### **Apex Specialist Superbadge-**

**Automate record creation:**

**2.MaintenanceRequestHelper-**

public with sharing class MaintenanceRequestHelper {

public static void updateworkOrders(List<Case> updWorkOrders, Map<Id,Case> nonUpdCaseMap) {

Set<Id> validIds = new Set<Id>();

For (Case c : updWorkOrders){

if (nonUpdCaseMap.get(c.Id).Status != 'Closed' && c.Status == 'Closed'){

if (c.Type == 'Repair' || c.Type == 'Routine Maintenance'){

validIds.add(c.Id);

}

}

}

if (!validIds.isEmpty()){

List<Case> newCases = new List<Case>();

Map<Id,Case> closedCasesM = new Map<Id,Case>([SELECT Id, Vehicle\_\_c, Equipment\_\_c, Equipment\_\_r.Maintenance\_Cycle\_\_c,(SELECT Id,Equipment\_\_c,Quantity\_\_c FROM Equipment\_Maintenance\_Items\_\_r)

FROM Case WHERE Id IN :validIds]);

Map<Id,Decimal> maintenanceCycles = new Map<ID,Decimal>();

AggregateResult[] results = [SELECT Maintenance\_Request\_\_c, MIN(Equipment\_\_r.Maintenance\_Cycle\_\_c)cycle FROM Equipment\_Maintenance\_Item\_\_c WHERE Maintenance\_Request\_\_c IN :ValidIds GROUP BY Maintenance\_Request\_\_c];

for (AggregateResult ar : results){

maintenanceCycles.put((Id) ar.get('Maintenance\_Request\_\_c'), (Decimal) ar.get('cycle'));

}

for(Case cc : closedCasesM.values()){

Case nc = new Case (

ParentId = cc.Id,

Status = 'New',

Subject = 'Routine Maintenance',

Type = 'Routine Maintenance',

Vehicle\_\_c = cc.Vehicle\_\_c,

Equipment\_\_c =cc.Equipment\_\_c,

Origin = 'Web',

Date\_Reported\_\_c = Date.Today()

);

If (maintenanceCycles.containskey(cc.Id)){

nc.Date\_Due\_\_c = Date.today().addDays((Integer) maintenanceCycles.get(cc.Id));

} else {

nc.Date\_Due\_\_c = Date.today().addDays((Integer) cc.Equipment\_\_r.maintenance\_Cycle\_\_c);

}

newCases.add(nc);

}

insert newCases;

List<Equipment\_Maintenance\_Item\_\_c> clonedWPs = new List<Equipment\_Maintenance\_Item\_\_c>();

for (Case nc : newCases){

for (Equipment\_Maintenance\_Item\_\_c wp : closedCasesM.get(nc.ParentId).Equipment\_Maintenance\_Items\_\_r){

Equipment\_Maintenance\_Item\_\_c wpClone = wp.clone();

wpClone.Maintenance\_Request\_\_c = nc.Id;

ClonedWPs.add(wpClone);

}

}

insert ClonedWPs;

}

}

}

MaintenanceRequest:

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

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

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

}

}

**3.Synchronize Salesforce data with an external system:**

**WarehouseCalloutService:**

**public with sharing class WarehouseCalloutService implements Queueable {**

**private static final String WAREHOUSE\_URL = 'https://th-superbadge-apex.herokuapp.com/equipment';**

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

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

**@future(callout=true)**

**public static void runWarehouseEquipmentSync(){**

**Http http = new Http();**

**HttpRequest request = new HttpRequest();**

**request.setEndpoint(WAREHOUSE\_URL);**

**request.setMethod('GET');**

**HttpResponse response = http.send(request);**

**List<Product2> warehouseEq = new List<Product2>();**

**if (response.getStatusCode() == 200){**

**List<Object> jsonResponse = (List<Object>)JSON.deserializeUntyped(response.getBody());**

**System.debug(response.getBody());**

**//class maps the following fields: replacement part (always true), cost, current inventory, lifespan, maintenance cycle, and warehouse SKU**

**//warehouse SKU will be external ID for identifying which equipment records to update within Salesforce**

**for (Object eq : jsonResponse){**

**Map<String,Object> mapJson = (Map<String,Object>)eq;**

**Product2 myEq = new Product2();**

**myEq.Replacement\_Part\_\_c = (Boolean) mapJson.get('replacement');**

**myEq.Name = (String) mapJson.get('name');**

**myEq.Maintenance\_Cycle\_\_c = (Integer) mapJson.get('maintenanceperiod');**

**myEq.Lifespan\_Months\_\_c = (Integer) mapJson.get('lifespan');**

**myEq.Cost\_\_c = (Integer) mapJson.get('cost');**

**myEq.Warehouse\_SKU\_\_c = (String) mapJson.get('sku');**

**myEq.Current\_Inventory\_\_c = (Double) mapJson.get('quantity');**

**myEq.ProductCode = (String) mapJson.get('\_id');**

**warehouseEq.add(myEq);**

**}**

**if (warehouseEq.size() > 0){**

**upsert warehouseEq;**

**System.debug('Your equipment was synced with the warehouse one');**

**}**

**}**

**}**

**public static void execute (QueueableContext context){**

**runWarehouseEquipmentSync();**

**}**

**}**

**4.Schedule synchronization:**

**WareHouseSyncSchedule:**

**global with sharing class WarehouseSyncSchedule implements Schedulable{**

**global void execute(SchedulableContext ctx){**

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

**}**

**}**

**5.Test automation logic:**

**@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:**

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

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

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

**}**

6.Test Callout Logic

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);

}

}

}

}

@isTest

global class WarehouseCalloutServiceMock implements HttpCalloutMock {

// implement http mock callout

global static HttpResponse respond(HttpRequest request){

System.assertEquals('https://th-superbadge-apex.herokuapp.com/equipment', request.getEndpoint());

System.assertEquals('GET', request.getMethod());

// Create a fake response

HttpResponse response = new HttpResponse();

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

response.setBody('[{"\_id":"55d66226726b611100aaf741","replacement":false,"quantity":5,"name":"Generator 1000 kW","maintenanceperiod":365,"lifespan":120,"cost":5000,"sku":"100003"}]');

response.setStatusCode(200);

return response;

}

}

@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]);

}

}

7.Testing Scheduling Logic-

Warehousesyncscheduletest-

@isTest

public class WarehouseSyncScheduleTest {

@isTest static void WarehousescheduleTest(){

String scheduleTime = '00 00 01 \* \* ?';

Test.startTest();

Test.setMock(HttpCalloutMock.class, new WarehouseCalloutServiceMock());

String jobID=System.schedule('Warehouse Time To Schedule to Test', scheduleTime, new WarehouseSyncSchedule());

Test.stopTest();

//Contains schedule information for a scheduled job. CronTrigger is similar to a cron job on UNIX systems.

// This object is available in API version 17.0 and later.

CronTrigger a=[SELECT Id FROM CronTrigger where NextFireTime > today];

System.assertEquals(jobID, a.Id,'Schedule ');

}

}

WarehouseSyncSchedule-

global class WarehouseSyncSchedule implements Schedulable {

global void execute(SchedulableContext ctx) {

WarehouseCalloutService.runWarehouseEquipmentSync();

}

}