APEX SPECIALISTSUPERBADGE-SOLUTIONS

Automated Record

CreationMaintenanceRequestHelper.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 :ValidIdsGROUPBY 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.Vehiclec, 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);
  }
}
```

Synchronize Salesforce data with an externalsystem

WarehouseCalloutService.apxc :-

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

   //class that makes a REST calloutto an external warehouse system to get a list of
equipment that needs to be updated.
   //The callout'sJSON response returnsthe equipment recordsthat you upsert in
Salesforce.

   @future(callout=true)
   public staticvoid 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 followingfields: replacement part (always true), cost,
currentinventory, lifespan, maintenance cycle, and warehouseSKU
         //warehouse SKU will be external ID for identifying which equipment records
```

toupdate 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 staticvoid execute (QueueableContext context){
    runWarehouseEquipmentSync();
  }


}
```

After savingthe code open execute anonymous window (CTRl+E) and run thismethod ,

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

Schedule synchronization using Apex

codeWarehouseSyncShedule.apxc :-

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

Test automation logic

MaintenanceRequestHelperTest.apxc:-

```
@istest
public with sharing class MaintenanceRequestHelperTest {

   private staticfinal string STATUS_NEW = 'New';
   privatestatic final stringWORKING = 'Working';
   private static final string CLOSED =
   'Closed';private static final string REPAIR =
   'Repair';
   private staticfinal string REQUEST_ORIGIN = 'Web';
   private static final stringREQUEST_TYPE = 'RoutineMaintenance';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 STATICCase 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(idequipmentId,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;

idvehicleId = vehicle.Id;

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

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

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

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

CasenewReq = [Select id, subject,type, Equipment__c, Date_Reported__c,
Vehicle__c, Date_Due__c
        fromcase
        where status =:STATUS_NEW];

Equipment_Maintenance_Item__c workPart= [select id
                fromEquipment_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();
    insertvehicle;
    id vehicleId= vehicle.Id;

    product2 equipment= createEq();
    insertequipment;
    id equipmentId = equipment.Id;

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

    Equipment_Maintenance_Item__c workP = createWorkPart(equipmentId,
emptyReq.Id);
    insert workP;

    test.startTest();
    emptyReq.Status = WORKING;
    updateemptyReq;
    test.stopTest();

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

    Equipment_Maintenance_Item__c workPart= [select id
                          fromEquipment_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(r
            eq.Id);
        }
        update requestList;
        test.stopTest();

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

        list<Equipment_Maintenance_Item__c> workParts = [select id
                                                         fromEquipment_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 :ValidIdsGROUPBY 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.Vehiclec, 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 (beforeupdate, after update){if(Trigger.isUpdate
```

```
    &&Trigger.isAfter){
        MaintenanceRequestHelper.updateWorkOrders(Trigger.New,  Trigger.OldMap);
    }
}
```

Test calloutlogic

WarehouseCalloutService.apxc :-

```
public with sharing classWarehouseCalloutService {
    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();
    // implementmock callout test here
    Test.setMock(HTTPCalloutMock.class, new WarehouseCalloutServiceMock());
```

```
        WarehouseCalloutService.runWarehouseEquipmentSync();
        Test.stopTest();
        System.assertEquals(1, [SELECTcount() 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;
    }
}
```

Test scheduling logic

WarehouseSyncSchedule.apxc :-

```
global classWarehouseSyncSchedule implements Schedulable {
   global void execute(SchedulableContext ctx) {

      WarehouseCalloutService.runWarehouseEquipmentSync();
   }
}
```

WarehouseSyncScheduleTest.apxc :-

```
@isTest
public class WarehouseSyncScheduleTest {


   @isTest static void WarehousescheduleTest(){
      StringscheduleTime = '00 00 01 * * ?';
      Test.startTest();
      Test.setMock(HttpCalloutMock.class, new WarehouseCalloutServiceMock());
      StringjobID=System.schedule('Warehouse Time To Schedule to Test',
scheduleTime, new WarehouseSyncSchedule());
      Test.stopTest();
      //Contains scheduleinformation for a scheduled job. CronTrigger is similar to a
cronjob on UNIX systems.
      // This object is availablein API version 17.0 and later.
      CronTrigger a=[SELECTId FROM CronTrigger where NextFireTime > today];
      System.assertEquals(jobID, a.Id,'Schedule ');


   }
}
```

PROCESS AUTOMATION SPECIALIST - SUPERBADGE

Automata leads

error condition formula

OR(AND(LEN(State) > 2,
NOT(CONTAINS("AL:AK:AZ:AR:CA:CO:CT:DE:DC:FL:GA:HI:ID:IL:IN:IA:KS:KY:LA:ME:MD
: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", State))), NOT(OR(Country ="US",Country ="USA",Country
="UnitedStates", ISBLANK(Country))))

Automate accounts

error condition formula1

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))))

error conditionformula1

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

Automata

stepsformula

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