

Apex Specialist Superbadge

Challenge 1:

This is the first challenge where we attend a quiz answering some general questions regarding the superbadge challenge that we are doing.

Challenge 2:

It is all about preparing my organization with the necessary package installations and customizations as per given in the Prepare Your Organization section to complete the Apex Specialist Superbadge.

Challenge 3:

In this challenge we automate record creation using apex class and apex trigger by creating an apex class called MaintenanceRequestHelper and an apex trigger called MaintenanceRequest.

Apex Class code:

```
public with sharing class MaintenanceRequestHelper {
    public static void updateWorkOrders(List<Case>
    caseList){List<Case> newCases = new List<Case>();
    Map<String,Integer> result=getDueDate(caseList);
    for(Case c : caseList){
        if(c.status=='closed')
        if(c.type=='Repair' || c.type=='Routine
        Maintenance'){Case newCase = new Case();
        newCase.Status='New';
        newCase.Origin='web';
        newCase.Type='Routine Maintenance';
        newCase.Subject='Routine Maintenance of
        Vehicle';newCase.Vehicle_c=c.Vehicle_c;
        newCase.Equipment_c=c.Equipment_c;
        newCase.Date_Reported_c=Date.today();
        if(result.get(c.Id)!=null)
        newCase.Date_Due__
        c=Date.today()+result.get(c.Id); else
        newCase.Date_Due_c=Date.today();
```

```

newCases.add(newCase);
}

}
insert newCases;
}
/

public static Map<String,Integer> getDueDate(List<case> CaseIDs){
Map<String,Integer> result = new Map<String,Integer>();
Map<Id, case> caseKeys = new Map<Id, case> (CaseIDs);
List<AggregateResult> wpc=[select Maintenance_Request_r.ID
cID,min(Equipment__r.Maintenance_Cycle__c)cycle
from Work_Part_c where Maintenance_Request_r.ID in :caseKeys.keySet() group by
Maintenance_Request_r.ID];
for(AggregateResult res :wpc){
Integer addDays=0;
if(res.get('cycle')!=null)
addDays+=Integer.valueOf(res.get('cycle'));
result.put((String)res.get('cID'),addDays);
}
return result;
}
}

```

Apex Trigger code:

```

trigger MaintenanceRequest on Case (before update, after update) {
/ ToDo: Call MaintenanceRequestHelper.updateWorkOrders
if(Trigger.isAfter)
MaintenanceRequestHelper.updateWorkOrders(Trigger.New);
}

```

Challenge4:

In challenge 3 we synchronize salesforce data with an external system using apex class of name WarehouseCalloutService which is already given and after writing code in it and executing it anonymously in a separate window, the process will be successful.

Apex class code:

```

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() {
        / ToDo: complete this method to make the callout (using @future) to the

        / REST endpoint and update equipment on
        hand. HttpResponse response = getResponse();
        if(response.getStatusCode() == 200)
        {
            List<Product2> results = getProductList(response); /get list of products from Http callout
            response
            if(results.size() >0)
            upsert results Warehouse_SKU__c; / Upsert the products in your org based on the
            externalIDSKU
        }
    }

    / Get the productlist from the external link
    public static List<Product2> getProductList(HttpResponse response)
    {
        List<Object> externalProducts = (List<Object>) JSON.deserializeUntyped(response.getBody());
        / desrialize the json response
        List<Product2> newProducts = new List<Product2>();
        for(Object p : externalProducts)
        {
            Map<String, Object> productMap= (Map<String, Object>)
            p; Product2 pr = new Product2();
            / Map the fields in the responseto the appropriate fields in the Equipmentobject
            pr.Replacement_Part__c = (Boolean)productMap.get('replacement');
            pr.Cost__c = (Integer)productMap.get('cost');
            pr.Current_Inventory__c =
            (Integer)productMap.get('quantity'); pr.Lifespan_Months__c
            = (Integer)productMap.get('lifespan');
            pr.Maintenance_Cycle__c =
            (Integer)productMap.get('maintenanceperiod'); pr.Warehouse_SKU__c =
            (String)productMap.get('sku');

```

```

pr.ProductCode = (String)productMap.get('_id');
pr.Name = (String)productMap.get('name');
newProducts.add(pr);
}
return newProducts;
}
/ Send Http GET request and receive Http
responsepublicstatic HttpResponse
getResponse() {
Http http = new Http();
HttpRequest request = new HttpRequest();
request.setEndpoint(WAREHOUSE_URL);
request.setMethod('GET');
HttpResponse response = http.send(request);

return response;
}
}

```

Execute anonymous window:

```
WarehouseCalloutService.runWarehouseEquipmentSync();
```

Challenge5:

In challenge 4 we will be scheduling our synchronization using WarehouseSyncSchedule in the apex class and execute a code in an anonymous window.

Apex Class code:

```

global class WarehouseSyncSchedule implements Schedulable{
/ implement scheduled code here
global void execute (SchedulableContext sc){
WarehouseCalloutService.runWarehouseEquipmentSync();
/ optionalthis can be done by debug
modeString sch = '00 00 01 * * ?';/on 1 pm
System.schedule('WarehouseSyncScheduleTest', sch, new WarehouseSyncSchedule());
}
}

```

Execute anonymous window:

```
WarehouseSyncSchedule scheduleInventoryCheck();
```

Challenge6:

In this challenge we are testing our automation logic using apex trigger class MaintenanceRequest and three apex classes where two are used for testing and one is used for sharing and those classes are given below.

Apex trigger:

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

Apex class:

```
@IsTest
private class InstallationTests {
    private static final String STRING_TEST = 'TEST';

    private static final String NEW_STATUS =
    '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 =
    'AMC Spirit'; public static String CRON_EXP = '0 0 1 * * ?';
    static testmethod void
    testMaintenanceRequestNegative() {Vehicle_cvehicle =
    createVehicle();
    insert vehicle;
    Id vehicleId = vehicle.Id;
    Product2 equipment =
    createEquipment();insert equipment;
```

```

Id equipmentId = equipment.Id;
Case r = createMaintenanceRequest(vehicleId,
equipmentId); insert r;
Work_Part__c w = createWorkPart(equipmentId,
r.Id); insert w;
Test.startTest();
r.Status =
WORKING;
update r;
Test.stopTest();
List<case> allRequest =
[SELECT IdFROM Case];
Work_Part__c workPart = [SELECT
IdFROM Work_Part__c
WHERE Maintenance_Request__c =:
r.Id];System.assert(workPart !=
null);System.assert(allRequest.size() == 1);
}

static testmethod void testWarehouseSync() {
Test.setMock(HttpCalloutMock.class, new WarehouseCalloutServiceMock());
Test.startTest();
String jobId =
System.schedule('WarehouseSyncSchedule', CRON_EXP,
new WarehouseSyncSchedule());
CronTrigger ct = [SELECT Id, CronExpression, TimesTriggered,
NextFireTime FROM CronTrigger
WHERE id = :jobId];
System.assertEquals(CRON_EXP, ct.CronExpression);

System.assertEquals(0, ct.TimesTriggered);
Test.stopTest();
}

private static Vehicle__c createVehicle() {
Vehicle__c v = new Vehicle__c(Name = STRING_TEST);
return v;
}

private static Product2 createEquipment() {
Product2 p = new Product2(Name =
STRING_TEST, Lifespan_Months__c = 10,

```

```

Maintenance_Cycle__c =
10, Replacement_Part_c
= true);return p;
}

private static Case createMaintenanceRequest(Id vehicleId,Id
equipmentId) { Case c = new Case(Type = REPAIR,
Status = NEW_STATUS,
Origin =
REQUEST_ORIGIN,
Subject = REQUEST_SUBJECT,
Equipment_c =
equipmentId,Vehicle_c =
vehicleId); return c;
}

private static Work_Part_c createWorkPart(Id equipmentId, Id
requestId) { Work_Part_c wp = new Work_Part_c(Equipment_c =
equipmentId, Maintenance_Request__c = requestId);
return wp;
}
}
}

```

Apex class:

```

public with sharing class MaintenanceRequestHelper
{ public static void updateWorkOrders(List<case>
caseList){List<case> newCases = new List<case>();
Map<String,Integer> result=getDueDate(caseList);
for(Case c : caseList){
if(c.status=='closed')
if(c.type=='Repair' || c.type=='Routine
Maintenance'){Case newCase = new Case();
newCase.Status='New';
newCase.Origin='web';

newCase.Type='Routine Maintenance';
newCase.Subject='Routine Maintenance of
Vehicle';newCase.Vehicle_c=c.Vehicle_c;
newCase.Equipment_c=c.Equipment_c;
newCase.Date_Reported_c=Date.today();
if(result.get(c.Id)!=null)

```

```

newCase.Date_Due__
c=Date.today()+result.get(c.Id); else
newCase.Date_Due_c=Date.today();
newCases.add(newCase);
}
}
insert newCases;
}
/
public static Map<String,Integer> getDueDate(List<case> CaseIDs){
Map<String,Integer> result = new Map<String,Integer>();
Map<Id, case> caseKeys = new Map<Id, case> (CaseIDs);
List<aggregateresult> wpc=[select Maintenance_Request_r.ID
cID,min(Equipment__r.Maintenance_Cycle__c)cycle
from Work_Part_c where Maintenance_Request_r.ID in :caseKeys.keySet() group by
Maintenance_Request_r.ID];
for(AggregateResult res :wpc){
Integer addDays=0;
if(res.get('cycle')!=null)
addDays+=Integer.valueOf(res.get('cycle'));
result.put((String)res.get('cID'),addDays);
}
return result;
}
}

```

Apex class:

```

@Test
public class MaintenanceRequestTest {
static List<case> caseList1 = new List<case>();
static List<product2> prodList = new List<product2>();
static List<work_part_c> wpList = new List<work_part_
c>();@testSetup
static void getData(){
caseList1= CreateData( 300,3,3,'Repair');

}
}

```



```

public static List<case> CreateData( Integer numOfcase, Integer numofProd, Integer
numofVehicle,
String type){
List<case> caseList = new List<case>();
/ Create Vehicle
Vehiclec vc = new Vehiclec();
vc.name='Test Vehicle';
upsert vc;
/ Create Equipment
for(Integer
i=0;i<numofProd;i++){Product2
prod = new Product2();
prod.Name='Test Product'+i;
if(i!=0)
prod.Maintenance_Cycle_
c=i;prod.Replacement_Part_c=true;
prodList.add(prod);
}
upsert prodlist;
/ Create Case
for(Integer i=0;i<
numOfcase;i++){ Case newCase
= new Case();
newCase.Status='New';
newCase.Origin='web';
if( math.mod(i, 2) ==0)
newCase.Type='Routine Maintenance';
else
newCase.Type='Repair';
newCase.Subject='Routine Maintenance of
Vehicle' +i;newCase.Vehicle__c=vc.Id;
if(i<numofProd)
newCase.Equipment__c=prodList.get(i).ID;
else
newCase.Equipment__c=prodList.get(0).ID;
caseList.add(newCase);
}
upsert caseList;

```

```

for(Integer i=0;i<numofProd;i++){
Work_Part_c wp = new Work_Part_c();
wp.Equipment_c =prodlist.get(i).Id ;
wp.Maintenance_Request__c=caseList.get(i).id;

wplist.add(wp) ;
}
upsert
wplist;
return
caseList;
}

public static testmethod void
testMaintenanceHelper(){ Test.startTest();
getData();
for(Case cas:
caseList1)cas.Status
='Closed'; update
caseList1;
Test.stopTest();
}
}

```

Challenge7:

In challenge6 we are testing our callout logic by using two apex classes which are used for testing where one of the classes implements HTTPCalloutMock.

Apex class:

```

@IsTest
private class WarehouseCalloutServiceTest {
/ implement your mock callout test
here@isTest
static void testWareHouseCallout(){
Test.setMock(HttpCalloutMock.class, new WarehouseCalloutServiceMock());
WarehouseCalloutService.runWarehouseEquipmentSync();
}
}

```

```
}
```

Apex class:

```
@isTest
public class WarehouseCalloutServiceMock implements HTTPCalloutMock {
    / implement http mock callout
    public HTTPResponse respond (HttpRequest
    request){HTTPResponse response = new
    HTTPResponse(); response.setHeader('Content-
    type','application/json');
    response.setBody('{"_id":"55d66226726b611100aaf741","replacement":false,"quantity":5,"name
    ": "Generator 1000
    kW","maintenanceperiod":365,"lifespan":120,"cost":5000,"sku":"100003"},{"_id":"55d66226726b6
    11
    100aaf742","replacement":true,"quantity":183,"name":"Cooling
    Fan","maintenanceperiod":0,"lifespan":0,"cost":300,"sku":"100004"},{"_id":"55d66226726b6111
    00a af743","replacement":true,"quantity":143,"name":"Fuse
    20A","maintenanceperiod":0,"lifespan":0,"cost":22,"sku":"100005"}]');
    response.setStatusCode(200);
    return response;
    }
}
```

Challenge8:

In this challenge we are testing our Scheduling logic by using a apex test class to test our scheduling logic and the code is given below.

Apex class:

```
@isTest
private class WarehouseSyncScheduleTest {
    public static String CRON_EXP = '0 0 0 15 3 ?
    2022'; static testmethod void testjob(){
    MaintenanceRequestTest.CreateData(
    5,2,2,'Repair'); Test.startTest();
    Test.setMock(HTTPCalloutMock.class, new WarehouseCalloutServiceMock());
}
```

```
String jobID= System.schedule('TestScheduleJob', CRON_EXP, new WarehouseSyncSchedule());
/ List<Case> caselist = [Select count(id) from case where
case]Test.stopTest();
}
}
```

with this the Apex Specialist Superbadge is completed successfully.

Process Automation Specialist Superbadge

Challenge 1:

It is the same as the previous superbadge challenge 1 where we answer a quiz before moving into the actual Superbadge challenges.

Challenge 2:

This challenge is all about automating leads where we create a Validation rule under leads and you can give any Rule Name and the Error condition formula will be given below for validating leads. After this we have to create two Queues with the given name as per in the instruction of the challenge and then create an assignment rule. If all these things are done properly, the challenge will be completed without any problems.

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))))
```

Lead Assignment Rule

Trailhead

Add rule entries that specify the criteria used to route leads. You can reorder rule entries on this page after you create them.

Rule Detail

Edit

Rule Name

Trailhead

Active

☒

Created By

Nidhi gupta, 2020-06-03, 8:33 a.m.

Modified By

Nidhi gupta, 2020-06-03, 8:33 a.m.

Edit

Rule Entries

New Reorder

Action	Order	Criteria	Assign To
Edit Del	<input type="text" value="1"/>	Lead: Lead Source EQUALS Web	Rainbow Sales
Edit Del	<input type="text" value="2"/>	Lead: Lead Source NOT EQUAL TO Web	Assembly System Sales

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

Step 2: Select the criteria for this rule entry

Run this rule if the :

Field	Operator	Value	
<input type="text" value="Lead: Lead Source"/>	<input type="text" value="not equal to"/>	<input type="text" value="Web"/>	<input checked="" type="checkbox"/> AND
<input type="text" value="--None--"/>	<input type="text" value="--None--"/>	<input type="text" value=""/>	<input type="checkbox"/> AND
<input type="text" value="--None--"/>	<input type="text" value="--None--"/>	<input type="text" value=""/>	<input type="checkbox"/> AND
<input type="text" value="--None--"/>	<input type="text" value="--None--"/>	<input type="text" value=""/>	<input type="checkbox"/> AND

Add Filter Logic...

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

Queue

Email Template

☐ Do Not Reassign Owner

Challenge 3:

In this challenge we are given the task of automating accounts by creating Roll Up Summary fields as it is given in the instructions and after that by creating two Error Condition Formulas we automate our accounts and the code will be given below for these two formulas

Error Condition Formula 1:

```

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 ))
)

```

N:MS:MO:MT:NE:NV:NH:NJ:NM:NY:NC:ND:OH:OK:OR:PA:RI:SC:SD:TN:TX:UT:VT:VA:WA:WV:WI:WY", ShippingState))
),NOT(OR(BillingCountry ="US",BillingCountry ="USA",BillingCountry ="United States",ISBLANK(BillingCountry))),
 NOT(OR(ShippingCountry ="US",ShippingCountry ="USA",ShippingCountry ="United States",ISBLANK(ShippingCountry))))

Error Condition Formula2:

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

Challenge 4:

It is the easiest challenge in this superbadge where we dont have to do a lot of things, we only have to create Robot Setup object with a master-detail relationship with the opportunity and the create a few fields as per given in the challenge instructions.

Challenge5:

In this challenge we are creating a Sales Process and Validating its opportunities, First we have to create a field with checkbox type with the name Approval where it can only be viewed by System Administrators and Sales Managers.Then we have add a picklist value as Awaiting Approval to the filed Stage.Lastly we have to add the desired fields and then add a Validation rule in the Opportunity object.

Validation Rule:

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

Challenge 6:

In this challenge we are Automating Opportunities, First we have to create three Email Templates upon reading instructions and create a approval process by selecting opportunity object in the approval process with the necessary field updatesin the process and set a criteriawhere this processwill only run if the criteria is met.

Then go to the process builderand start buildinga process by selecting a object firstandby setting four criterias where each criteria will do a action upon meeting the criterias.

Process Definition Detail

[Edit](#) [Clone](#) [Deactivate](#)

Active ☒

Process Name	prospect	Next Automated Approver Determined By
Unique Name	prospect	
Description		
Entry Criteria	(Opportunity: Stage EQUALS Negotiation/Review) AND (Opportunity: Amount GREATER THAN 100000)	
Record Editability	Administrator ONLY	Allow Submitters to Recall Approval Requests <input type="checkbox"/>
Approval Assignment Email Template	SALES_Opportunity_Needs Approval	
Initial Submitters	Opportunity Owner	
Created By	Nithi nysa, 2020-06-04, 9:42 a.m.	Modified By Nithi nysa, 2020-06-05, 3:36 a.m.

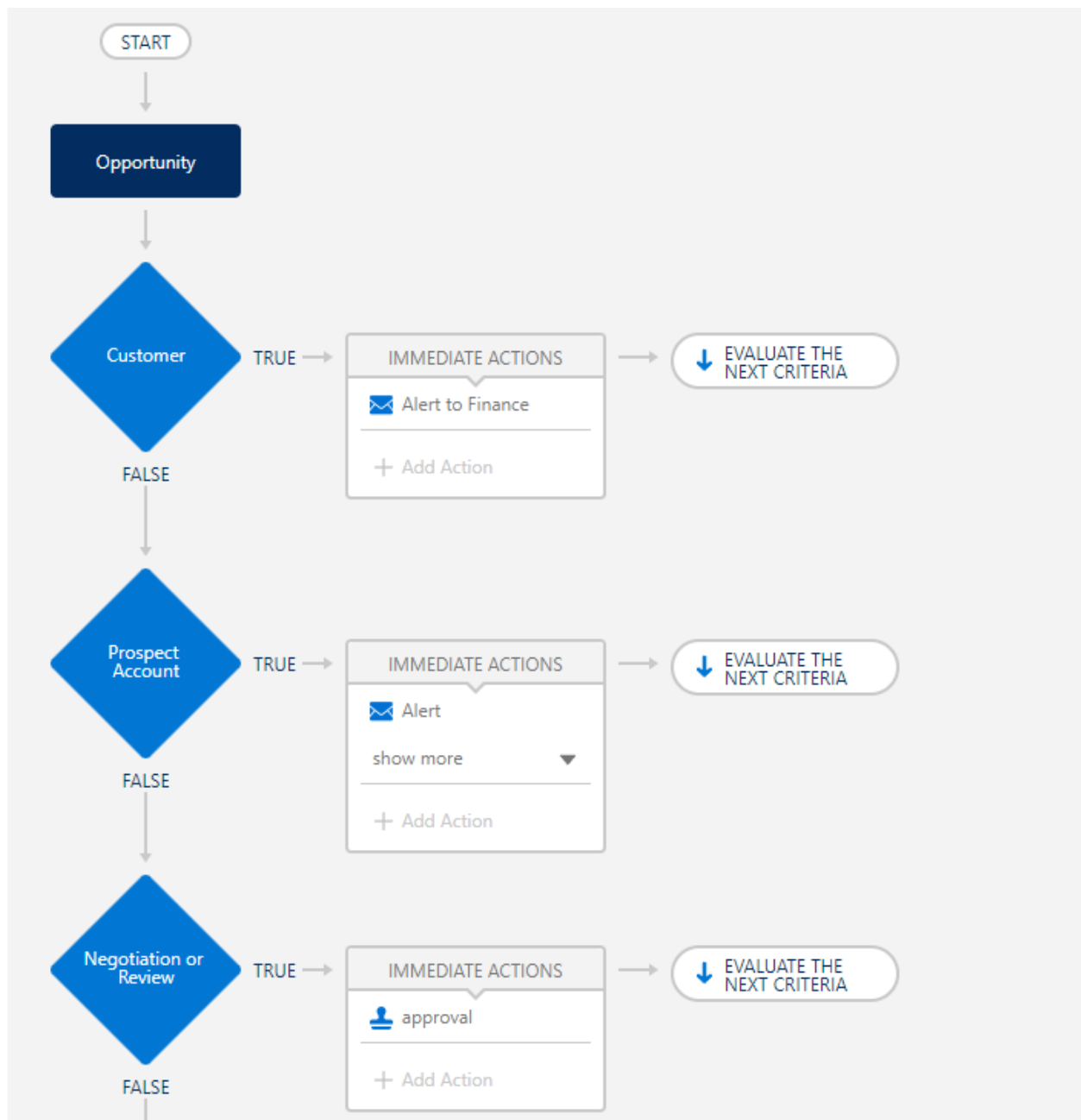
Initial Submission Actions 1			Add Existing Add New
Action	Type	Description	
	Record Lock	Lock the record from being edited	
Edit Remove	Field Update	approval update	

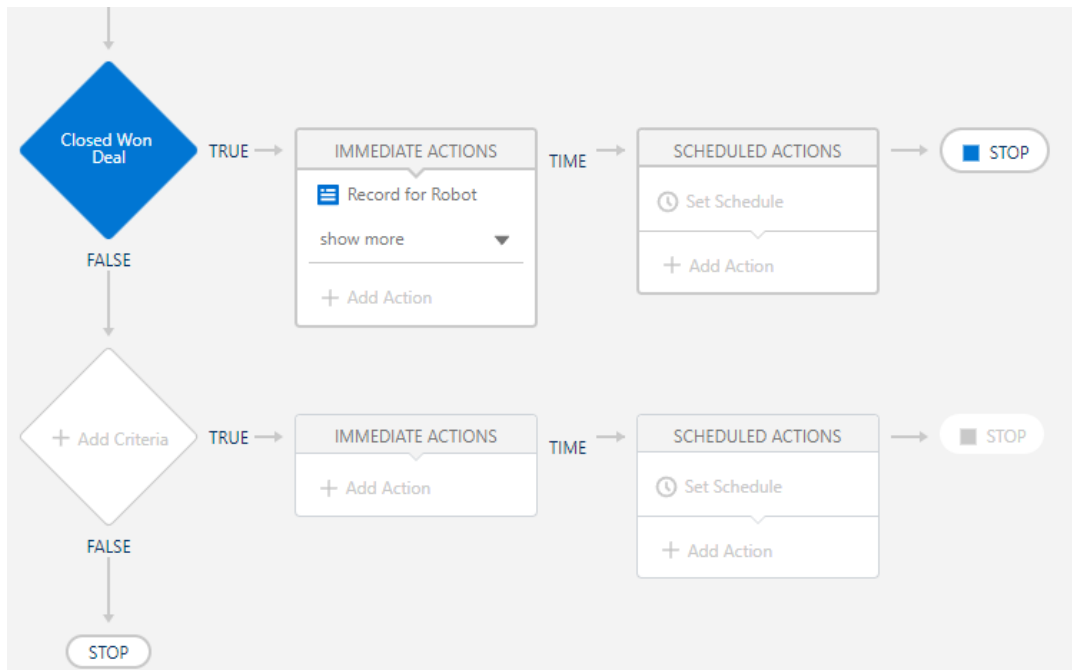
Approval Steps 1						
Action	Step Number	Name	Description	Criteria	Assigned Approver	Reject Behavior
Show Actions Edit 1						
		Step 1			User:Nithi Prasad	Final Rejection

Final Approval Actions 1			Add Existing Add New
Action	Type	Description	
Edit	Record Lock	Lock the record from being edited	
Edit Remove	Field Update	won deal	

Final Rejection Actions 1			Add Existing Add New
Action	Type	Description	
Edit	Record Lock	Unlock the record for editing	
Edit Remove	Field Update	reject	

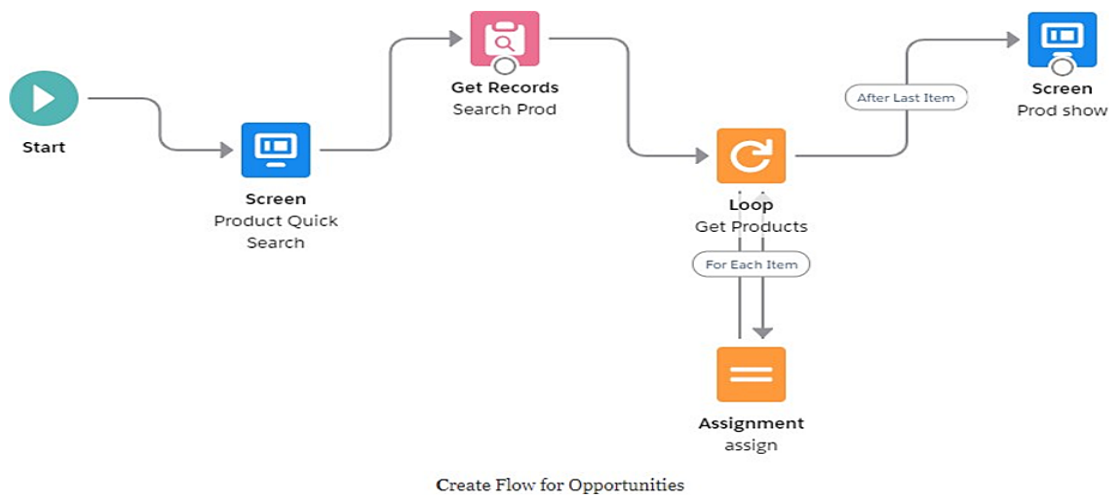
Recall Actions 1			Add Existing Add New
Action	Type	Description	
	Record Lock	Unlock the record for editing	





Challenge 7:

In this challenge we are creating Flow for Opportunities, First with a Start element then Screen element where it then gets Records and there's a loop to get each record and after that the process ends with a screen element where it shows the products. The products are created as per given in the challenge instructions to successfully complete the challenge.



Challenge 8:

It is the last challenge of the superbade where we Automate Setups, First we have to change the formula in one of the fields of the Robot object where the Formula will be given below and then we have go to the flows processthat we created previously and clone it to makechanges where we change the formula for the last criteria to Automate setups according to dates.

Formula 1:

```

Case ( WEEKDAY( Datec ),
1,"Sunday",
2,"Monday",
3,"Tuesday",
",
4,"Wednesday",
5,"Thursday",
6,"Friday",
7,"Saturday",
Text(WEEKDay(Date_c)))

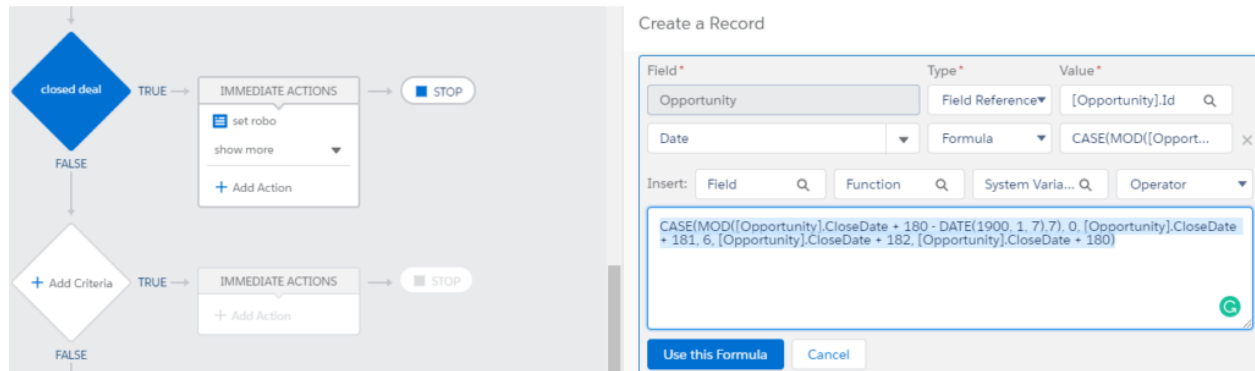
```

Formula 2:

```

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

```



And with this you will have successfully completed this Superbadge.

Apex Triggers

Get Started with Apex Triggers:

Apex trigger:

```
trigger AccountAddressTrigger on Account (before insert,before update){
```

```
List<Account> acclst=new List<Account>();
for(account a:trigger.new){
    if(a.Match_Billing_Address_c==true && a.BillingPostalCode!=null){
        a.ShippingPostalCode=a.BillingPostalCode;
    }
}
```

```
}
```

```
}
```

Bulk Apex Triggers:

Apex Trigger:

```
trigger ClosedOpportunityTrigger on Opportunity (after insert, after update) {
```

```
    List<Task> tasks = new List<Task>();
    for(Opportunity opp : [SELECT Id, StageName FROM Opportunity WHERE
        StageName='ClosedWon'
            AND Id IN :Trigger.new]){
        tasks.add(new Task(Subject = 'Follow Up Test Task' , WhatId = opp.Id));
    }

    if(tasks.size() >
        0){insert
        tasks;
    }
}
```

Apex Testing

Get Started with Apex Unit Tests:

Apex class:

```
@isTest
private class TestVerifyDate {
    @isTest static void testWithin30Days() {
        Date Datetest = VerifyDate.CheckDates(System.today(), System.today()+10);
        System.assertEquals(System.today()+10, Datetest);
    }

    @isTest static void testSetEndOfMonth() {
        Date Datetest = VerifyDate.CheckDates(System.today(), System.today()+52);
        System.assertEquals(System.today()+27, Datetest); <!--27days until last day of Current
Month-->
    }

}
```

Test Apex Triggers:

Apex Class:

```
@isTest
private class TestRestrictContactByName {

    static testMethod void metodoTest()

    {

        List<Contact> listContact= new List<Contact>();
        Contact c1 = new Contact(FirstName='Francesco', LastName='Riggio' ,
email='Test@test.com');
        Contact c2 = new Contact(FirstName='Francesco1', LastName =
'INVALIDNAME',email='Test@test.com');
        listContact.add(c1);
        listContact.add(c2);

        Test.startTest();
        try
        {
            insert listContact;
        }
        catch(Exception ee)
        {
        }

        Test.stopTest();

    }

}
```

CreateTest Data for Apex Tests:

Apex Class:

```
public with sharing class RandomContactFactory
{
    public static List<Contact> generateRandomContacts( Integer noOfContacts, String
lastName )
    {
        List<Contact> contacts = new List<Contact>();

        for( Integer i = 0; i < noOfContacts; i++ )
        {
            Contact con = new Contact( FirstName = 'Test '+i, LastName = lastName
);

            contacts.add( con );

        }

        return contacts;
    }
}
```

Asynchronous Apex

Use Future Methods:

Apex class:

```
public class
AccountProcessor{
```

```

@future
public static void countContacts(List<Id>
accountIds){ List<Account> vAccountList = new
List<Account>(); List<Account> acc = [SELECT
Id,Name,
(SELECT Id,Name FROM Contacts)
FROM Account WHERE Id IN
:accountIds]; System.debug('total contact in
Account:' + acc);

if(acc.size() > 0){
for(Account a:
acc){
List<Contact> con = [SELECT Id,Name FROM Contact WHERE accountId = :a.Id];
a.Number_of_Contacts__c = con.size();
vAccountList.add(a);
}
if(vAccountList.size()>0)
{
update vAccountList;
}
}
}
}
}

```

Test Class:

=====

```

@isTest
public class AccountProcessorTest {
    @isTest      public      staticvoid
    testNoOfContacts(){ Account a = new
    Account(Name = 'Acme1');Inserta;
    Account b = new Account(Name = 'Acme2');
    insertb;
    Contact c = new Contact(FirstName = 'Gk', LastName = 'Gupta', accountId = a.Id);

```

```

insert c;
Contact c1 = new Contact(FirstName = 'Gk1', LastName = 'Gupta1', accountId = b.Id);
insert c1;

List<account> acnt = [SELECT Id FROM Account WHERE Name = :a.Name OR Name =
:b.Name];
System.debug('size of acnt: ' +
acnt); List<ID> acntIDLST = new
List<ID>(); for(Account ac: acnt){
    acntIDLST.add(ac.Id);
}
Test.startTest();
AccountProcessor.countContacts(acntIDLST);
Test.stopTest();
}
}

```

Use Batch Apex:

Apex Class:

```

global class LeadProcessor implements Database.Batchable<Subject>
{
    global Database.QueryLocator start(Database.BatchableContext bc)
    {
        return Database.getQueryLocator([Select LeadSourceFrom Lead ]);
    }

    global void execute(Database.BatchableContext bc, List<Lead> scope)
    {
        for (Lead Leads : scope)
        {
            Leads.LeadSource = 'Dreamforce';
        }
        update scope;
    }
}

```



```

    }

    global void finish(Database.BatchableContext bc){ }
}

@isTest
public class LeadProcessorTest
{
    static testMethod void testMethod1()
    {
        List<Lead> lstLead = new List<Lead>();
        for(Integer i=0;i <200;i++)
        {
            Lead led = new Lead();
            led.FirstName ='FirstName';
            led.LastName ='LastName'+i;
            led.Company ='demo'+i;
            lstLead.add(led);
        }

        insert

        lstLead;

        Test.startTe

        st();

        LeadProcessor obj = new LeadProcessor();
        DataBase.executeBatch(obj);

        Test.stopTest();
    }
}

```

Control Processes with Queueable Apex:

Apex Class:

```
public class AddPrimaryContact implements Queueable
{
    private Contact
    c;private String
    state;

    public AddPrimaryContact(Contact c, String state)
    {
        this.c = c;
        this.state =
        state;
    }
    public void execute(QueueableContext context)
    {
        List<Account> ListAccount = [SELECT ID, Name ,(Select id,FirstName,LastName from
contacts) FROM ACCOUNT WHERE BillingState = :state LIMIT 200];
        List<Contact> lstContact = new
        List<Contact>(); for (Account acc:ListAccount)
        {
            Contact cont = c.clone(false,false,false,false);
            cont.AccountId = acc.id;
            lstContact.add( cont );
        }

        if(lstContact.size() >0 )
        {
            insert lstContact;
        }

    }
}
```

```

@isTest
public class AddPrimaryContactTest
{
    @isTest static void TestList()
    {
        List<Account> Teste = new List <Account>();
        for(Integer i=0;i<50;i++)
        {
            Teste.add(new Account(BillingState = 'CA', name = 'Test'+i));
        }
        for(Integer j=0;j<50;j++)
        {
            Teste.add(new Account(BillingState = 'NY', name = 'Test'+j));
        }
        insert Teste;

        Contact co = new
        Contact();co.FirstName=
        'demo'; co.LastName
        ='demo';insert co;
        String state = 'CA';

        AddPrimaryContact apc = new AddPrimaryContact(co, state);
        Test.startTest();
        System.enqueueJob(apc);
        Test.stopTest();
    }
}

```

Schedule Jobs Using the Apex Scheduler:

Apex Class:

global class DailyLeadProcessor implements

Schedulable {globalvoid

execute(SchedulableContext ctx){

List<Lead> lList = [Select Id, LeadSource from Lead where LeadSource = null];

if(!lList.isEmpty()){

for(Lead l: lList){

l.LeadSource = 'Dreamforce';

}

update lList;

}

}

}

@isTest

public class DailyLeadProcessorTest {

public static String CRON_EXP = '0 0 0 15 3 ? 2022';

static testMethod void testDailyLeadProcessorTest()

{

List<Lead> listLead = new

List<Lead>(); for (Integer i=0; i<200;

i++){

Lead ll = new Lead();

ll.LastName = 'Test' + i;

ll.Company = 'Company'

+ i;

ll.Status = 'Open - Not

Contacted';listLead.add(ll);

}

insert listLead;

Test.startTest();

DailyLeadProcessor daily = new DailyLeadProcessor();

```

String jobId = System.schedule('Update LeadSource to Dreamforce', CRON_EXP, daily);

List<Lead> liss = new List<Lead>([SELECT Id, LeadSource FROM Lead WHERE
LeadSource != 'Dreamforce']);
Test.stopTest();
}
}

```

Apex Integration Services

Apex Rest Callouts:

Apex Class:

```

public class AnimalLocator {
    public static String getAnimalNameById(Integer
    id) {Http http = new Http();
    HttpRequest request = new HttpRequest();
    request.setEndpoint('https://th-apex-http-callout.herokuapp.com/animals/'+id);
    request.setMethod('GET');
    HttpResponse response = http.send(request);
    /*Map<String,Object> results =
(Map<String,Object>)JSON.deserializeUntyped(response.getBody());
    system.debug('--- >results'+results);
    List<Object> animals = (List<Object>)
    results.get('animal'); system.debug('-----
>animal'+animals);*/
    Map<Integer,String> mapAnimal = new Map<Integer,String>();

    Integer
    varId; String
    varName;
    JSONParser parser1= JSON.createParser(response.getBody());

```

```

while (parser1.nextToken() != null) {
    if ((parser1.getCurrentToken() == JSONToken.FIELD_NAME) && (parser1.getText() ==
'id')) {

}

/ Get the value.
parser1.nextToken();
/ Fetch the ids for all animals in JSON Response.
varId=parser1.getIntegerValue();
System.debug('-->varId-->'+varId);
parser1.nextToken();

    if ((parser1.getCurrentToken() == JSONToken.FIELD_NAME) && (parser1.getText() ==
'name')){
        parser1.nextToken();
        / Fetch the names for all animals in JSON
        Response.varName=parser1.getText();
        System.debug('-->varName-->'+varName);
    }
    mapAnimal.put(varId,varName);
}
system.debug('-->mapAnimal-->'+mapAnimal);
return mapAnimal.get(id);

}
}

```

Mock Test Class:

@isTest

```

global class AnimalLocatorMock implements HttpCalloutMock {
    / Implement this interfacemethod
    global HTTPResponse respond(HTTPRequest request){
        / Create a fake response
        HTTPResponse response = new HTTPResponse();
        response.setHeader('Content-Type',
            'application/json');
        response.setBody('{"animal":[{"id":1,"name":"chicken","eats":"chicken food","says":"cluck
cluck"}, {"id":2,"name":"duck","eats":"worms","says":"pek pek"}]}');
        response.setStatusCode(200);
        return response;
    }
}

```

Test Class:

```

@isTest
private class AnimalLocatorTest {
    @isTest static void testGetCallout() {
        / Set mock calloutclass
        Test.setMock(HttpCalloutMock.class, new AnimalLocatorMock());
        / This causes a fake responseto be sent
        / from the class that implements HttpCalloutMock.
        String response =
        AnimalLocator.getAnimalNameById(1);
        system.debug('Test Response1--->'+response);
        String expectedValue = 'chicken';
        System.assertEquals(expectedValue,response);
        String response2= AnimalLocator.getAnimalNameById(2);
        system.debug('Test Response2--->'+response2);
        String expectedValue2 = 'duck';
        System.assertEquals(expectedValue2,response2);
    }
}

```

Apex SOAP Callouts:

Apex Class:

Service:

/ Generatedby wsdl2apex

```
public class ParkService {
    public class byCountryResponse {
        public String[] return_x;
        private String[] return_x_type_info = new
String[]{'return','http://parks.services/',null,'0','- 1','false'};
        private String[] apex_schema_type_info = new
String[]{'http://parks.services/',false,false'}; private String[] field_order_type_info =
        new String[]{'return_x'};
    }
    public class
        byCountry{public
        Stringarg0;
        private String[] arg0_type_info = new
String[]{'arg0','http://parks.services/',null,'0','1','false'}; private String[]
        apex_schema_type_info = new String[]{'http://parks.services/',false,false'};

        private String[] field_order_type_info = new String[]{'arg0'};
    }
    public class ParksImplPort {
        public String endpoint_x = 'https://th-apex-soap-
service.herokuapp.com/service/parks'; public Map<String,String>
        inputHttpHeaders_x;
        public Map<String,String>
        outputHttpHeaders_x; public String
        clientCertName_x;
        public String clientCert_x;
        public String clientCertPasswd_x;
        public Integertimeout_x;
        private String[] ns_map_type_info = new String[]{'http://parks.services/',
'ParkService'}; public String[]byCountry(String arg0) {
            ParkService.byCountry request_x = new ParkService.byCountry();
```



```

        request_x.arg0 = arg0;
        ParkService.byCountryResponse response_x;
        Map<String, ParkService.byCountryResponse> response_map_x = new Map<String,
ParkService.byCountryResponse>();
        response_map_x.put('response_x', response_x);
        WebServiceCallout.invoke(
            this,
            reque
            st_x,
            response_map_x,
            new
            String[]{endpoint_x,
            ",
            'http://parks.services/',
            'byCountry',
            'http://parks.services/',
            'byCountryResponse',
            'ParkService.byCountryResponse'}
        );
        response_x = response_map_x.get('response_x');
        return response_x.return_x;
    }
}
}

```

Class:

```

public class ParkLocator {
    public static String[] country(String country){
        ParkService.ParksImplPort parks = new
        ParkService.ParksImplPort(); String[] parksname =
        parks.byCountry(country);

        return parksname;
    }
}

```

Test:

@isTest

```

private class ParkLocatorTest{
    @isTest
    static void testParkLocator() {
        Test.setMock(WebServiceMock.class, new
        ParkServiceMock()); String[] arrayOfParks =
        ParkLocator.country('India');

        System.assertEquals('Park1', arrayOfParks[0]);
    }
}

```

Mock Test:

```

@isTest
global class ParkServiceMock implements
WebServiceMock {global void doInvoke(
    Object
    stub,
    Object
    request,
    Map<String, Object> response,
    String endpoint,
    String soapAction,
    String
    requestName,
    String
    responseNS,
    String
    responseName,
    String
    responseType) {
    ParkService.byCountryResponse response_x = new ParkService.byCountryResponse();
    List<String> lstOfDummyParks = new List<String> {'Park1','Park2','Park3'};
    response_x.return_x = lstOfDummyParks;

    response.put('response_x', response_x);
}
}

```

Apex Web Services:

Apex Class:

```
@RestResource(urlMapping='/Accounts/*/contacts') global with sharing class AccountManager{
    @HttpGet
    global static Account getAccount(){
        RestRequest request =
        RestContext.request;
        String accountId = request.requestURI.substringBetween('Accounts/','/contacts');
        system.debug(accountId);
        Account objAccount = [SELECT Id,Name,(SELECT Id,Name FROM Contacts) FROM Account
WHERE Id = :accountId LIMIT 1];
        return objAccount;
    }
}
```

```
/ Test
class
@Test
private class AccountManagerTest{
    static testMethod void
    testMethod1(){
        Account objAccount = new Account(Name = 'test Account');
        insert objAccount;
        Contact objContact = new Contact(LastName = 'test Contact',
            AccountId = objAccount.Id);
        insert objContact;
        Id recordId = objAccount.Id;
        RestRequest request = new RestRequest();
        request.requestUri =
            'https://sandeepidentity-dev-ed.my.salesforce.com/services/apexrest/Accounts/'
            + recordId + '/contacts';
        request.httpMethod = 'GET';
    }
}
```

```

    RestContext.request = request;
    / Call the method to test
    Account thisAccount = AccountManager.getAccount();
    / Verify results
    System.assert(thisAccount!=
    null);
    System.assertEquals('test Account', thisAccount.Name);
  }
}

```

Lightning Web Components

Deploy LightningWeb Component Files:

bikeCard.html:

```

<template>
  <div>
    <div>Name: {name}</div>
    <div>Description: {description}</div>
    <lightning-badge label={material}></lightning-badge>
    <lightning-badge label={category}></lightning-badge>
    <div>Price: {price}</div>
    <div><img src={pictureUrl}/></div>
  </div>
</template>

```

bikeCard.js:

```

import { LightningElement } from 'lwc';
export default class BikeCard extends LightningElement {
  name = 'Electra X4';
  description = 'A sweet bike built for comfort.';
  category = 'Mountain';
  material =

```

```
'Steel'; price  
= '$2,700';  
pictureUrl = 'https://s3-us-west-1.amazonaws.com/sfdc-demo/ebikes/electrax4.jpg';  
}
```

bikeCard.js-meta.xml:

```
<?xml version="1.0" encoding="UTF-8"?>  
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">  
  <!-- The apiVersion may need to be increased for the current release -->  
  <apiVersion>52.0</apiVersion>  
  <isExposed>true</isExposed>  
  <masterLabel>Product Card</masterLabel>  
  <targets>  
    <target>lightning_AppPage</target>  
    <target>lightningRecordPage</target>  
    <target>lightning_HomePage</target>  
  </targets>  
</LightningComponentBundle>
```