APEX SPECIALIST SUPERBADGE-SOLUTIONS Automated Record Creation MaintenanceRequestHelper.apxc public with sharing class MaintenanceRequestHelper { public static void updateworkOrders(List updWorkOrders, Map nonUpdCaseMap) { Set validIds = new Set(); 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 newCases = new List(); Map closedCasesM = new Map([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 maintenanceCycles = new Map(); 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 clonedWPs = new List(); 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 external system WarehouseCalloutService.apxc :- public with sharing class WarehouseCalloutService implements Queueable { private static final String WAREHOUSE_URL = 'https://th-superbadgeapex.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 warehouseEq = new List(); if (response.getStatusCode() == 200){ List jsonResponse = (List)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 mapJson = (Map)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()); Schedule synchronization using Apex code WarehouseSyncShedule.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 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 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 vehicleList = new list(); list equipmentList = new list(); list workPartList = new list(); list requestList = new list(); list oldRequestIds = new list(); 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 allRequests =

[select id from case where status =: STATUS_NEW]; list 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 updWorkOrders, Map nonUpdCaseMap) { Set validIds = new Set(); 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 newCases = new List(); Map closedCasesM = new Map([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 maintenanceCycles = new Map(); 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 clonedWPs = new List(); 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); } } Test callout logic WarehouseCalloutService.apxc :- public with sharing class WarehouseCalloutService { private static final String WAREHOUSE_URL = 'https://th-superbadgeapex.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 warehouseEq = new List(); if (response.getStatusCode() == 200){ List jsonResponse = (List)JSON.deserializeUntyped(response.getBody());

System.debug(response.getBody()); for (Object eq : jsonResponse){ Map mapJson = (Map)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; } } Test scheduling logic 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 '); } } 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: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)))) 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: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))),NOT(OR(BillingCountry ="US",BillingCountry ="USA",BillingCountry ="United States", ISBLANK(BillingCountry))), NOT(OR(ShippingCountry ="US",ShippingCountry ="USA",ShippingCountry ="United States", ISBLANK(ShippingCountry)))) error condition formula1 ISCHANGED(Name) && (OR(ISPICKVAL(Type ,'Customer - Direct') ,ISPICKVAL(Type ,'Customer - Channel'))) Automata steps formula Case (WEEKDAY(Date__c), 1,"Sunday", 2,"Monday", 3,"Tuesday", 4,"Wednesday", 5,"Thursday", 6,"Friday", 7,"Saturday", Text(WEEKDay(Date__c))