# Apex Specialist Super Badge

1. At first create a new Trailhead Playground.
2. Install a package How We Roll Maintenance(package ID: 04t6g000008av9iAAA)which contain metadata for the completion of superbadge.
3. Add two picklist values Repair and Routine Maintenance to Type field on case Object. 4.Update Case page layout assignment to Case(HowWeRoll)Layout for your profile.

5.Rename tab/label for case tab to Maintenance Request. 6.Update the Product page Layout assignment to Product(HowWeRol)Layout for profile.

1. Rename the tab of product to Equipment.
2. Use App Launcher to navigate to the **Create Default Data** tab of the **How We Roll Maintenance** app. Click **Create Data**
3. Standard objects used are Maintenance Request and Equipment. 10.Custom objects used vehicle and Equipment Maintenance Item.

11.Apex classes and Triggers are used to complete this badge. 12.Trigers and classes can be created using the developer console in trailhead.

13.The code must be executed once to complete the challenge in the superbadge.

ApexSpecialist

## Challenge-1

### MaintenanceRequestHelper.apxc

public class MaintenanceRequestHelper { public static void updateWorkOrders(){

Map<Id, Case> mantnceReqToEvaluate = new Map<Id, Case>(); for(Case mantnceReq : (List<Case>)Trigger.new){

if((mantnceReq.Type.contains('Repair') || mantnceReq.Type.contains('Routine Maintenance'))

&& mantnceReq.Status == 'Closed'){ mantnceReqToEvaluate.put(mantnceReq.Id,mantnceReq);

}

Map<Id, decimal> mapOfProdIdWithMaintenanceCycle =

getMapOfProdIdWithMaintenanceCycle(); List<Case> lstOfMaintenanceRoutines

= getListOfMaintenanceRoutineList(mantnceReqToEvaluate, mapOfProdIdWithMaintenanceCycle);

System.debug('lstOfMaintenanceRoutines :::::::: '+lstOfMaintenanceRoutines);

if(lstOfMaintenanceRoutines != null && lstOfMaintenanceRoutines.size() > 0) INSERT lstOfMaintenanceRoutines;

}

private static Map<Id, decimal> getMapOfProdIdWithMaintenanceCycle(){ Map<Id,decimal> mapOfProdIdWithMaintenanceCycle = new

Map<Id, decimal>();

for(Product2 prod : [SELECT Id, Maintenance\_Cycle c from Product2]){

mapOfProdIdWithMaintenanceCycle.put(prod.Id, prod.Maintenance\_Cycle c);

}

return mapOfProdIdWithMaintenanceCycle;

}

private static List<Case> getListOfMaintenanceRoutineList(Map<Id, Case> mantnceReqToEvaluate, Map<Id, decimal> mapOfProdIdWithMaintenanceCycle){

List<Case> lstOfMaintenanceRoutines = new List<Case>(); for(Case maintenance : mantnceReqToEvaluate.values()){

Case maintenanceNewIns = new Case(); maintenanceNewIns.Vehicle\_\_c = maintenance.Vehicle c;

maintenanceNewIns.Equipment c = maintenance.Equipment c; maintenanceNewIns.Type = 'Routine Maintenance'; maintenanceNewIns.Subject = 'Your Routine Maintenance Schedule'; maintenanceNewIns.Date\_Reported c = Date.today();

maintenanceNewIns.Date\_Due c = getDueDate(maintenance,

mapOfProdIdWithMaintenanceCycle); maintenanceNewIns.Status = 'New'; maintenanceNewIns.Origin = 'Phone'; lstOfMaintenanceRoutines.add(maintenanceNewIns);

return lstOfMaintenanceRoutines;

}

private static Date getDueDate(Case maintenance, Map<Id, decimal> mapOfProdIdWithMaintenanceCycle){

Date dt = null; if

(mapOfProdIdWithMaintenanceCycle.get(maintenance.Equipment c) != null) { dt =

Date.today().addDays(Integer.valueOf(mapOfProdIdWithMaintenanceC ycle.get(maintenance.Equipment c)));

}

return dt;

}

}

### MaintetanceRequest.apxt

trigger MaintenanceRequest on Case (before update, after update)

{

// ToDo: Call MaintenanceRequestHelper.updateWorkOrders MaintenanceRequestHelper.updateWorkOrders();

}

## Challenge2

### WarehouseCalloutService.apxc

public with sharing class WarehouseCalloutService { private static final String WAREHOUSE\_URL = 'https://thsuperbadge-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);

// If the request is successful, parse the JSON response. if (response.getStatusCode() == 200) {

// Deserialize the JSON string into collections of primitive data types.

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

List<Product2> products = new List<Product2>(); for(Object o : equipments){

Map<String, Object> mapProduct = (Map<String, Object>)o;

Product2 product = new Product2(); product.Name = (String)mapProduct.get('name'); product.Cost c =

(Integer)mapProduct.get('cost'); product.Current\_Inventory c =

(Integer)mapProduct.get('quantity');

product.Maintenance\_Cycle c = (Integer)mapProduct.get('maintenanceperiod'); product.Replacement\_Part c =

(Boolean)mapProduct.get('replacement'); product.Lifespan\_Months c =

(Integer)mapProduct.get('lifespan'); product.Warehouse\_SKU c =

(String)mapProduct.get('sku'); product.ProductCode =

(String)mapProduct.get('\_id'); products.add(product);

}

if(products.size() > 0){ System.debug(products); upsert products;

}

}

}

}

### WarehouseCallutServiceMock.apxc

@isTest global class WarehouseCalloutServiceMock implements HttpCalloutMock {

// implement http mock callout

global static HttpResponse respond(HttpRequest request){ System.assertEquals('https://th-superbadgeapex.herokuapp.com/equipment', request.getEndpoint());

System.assertEquals('GET', request.getMethod()); HttpResponse response = new HttpResponse(); response.setHeader('Content-Type', 'application/json');

response.setBody('[{"\_id":"55d66226726b611100aaf741","replacemen t":false,"quantity":5,"name":"Generator 1000

kW","maintenanceperiod":365,"lifespan":120,"cost":5000,"sku":"10 0003"}]'); response.setStatusCode(200);

return response;

}

}

### WarehouseCalloutSeviceTest.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]); }

}

## Challenge3

### WarehouseSyncSchedule.apxc

global with sharing class WarehouseSyncSchedule implements Schedulable{

global void execute(SchedulableContext ctx){ System.enqueueJob(new WarehouseCalloutService());

}

}

### 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 ');

}

}

## Challenge4

### 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;

}

}

}

### 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);

}

}

## Challenge5

### 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-superbadgeapex.herokuapp.com/equipment', request.getEndpoint());

System.assertEquals('GET', request.getMethod()); HttpResponse response = new HttpResponse(); response.setHeader('Content-Type', 'application/json');

response.setBody('[{"\_id":"55d66226726b611100aaf741","replacemen t":false,"quantity":5,"name":"Generator 1000

kW","maintenanceperiod":365,"lifespan":120,"cost":5000,"sku":"10 0003"}]'); response.setStatusCode(200);

return response;

}

}

## Challenge6

### WarehouseSuncScheduleTest.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

1. Create a Trailhead Playground and install a package (package ID 04t46000001Zch4).
2. Standard objects used are Account,Contact,Opportunity. 2.Custom objects used are RobotSetup with record type with Master-DetailRelationship to an opportunity.

Fields created are:

Name Date Notes

Day of the week

### Challenge 1 Automate Leads

1.First create Validation Rule with the formula using the condition given in the transcript and then create the leads. 2.From quick find box search for assignment rule and in rule entry enter Field as Lead Source and operator as not equals to and value as web.

3. with this this challenge completes.

### challenge 2 Automate Accounts

1. Create the given fields for Account object in the description i.e., Number\_of\_deals c,Number\_of\_won\_deals c, last\_won\_deal\_Date c,Amount\_of\_won\_deals c, Deal\_win\_percent c, Call\_for\_service c
2. And then create two Validation Rules for this according to given description and give Error messages to them.

### challenge 3

**Create Robot Setup Object**

1.Atfirst create Robot Setup with master-Detail relationship to opportunity and create the fields Date,Notes,Day of the week with certain type

### challenge 4

**Create Sales Process and Validate Opportunities**

1. Add a picklist value to the Stage field in Opportunity called Awaiting Approval
2. And add a validation rule with the certain formula having Amount>100000
3. This concludes challenge.

### challenge 5

**Automate the opportunities**

1.Create three email templates called Finance:Account Creation SALES:Opportunity Needs Approval Sales:Opportunity Approval Status

these email alerts are used to create alerts when creating a process. 2.Critera must be Opporunity stage equals Negotiation/review and Opportunity ammount Greater Than 100000

1. Make sure that manager as Nushi Davoud in manage Users.
2. Create Process in the process builder for opportunity. 5.Create the nodes for all the criteria mentioned in the description

and also the email alerts.

1. Create email alert for Finance:Account creation and a record with subject as 'Send marketing materials'that was assigned to owner.
2. and a node for Approvals and also a record for Robot setup with certain Formula.

### Challenge 6

**Create flow for opportunities**

Create flow for opportunities and name it as Product Quick Search

* 1. Add Screen Component named product Quick Search and add radio components CloudyBot,Assembly System,Rainbow Bot.
  2. Add an element to it name it as get Record and label it as Search Product and object as Product.
  3. And add Display Screen at last and link all three together in certain order to build a successful flow.

### challenge 7 Automate setups

1. Goto Day of the week field which was created earlier in robot object and add the formula to it i.e.,

Case(WeekDay(Date c), 1,"Sunday",

2,"Monday",

3,"Tuesday", 4,"Wednesday", 5,"Thursday",

6,"Friday",

7,"Saturday",

Text(WeekDay(Date c))) then click save

1. Goto process builder,clone the process you made to change the formula given there to meet the business requirements as mentioned

Then click check Challenge gives the result.