Apex Specialist SuperBadge

1. Automate Record Creation

- Install the unlocked package and configure the development org.
- Use the included package content to automatically create a Routine Maintenance request of type Repair or Routine Maintenance is updated to Closed.
- Follow the specifications and naming conventions outlined in the business requirements.

Code for 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));
        }

        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;
        }
    }
}
```

Code for MaintenanceRequest

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

2. *Synchronize Salesforce data with an external system*

Implement an Apex class(class WarehouseCalloutService) that implements the queueable interface and makes a callout to the external service used for warehouse inventory management. this servise receives updated values in the external system and updates the realated records in Salesforce.Before checking this section,**enqueue the job at least once to confirm that it's working as expected**.

Code for WarehouseCalloutService

```
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 external ID SKU
        }


    }
        //Get the product list 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 response to the appropriate fields in the Equipment object
            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 response
      public static HttpResponse getResponse() {

    Http http = new Http();
    HttpRequest request = new HttpRequest();
    request.setEndpoint(WAREHOUSE_URL);
    request.setMethod('GET');
    HttpResponse response = http.send(request);

    return response;

  }
  }
```

3. Schedule Sychronization

- Build scheduling logic that executes your callout and runs your code daily. The name of the schedulable class should be WarehouseSynScheduled,and the WarehouseSyncSchedule job.

Code for WarehouseSyncSchedule

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

4. Test automation logic

Build tests for all cases(positive,negative,and bulk) specified in the business requirements by using a class named MaintenanceRequestHelperTest. You must have 100% test coverage to pass this section and assert values to prove tht your logic is working as expected.Choose Run All Tests in the Developer Console at least once before attempting to submit this section. Be patient as it may take 10-20 seconds to process the challenge check.

Code for MaintenanceRequestHelperTest

```
@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);
    }
```

5. Test callout Logic

- Build tests for your callout using the included class for the callout mock(WarehouseCalloutServiceMock) and callout test class (WarehousecalloutServiceTest) in the package. You must have 100% test coverage to pass this challenge and assert values to prove that your logic is working as expected.

Code for WarehouseCalloutServiceMock

```
@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;
    }
}
```

Code for WarehouseCalloutServiceTest

```
@isTest
private class WarehouseCalloutServiceTest {
  // implement your mock callout test here
    @isTest
    static void WarehouseEquipmentSync(){
        Test.startTest();
        // Set mock callout class
        Test.setMock(HttpCalloutMock.class, new
WarehouseCalloutServiceMock());
        // This causes a fake response to be sent from the class that
implements HttpCalloutMock.
```

```
        WarehouseCalloutService.runWarehouseEquipmentSync();
        Test.stopTest();
        System.assertEquals(1, [SELECT count() FROM Product2]);
    }
}
```

6. Test Scheduling Logic

- Build unit tests for the class WarehouseSyncSchedule in class named WarehouseSyncScheduleTest. You must have 100% test coverage to pass this challenge and assert values to prove that your logic is working as expected.

Code for WarehouseSyncScheduleTest

```
isTest
public with sharing class WarehouseSyncScheduleTest {
    // implement scheduled code here
    //
    @isTest static void test() {
        String scheduleTime = '00 00 00 * * ? *';
        Test.startTest();
        Test.setMock(HttpCalloutMock.class, new WarehouseCalloutServiceMock());
        String jobId = System.schedule('Warehouse Time to Schedule to test', scheduleTime, new
WarehouseSyncSchedule());
        CronTrigger c = [SELECT State FROM CronTrigger WHERE Id =: jobId];
        System.assertEquals('WAITING', String.valueOf(c.State), 'JobId does not match');
```

```
        Test.stopTest();
    }
}
```