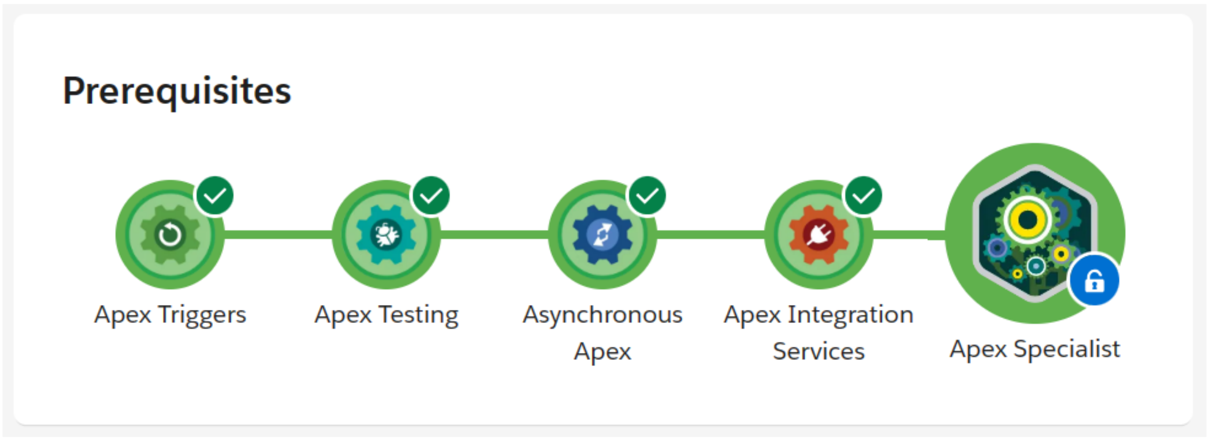
Apex Specialist Superbadge

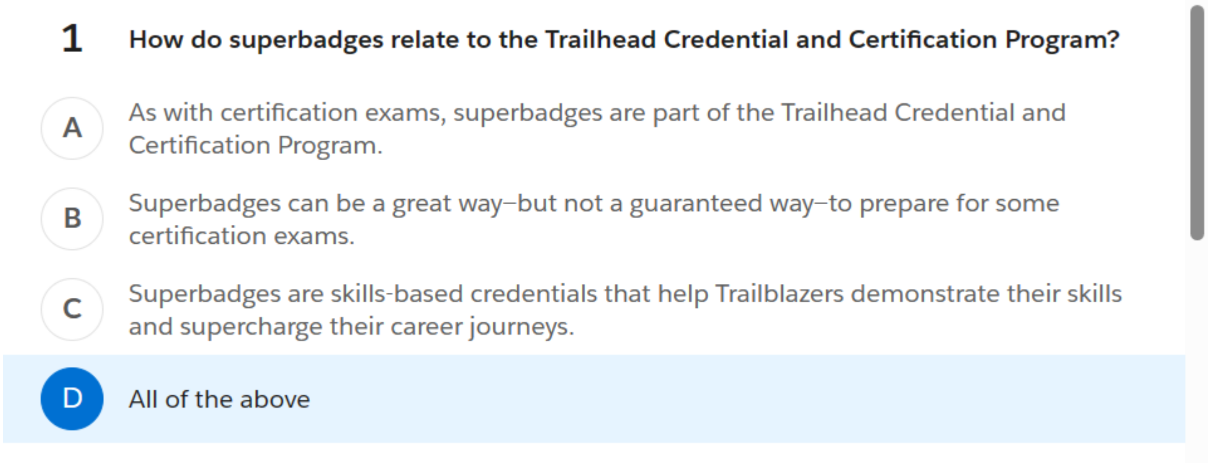
Firstly, I have completed all the prerequisites modules which are required for unlocking this Specialist Superbadge and are:

(1) Apex Triggers.
(2) Apex Testing.
(3) Asynchronous Apex.
(4) Apex Integration Services.



Now, let us start this Specialist Superbadge.

(1) Quiz: Credential Security.



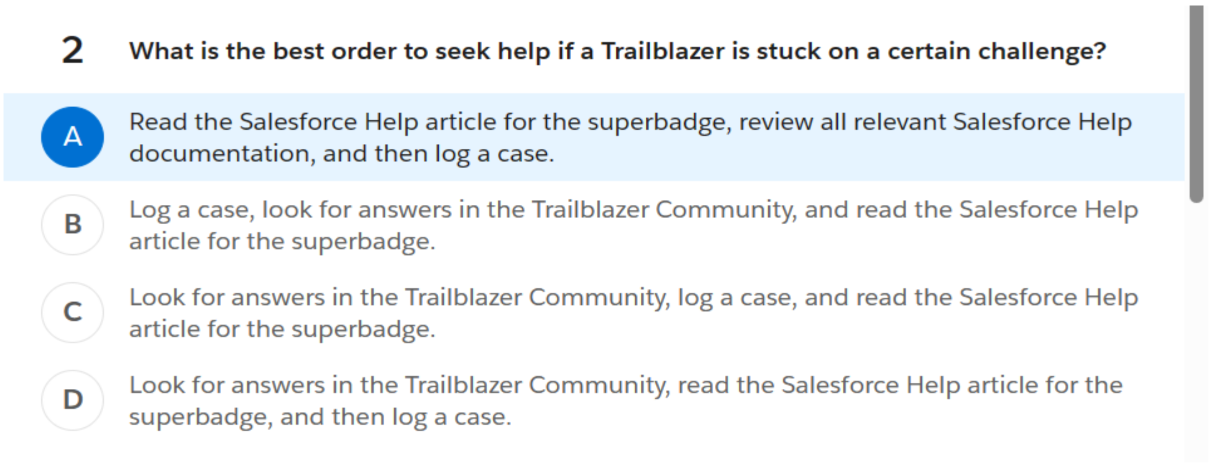
1 **How do superbadges relate to the Trailhead Credential and Certification Program?**

A As with certification exams, superbadges are part of the Trailhead Credential and Certification Program.

B Superbadges can be a great way–but not a guaranteed way–to prepare for some certification exams.

C Superbadges are skills-based credentials that help Trailblazers demonstrate their skills and supercharge their career journeys.

[✓ D] All of the above



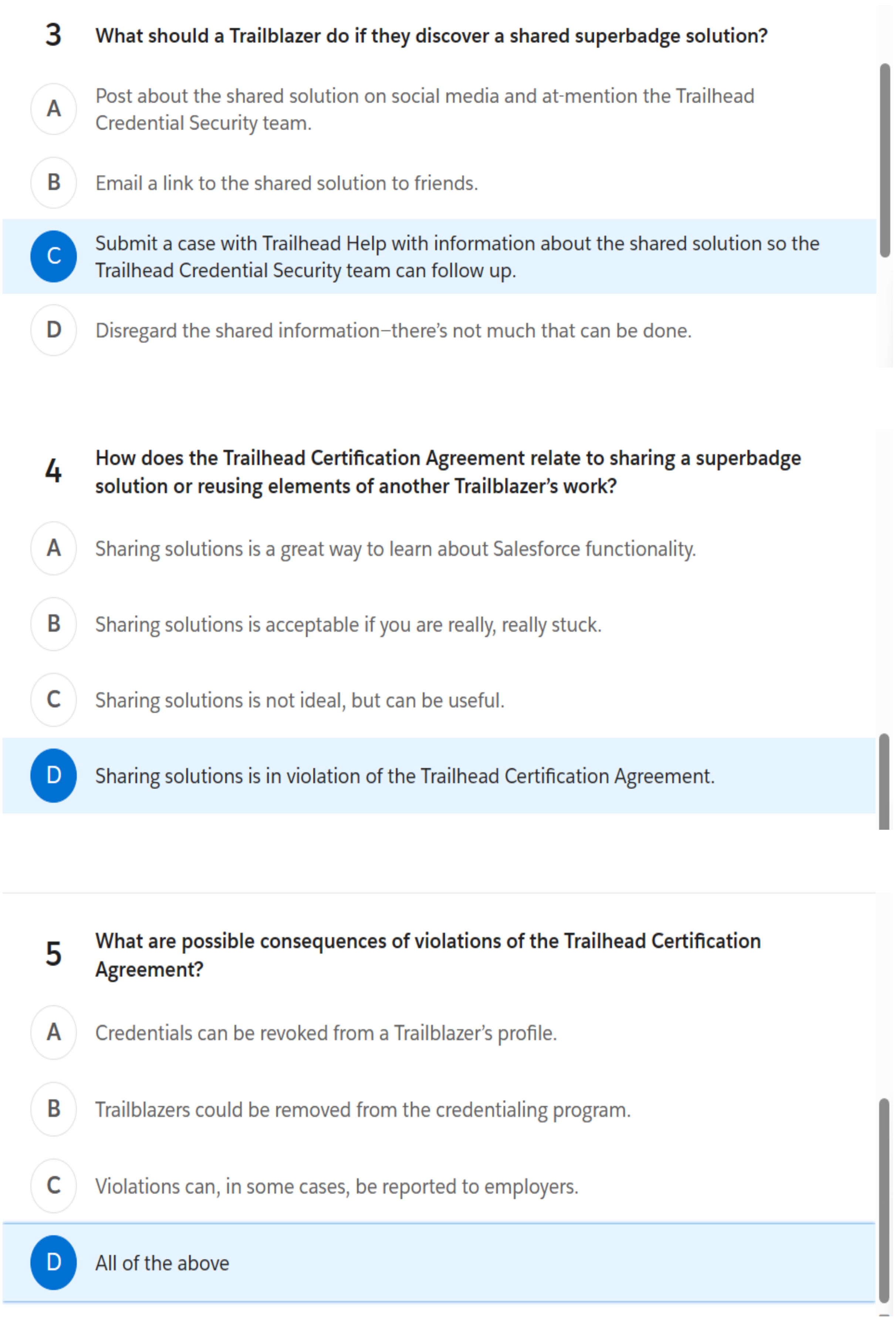
2 **What is the best order to seek help if a Trailblazer is stuck on a certain challenge?**

[✓ A] Read the Salesforce Help article for the superbadge, review all relevant Salesforce Help documentation, and then log a case.

B Log a case, look for answers in the Trailblazer Community, and read the Salesforce Help article for the superbadge.

C Look for answers in the Trailblazer Community, log a case, and read the Salesforce Help article for the superbadge.

D Look for answers in the Trailblazer Community, read the Salesforce Help article for the superbadge, and then log a case.



3 **What should a Trailblazer do if they discover a shared superbadge solution?**

A Post about the shared solution on social media and at-mention the Trailhead Credential Security team.

B Email a link to the shared solution to friends.

[✓ C Submit a case with Trailhead Help with information about the shared solution so the Trailhead Credential Security team can follow up.]

D Disregard the shared information–there's not much that can be done.

4 **How does the Trailhead Certification Agreement relate to sharing a superbadge solution or reusing elements of another Trailblazer's work?**

A Sharing solutions is a great way to learn about Salesforce functionality.

B Sharing solutions is acceptable if you are really, really stuck.

C Sharing solutions is not ideal, but can be useful.

[✓ D Sharing solutions is in violation of the Trailhead Certification Agreement.]

5 **What are possible consequences of violations of the Trailhead Certification Agreement?**

A Credentials can be revoked from a Trailblazer's profile.

B Trailblazers could be removed from the credentialing program.

C Violations can, in some cases, be reported to employers.

[✓ D All of the above]

Now, let us Setup a Development Org. Steps carried out to setup a Development Org for this entire Specialist Superbadge:

1. Create a new Trailhead Playground or Developer Edition Org for this superbadge. Using this org for any other reason might create problems when validating the challenge. If you choose to use a development org, make sure you deploy **My Domain** to all the users. The package you will install has some custom lightning components that only show when My Domain is deployed.
2. Install this unlocked package (package ID: 04t6g000008av9iAAA). This package contains metadata you'll use to complete this challenge. If you have trouble installing this package, follow the steps in the Install a Package or App to Complete a Trailhead Challenge help article.
3. Add picklist values `Repair` and `Routine Maintenance` to the **Type** field on the Case object.
4. Update the Case page layout assignment to use the **Case (HowWeRoll) Layout** for your profile.
5. Rename the tab/label for the Case tab to `Maintenance Request`.
6. Update the Product page layout assignment to use the **Product (HowWeRoll) Layout** for your profile.
7. Rename the tab/label for the Product object to `Equipment`.
8. Use App Launcher to navigate to the **Create Default Data** tab of the **How We Roll Maintenance** App. Click **Create Data** to generate sample data for the application.
9. Review the newly created records to get acquainted with the data model.

(2) Automate record creation.

Steps followed to complete this challenge:

1) After launching a Development Org, open “How We Roll Maintenance” via App Launcher and Click on Maintenance Request Tab and then click on case number “00001027”.
2) Then, click on Case Origin in Details Tab of chosen case number and select “Phone” and click Save button. After that click on Close Case Tab inside Feed and click Status as to be “Closed” and then click Save button.
3) Then, click on Maintenance Request Tab again and click on case number “00001026”.
4) Then, click on Case Origin in Details Tab of chosen case number and select “Phone” and in How We Roll Service Details Tab click on Vehicle and change it to “Teardrop Camper” and click Save button. After that click on Close Case Tab inside Feed and click Status as to be “Closed” and then click Save button.
5) Click on “Gear” icon on top-right side corner and select Setup and then, click on Object Manager Tab and then, search for “Maintenance Request” in Quick Find box and click on it.
6) After coming on to the Maintenance Request Object, click on Fields & Relationships and Click on New.
7) Select “Lookup Relationship” as Datatype and choose as “Equipment” in Related to object and click on Next.
8) Mention the Field Label and Field Name as “Equipment”, then click Next, Next and Save.

9) Now, again click on “Gear” icon and select Developer Console and open “MaintenanceRequestHelper.apxc” and “MaintenanceRequest.apxt” files which already exists.
10) Over-write the code mentioned in it with this new code in “MaintenanceRequestHelper.apxc” file:

```
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;
        }
    }
}
```

11) Over-write the code mentioned in it with this new code in "MaintenanceRequest.apxt" file:

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

12) Save both of the Apex code files and click on Check Challenge in your Trailhead Account, it will be successfully completed and submitted.

(3) Synchronize Salesforce data with an external system.
Steps followed to complete this challenge:

1) After launching a Development Org, click on "Gear" icon on top-right side corner and select Setup.
2) Now, search for "Remote Site Settings" in Quick Find box and click on New Remote Site.
3) Fill the details such as Remote Site Name and Remote Site URL as "WarehouseURL" and "https://th-superbadge-apex.herokuapp.com" respectively and click Save button.
4) Now, again click on "Gear" icon and select Developer Console and open "WarehouseCalloutService.apxc" file which already exists over there.
5) Over-write the code mentioned in it with this new code in "WarehouseCalloutService.apxc" file:

```
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();
    }
}
```

6) Save the above Apex code file named as “WarehouseCalloutService.apxc” and click on Debug > Open Execute Anonymous Window and mention this code in that window:
   System.enqueueJob(new WarehouseCalloutService());

7) After mentioning this code click on Open Log checkbox and Click “Execute” and then, click on Check Challenge in your Trailhead Account, it will be successfully completed and submitted.

(4) Schedule synchronization.
Steps followed to complete this challenge:

1) After launching a Development Org, click on “Gear” icon on top-right side corner and select “Developer Console”.
2) Click on File > Open and then click on “WarehouseSyncSchedule.apxc” and open it.
3) Over-write the code mentioned in it with this new code:

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

4) Save the “WarehouseSyncSchedule.apxc” file and then again launch a Development Org and click on “Gear” icon and select “Setup”
5) Search for “Apex Classes” in Quick Find box and click on it and then click on “Schedule Apex” button.
6) Now fill in the details as follows:
   Job Name: WarehouseSyncScheduleJob
   Select an Apex class as “WarehouseSyncSchedule” from Lookup.
   Select Sunday, Monday, etc. Everyday’s checkboxes and Preferred Start Time as 1:00 am.
7) Click on Save button and then, click on Check Challenge in your Trailhead Account, it will be successfully completed and submitted.

(5) Test automation logic.
Steps followed to complete this challenge:

1) After launching a Development Org, click on “Gear” icon on top-right side corner and select “Developer Console”.
2) Click on File > Open and then click on “MaintenanceRequestHelper.apxc” and “MaintenanceRequestHelperTest.apxc” and open both the files respectively.
3) Code needs to be written inside “MaintenanceRequestHelper.apxc” file:

```
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;
        }
    }
}
```

4) Code needs to be written inside “MaintenanceRequestHelperTest.apxc” file:

```
@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) Then, click on File > Save All and then followed by click on Test > Run All.
6) Then, click on Check Challenge in your Trailhead Account, it will be successfully completed and submitted.

(6) Test callout logic.

Steps followed to complete this challenge:

1) After launching a Development Org, click on “Gear” icon on top-right side corner and select “Setup”.
2) Search for “Scheduled Jobs” in Quick Find box and click on it and then, click on “Del” button just next to “WarehouseSyncScheduleJob” and delete that job.
3) Now, again click on “Gear” icon and select Developer Console and then, click on File > Open and open three existing Apex Class Files, namely, WarehouseCalloutService.apxc, WarehouseCalloutServiceMock.apxc, WarehouseCalloutServiceTest.apxc files respectively.
4) Code needs to be written inside “WarehouseCalloutService.apxc”:

```
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);
            }
        }
    }
}
```

5) Code needs to be written inside “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,"qu
antity":5,"name":"Generator 1000
kW","maintenanceperiod":365,"lifespan":120,"cost":5000,"sku":"100003"}]');
        response.setStatusCode(200);
        return response;
    }
}
```

6) Code needs to be written inside "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]);
    }
}
```

7) Then, click on File > Save All and then followed by click on Test > Run All.
8) Then, click on Check Challenge in your Trailhead Account, it will be successfully completed and submitted.

(7) Test scheduling logic.

Steps followed to complete this challenge:

1) After launching a Development Org, click on "Gear" icon on top-right side corner and select "Developer Console".
2) Click on File > Open and then click on "WarehouseSyncSchedule.apxc" and "WarehouseSyncScheduleTest.apxc" and open both the files respectively.
3) Code needs to be written inside "WarehouseSyncSchedule.apxc":

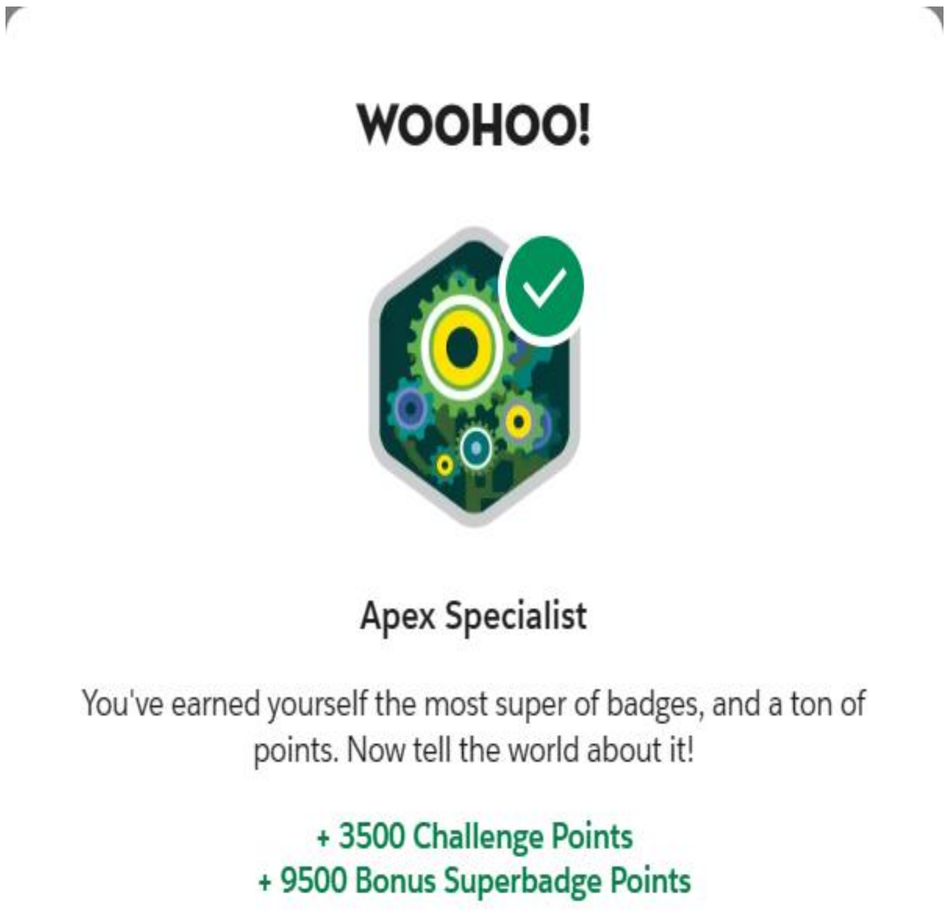
```
global class WarehouseSyncSchedule implements Schedulable {
    global void execute(SchedulableContext ctx) {
        WarehouseCalloutService.runWarehouseEquipmentSync();
    }
}
```

4) Code needs to be written inside "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 ');
    }
}
```

5) Then, click on File > Save All and then followed by click on Test > Run All.
6) Then, click on Check Challenge in your Trailhead Account, it will be successfully completed and submitted.

Finally, I have achieved the Apex Specialist Superbadge, the screenshot of completion is as follows:



WOOHOO!

Apex Specialist

You've earned yourself the most super of badges, and a ton of points. Now tell the world about it!

+ 3500 Challenge Points
+ 9500 Bonus Superbadge Points