	Initial Field Update

Approval Steps

Action	Step Number	Name	Description	Criteria	Assigned Approver	Reject Behavior
Show Actions | Edit	1	Step 1			Manager	Final Rejection

Final Approval Actions

Action	Type	Description
Edit	Record Lock	Lock the record from being edited
Edit | Remove	Field Update	Approved Check
Edit | Remove	Email Alert	Sales Opportunity Approval Request Mail
Edit | Remove	Field Update	Stage-Closed Won



Final Rejection Actions

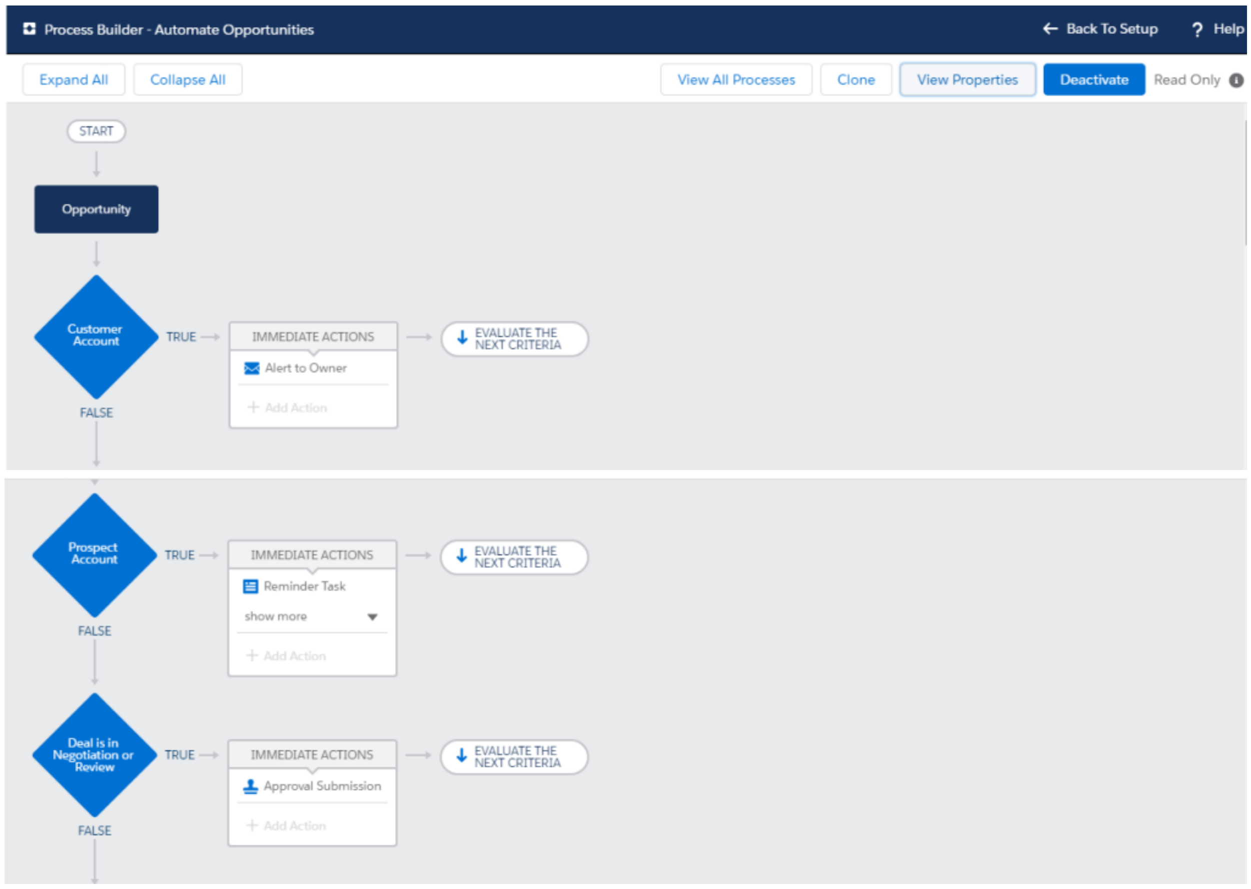
Action	Type	Description
Edit	Record Lock	Unlock the record for editing
Edit | Remove	Field Update	Stage Nego
Edit | Remove	Email Alert	Sales approval email

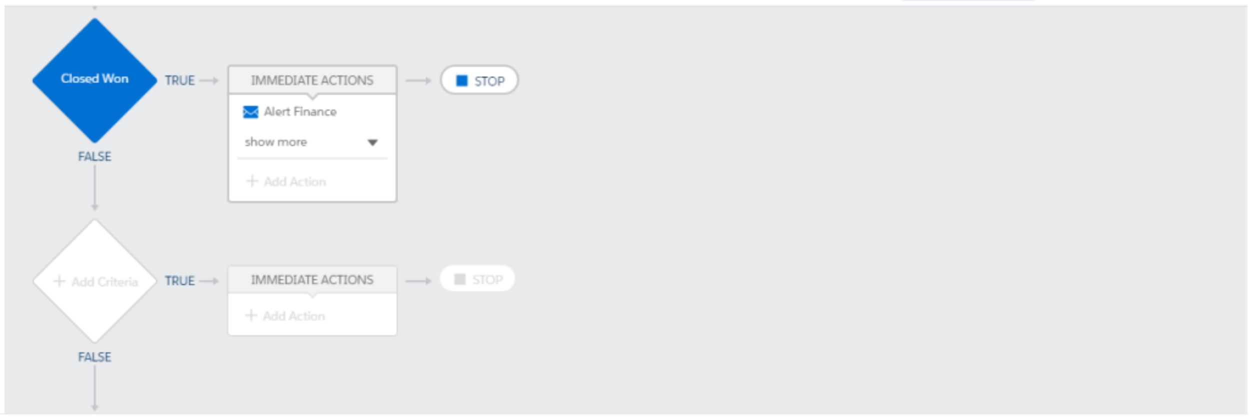
Recall Actions

Action	Type	Description
	Record Lock	Unlock the record for editing
Edit | Remove	Field Update	approvedcheck
Edit | Remove	Field Update	StagesClosed Won

It’s time to create **Process Builder**.

Name: **Automate Opportunities**

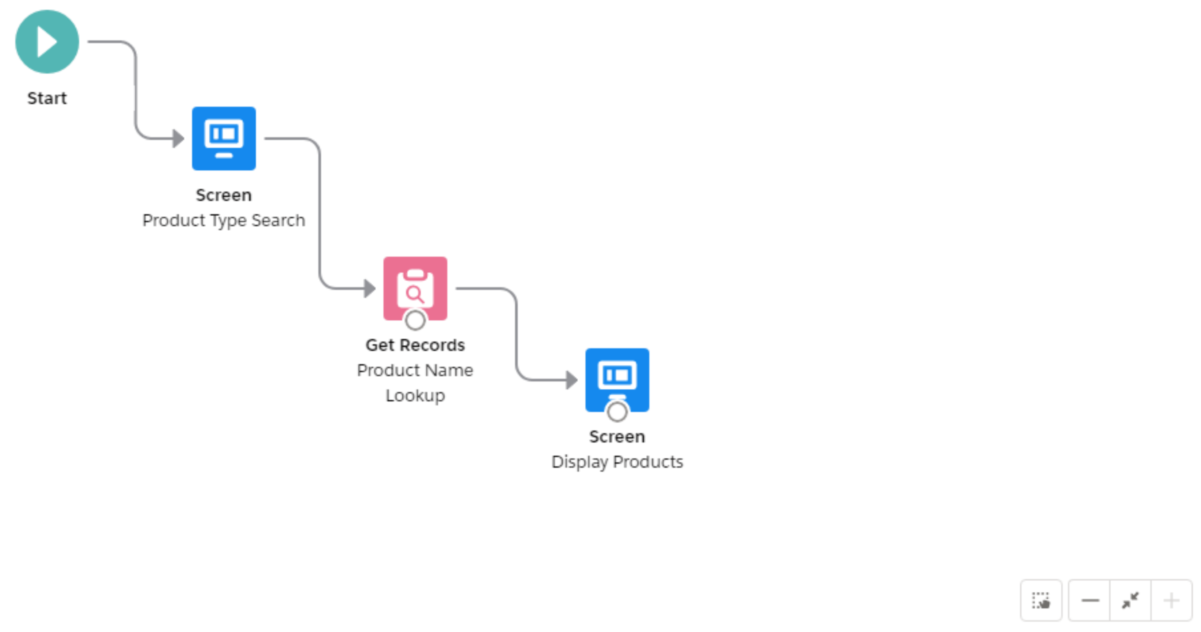




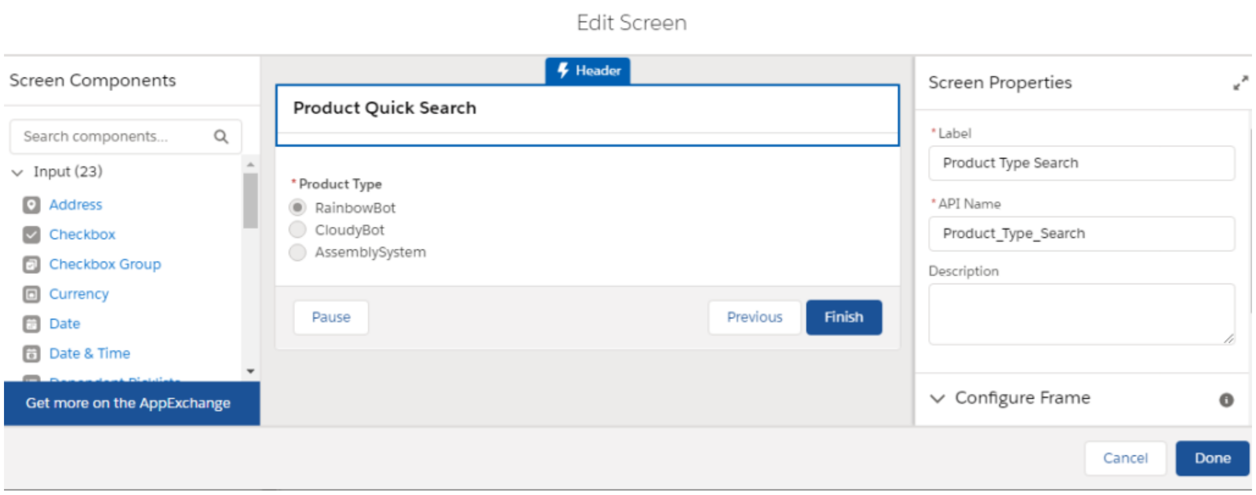
Note: If you have trouble in creating process builder, comment the errors you are getting, so that I will guide you to process it.

Challenge 6: Create Flow for Opportunities

Create the flow to display products.



Screen (Product Type Search) Properties:


Edit Screen

Screen Components

Search components...

Input (23)

- Address
- Checkbox
- Checkbox Group
- Currency
- Date
- Date & Time

Get more on the AppExchange

Header

Product Quick Search

* Product Type

- (●) RainbowBot
- ( ) CloudyBot
- ( ) AssemblySystem

Pause | Previous | Finish

Screen Properties

* Label: Product Type Search

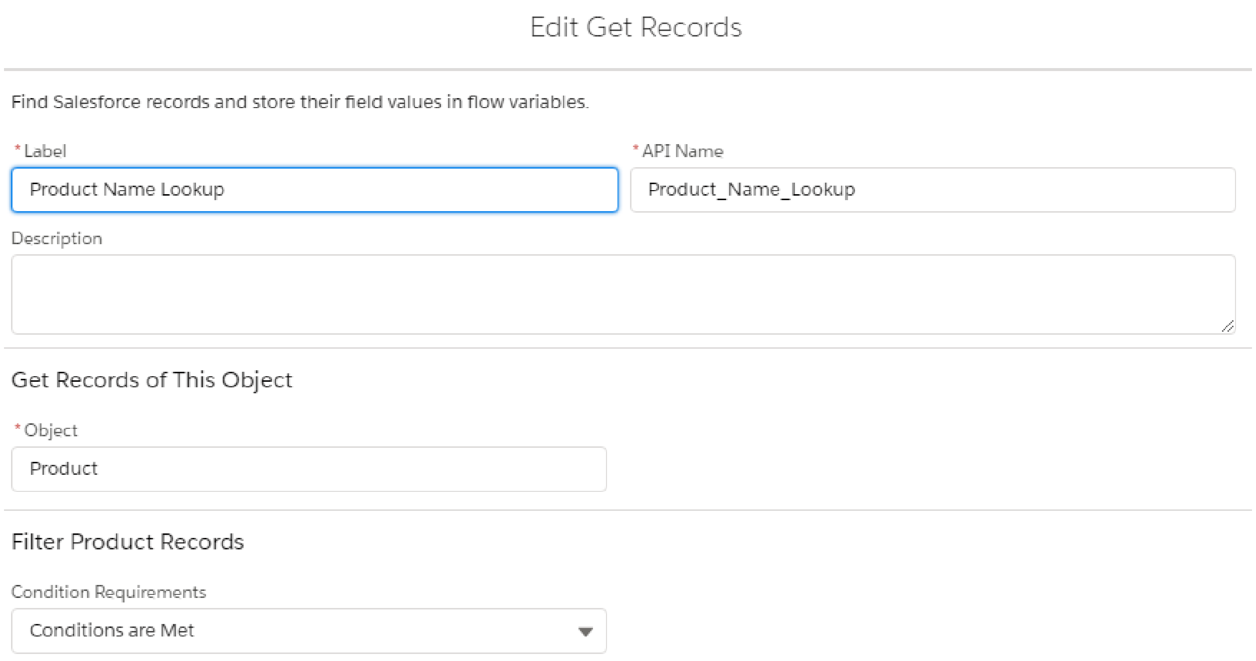
* API Name: Product_Type_Search

Description:

Configure Frame

Cancel | Done

Get Records (Product Name Lookup) Properties:


Edit Get Records

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

* Label: Product Name Lookup

* API Name: Product_Name_Lookup

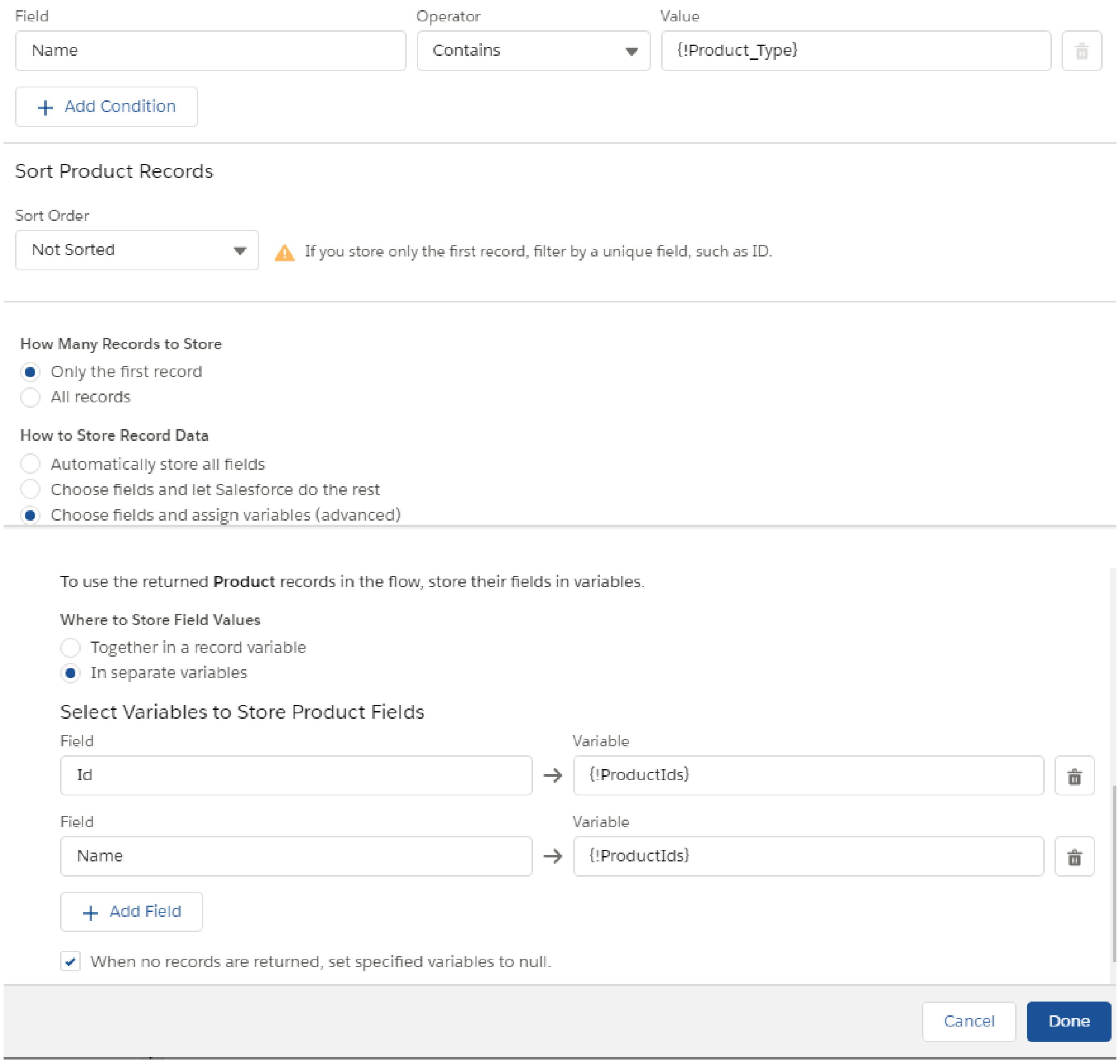
Description:

Get Records of This Object

* Object: Product

Filter Product Records

Condition Requirements: Conditions are Met



Field	Operator	Value
Name	Contains	{!Product_Type}

Sort Product Records

Sort Order: Not Sorted

If you store only the first record, filter by a unique field, such as ID.

How Many Records to Store

- [✓ 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.

Where to Store Field Values

- Together in a record variable
- [✓ In separate variables]

Select Variables to Store Product Fields

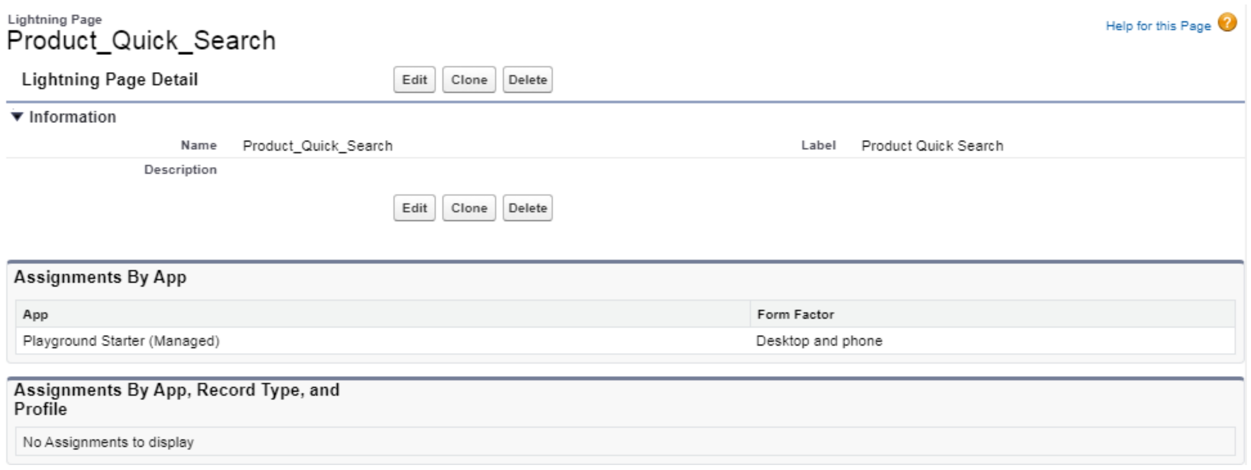
Field	Variable
Id	{!ProductIds}
Name	{!ProductIds}

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

- Activate the flow
- Add the flow to the opportunity screen using app builder.

Create a Record Page on Opportunity Object:

Go to Lightning App Builder page and click new. Record Page Properties are as follows


Lightning Page

Product_Quick_Search

Lightning Page Detail

Information

Name	Product_Quick_Search	Label	Product Quick Search
Description			

Assignments By App

App	Form Factor
Playground Starter (Managed)	Desktop and phone

Assignments By App, Record Type, and Profile

No Assignments to display

- Add the component on newly created Opportunity Record Page.
- Please don't forgot to Activate the page.

Challenge 7: Automate Setup

- Change the datatype for "Day of the week" field from TEXT to Formula (TEXT) and use the following the formula to get Day of the week

CASE(MOD(Date__c – DATE(1900, 1, 7), 7), 0, "Sunday", 1, "Monday", 2, "Tuesday", 3, "Wednesday", 4, "Thursday", 5, "Friday", 6, "Saturday",''Error")

Or You can use this formula also instead of above formula

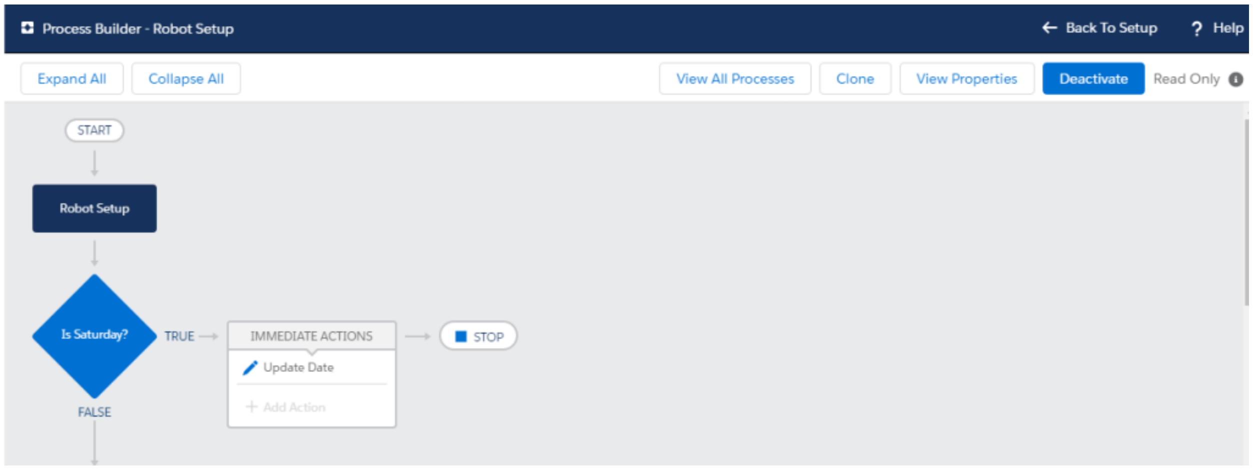
```
CASE(WEEKDAY( Date__c ),
1, "Sunday",
2, "Monday",
3, "Tuesday",
4, "Wednesday",
5, "Thursday",
6, "Friday",
7, "Saturday",
Text(WEEKDAY( Date__c )))
```

Create Another **Process Builder (Name: Robot Setup)**

Conditions are as below:

- If Day of the week is Saturday , change [Robot_Setup__c].Date__c +2

- If Day of the week is Saturday , change [Robot_Setup__c].Date__c +1



Activate the Process and you are done!

APEX SPECIALIST SUPERBADGE

Set Up Development Org :-

1. Create a new Trailhead Playground for this superbadge.
2. Install this unlocked package (package ID: 04t6g000008av9iAAA).
3. Add picklist values Repair and Routine Maintenance to the Type field on the Case object.
4. Update the Case page layout assignment to use the Case (HowWeRoll) Layout for your profile.
5. Rename the tab/label for the Case tab to Maintenance Request.
6. Update the Product page layout assignment to use the Product (HowWeRoll) Layout for your profile.
7. Rename the tab/label for the Product object to Equipment.
8. Click on App Launcher and search Create Default Data then Click Create Data to generate sample data for the application.

Challenge 1

Quiz

There is a basic quiz on the reles on not sharing the superbadge answers.

Challenge 2

Automated Record Creation

1.Go to the App Launcher -> Search How We Roll Maintenance -> click on Maintenance Requests -> click on first case -> click Details -> change the type Repair to Routine Maintenance -> select Origin = Phone -> Vehicle = select Teardrop Camper , save it.
2.Feed -> Close Case = save it..
3.Go to the Object Manager -> Maintenance Request ->Field & Relationships ->New ->Lookup Relationship -> next -> select Equipment ->next -> Field Label = Equipment ->next->next->next -> save it .
4.Now go to the developer console use below code.

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);

    }

}
```

1. After saving the code go back the How We Roll Maintenance ,
2. click on Maintenance Requests -> click on 2nd case -> click Details -> change the type Repair to Routine Maintenance -> select Origin = Phone -> Vehicle = select Teardrop Camper , save it.
3. Feed -> Close Case = save it..

Now check challenge.

<u>Challenge 3</u>

Synchronize Salesforce data with an external system

- Setup -> Search in quick find box -> click Remote Site Settings -> Name = Warehouse URL , Remote Site URL = https://th-superbadge-apex.herokuapp.com , make sure active is selected.
- Go to the developer console use below code .
-

WarehouseCalloutService.apxc :-

```
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());
```

Now check Challenge.

Challenge 4
Schedule synchronization using Apex code

- Go to the developer console use below code ,

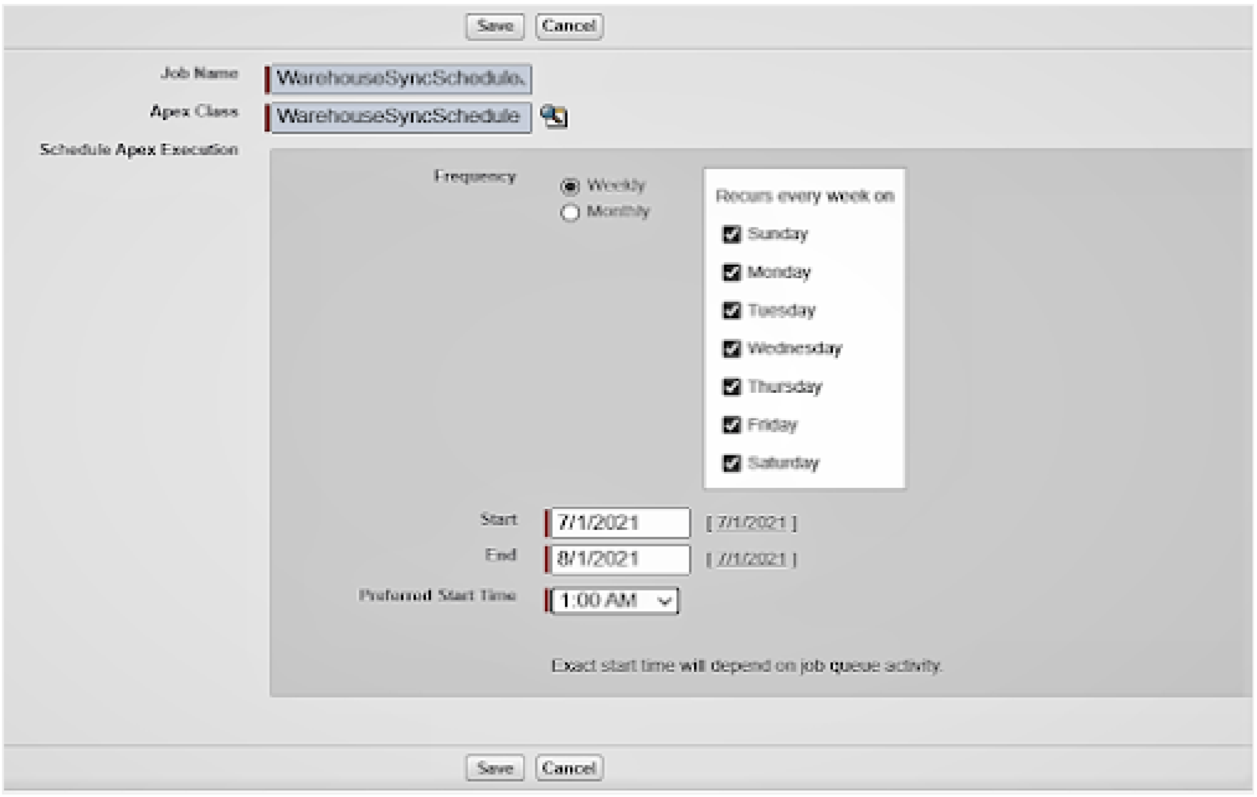
WarehouseSyncShedule.apxc :-

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

Save it , after that...

- Go to setup -> Seacrh in Quick find box -> Apex Classes -> click Schedule

Apex and Jb Name = WarehouseSyncScheduleJob , Apex Class = WarehouseSyncSchedule as it is below shown in the image ,



Job Name: WarehouseSyncSchedule
Apex Class: WarehouseSyncSchedule
Schedule Apex Execution
Frequency: (●) Weekly ( ) Monthly
Recurs every week on: ✓ Sunday ✓ Monday ✓ Tuesday ✓ Wednesday ✓ Thursday ✓ Friday ✓ Saturday
Start: 7/1/2021 [7/1/2021]
End: 8/1/2021 [7/1/2021]
Preferred Start Time: 1:00 AM
Exact start time will depend on job queue activity.

Now check challenge.

Challenge 5

Test automation logic

- Go to the developer console use below code ,

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);
  }
}
```

run all

Now check challenge.

Challenge 6

Test callout logic

- Go to the developer console use below code ,

WarehouseCalloutService.apxc :-

```
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);
      }
```

```
        }
    }
}
```

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-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,"quant
ity":5,"name":"Generator 1000
kW","maintenanceperiod":365,"lifespan":120,"cost":5000,"sku":"100003"}]');
        response.setStatusCode(200);
        return response;
    }
}
```

run all

Now check challenge.

Challenge 7

Test scheduling logic

- Go to the developer console use below code ,

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();
        //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 ');


    }
}
```

run all
Now check challenge.