Apex Specialist Superbadge:

1. Create a new Trailhead Playground 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.

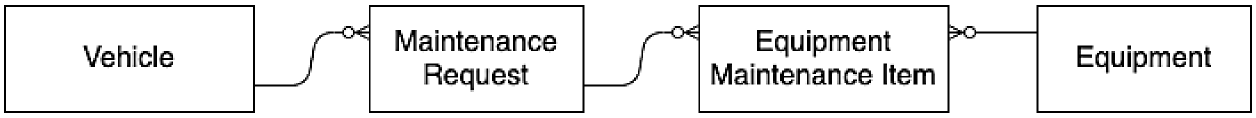
Standard Objects

Maintenance Request (renamed Case) — Service requests for broken vehicles, malfunctions, and routine maintenance.
Equipment (renamed Product) — Parts and items in the warehouse used to fix or maintain RVs.

Custom Objects

Vehicle — Vehicles in HowWeRoll's rental fleet.
Equipment Maintenance Item — Joins an Equipment record with a Maintenance Request record, indicating the equipment needed for the maintenance request.

Entity Diagram



Then according to the given process, write the code for each step mentioned below:

Step 1 : Answering the multiple choice questions.

Step 2 - Automate Record Creation : Automate record creation using apex triggers. Go to developer console and edit the apex class and the triggers for below:

MaintenanceRequestHelper

```
public with sharing class MaintenanceRequestHelper {
   public static void updateWorkOrders() {
      List<case> newCaseList = new List<case>();
      Integer avgAmount=10000;

      List<Equipment_Maintenance_Item__c> newEMI = new
List<Equipment_Maintenance_Item__c>();
      List<case> caseList = [SELECT id,Vehicle__c,Subject,ProductID,Product__c, (SELECT
id from Equipment_Maintenance_Items__r) from case where status='closed' and Type
IN ('Repair', 'Routine Maintenance') and ID IN :Trigger.new LIMIT 200];
      Map<id,Equipment_Maintenance_Item__c> equip = new
map<id,Equipment_Maintenance_Item__c>([Select ID, Equipment__c,
Quantity__c,Equipment__r.id,Equipment__r.Maintenance_Cycle__c from
Equipment_Maintenance_Item__c ]);
      for(case c: caseList){
         case newCase = new Case();
         newCase.Type = 'Routine Maintenance';
         newCase.Status = 'New';
         newCase.Vehicle__c = c.Vehicle__c;
         newCase.Subject =  String.isBlank(c.Subject) ? 'Routine Maintenance Request' :
c.Subject;
         newCase.Date_Reported__c = Date.today();
         newCase.ProductId = c.ProductId;
         newCase.Product__c = c.Product__c;
         newCase.parentID = c.Id;
         for(Equipment_Maintenance_Item__c emi : c.Equipment_Maintenance_Items__r ){
            avgAmount =
Math.min(avgAmount,Integer.valueOf(equip.get(emi.id).Equipment__r.Maintenance_Cyc
le__c));
            newEMI.add(new Equipment_Maintenance_Item__c(
               Equipment__c = equip.get(emi.id).Equipment__c,
               Maintenance_Request__c = c.id,
               Quantity__c = equip.get(emi.id).Quantity__c));
         }
         Date dueDate = date.TODAY().adddays(avgAmount);
         newCase.Date_Due__c =dueDate;
         newCaseList.add(newCase);

      }
      if(newCaseList.size()>0){
         Database.insert(newCaseList);
      }

      for(Case c2: newCaseList){
         for(Equipment_Maintenance_Item__c emi2 : newEmi){
            if(c2.parentID == emi2.Maintenance_Request__c){
               emi2.Maintenance_Request__c = c2.id;
            }
         }
      }

      if(newEmi.size()>0){
         Database.insert(newEmi);
      }
   }
}
```

MaintenanceRequestHelperTest

```
@istest
public with sharing class MaintenanceRequestHelperTest {
   @istest
```

```
    public static void BulkTesting(){
        product2 pt2 = new product2(Name = 'tester',Maintenance_Cycle__c = 10,
Replacement_Part__c = true);

        Database.insert(pt2);


        List<case> caseList = new List<case>();
        for(Integer i=0;i<300;i++){
            caseList.add(new case(
                Type = 'Routine Maintenance',
                Status = 'Closed',
                Subject = 'testing',
                Date_Reported__c = Date.today(),
                ProductId = pt2.id
            ));
        }
        if(caseList.size()>0){
            Database.insert(caseList);
            System.debug(pt2.id);
            System.debug(caseList.size());
        }


        List<Equipment_Maintenance_Item__c> newEMI = new
List<Equipment_Maintenance_Item__c>();
        for(Integer i=0;i<5;i++){
            newEMI.add(new Equipment_Maintenance_Item__c(
                Equipment__c = pt2.id,
                Maintenance_Request__c = caseList[1].id,
                Quantity__c = 10));
        }
        if(newEmi.size()>0){
            Database.insert(newEmi);
        }

        for(case c :caseList){
            c.Subject = 'For Testing';
        }
        Database.update(caseList);
        Integer newcase = [Select count() from case where ParentId = :caseList[0].id];
        System.assertEquals(1, newcase);

    }

    @istest
    public static void positive(){
        product2 pt2 = new product2(Name = 'tester',Maintenance_Cycle__c = 10);
        insert pt2;

        Case cParent = new Case(Type = 'Repair',status = 'Closed',Date_Reported__c =
Date.today(),
                                ProductId = pt2.id);
        insert cParent;
        Case cChild = new Case(Type = 'Repair',status = 'Closed',Date_Reported__c =
Date.today(),
                               ProductId = pt2.id,parentID = cParent.ParentId);
        insert cChild;

        cParent.subject = 'child refrecer record';
        update cParent;

        Integer newcase = [Select count() from case where ParentId = :cParent.id];
        System.assertEquals(1, newcase);

    }
    @istest public static void negetive(){
        product2 pt2 = new product2(Name = 'tester',Maintenance_Cycle__c = 10);
        insert pt2;

        Case c = new Case(Type = 'Repair',status = 'New',Date_Reported__c = Date.today(),
                          ProductId = pt2.id);
        insert c;
        c.Status = 'Working';
        update c;

        Integer newcase = [Select count() from case where ParentId = :c.id];
        System.assertEquals(0, newcase);
    }

}
```

Step 3 - Synchronize the salesforce data with an external system:

Modify the Apex Classes as below, save and run all.

WarehouseCalloutService

```
public with sharing class WarehouseCalloutService implements Queueable,
Database.AllowsCallouts{
    public List<product2> equip = new List<product2>();
    private static final String WAREHOUSE_URL = 'https://th-superbadge-
apex.herokuapp.com/equipment';



    public void execute(QueueableContext context) {
        //System.debug('Equipments'+equip );
        Http h = new Http();
        HttpRequest httpReq = new HttpRequest();
        httpReq.setMethod('GET');
        httpReq.setHeader('Content-Type','application/json');
        httpReq.setEndpoint(WAREHOUSE_URL);
        HttpResponse res = h.send(httpReq);
        List<Object> results = (List<Object>) JSON.deserializeUntyped(res.getBody());
        System.debug(results.size());

        for(Object fld : results){
            Map<String,Object> entry = (Map<String,Object>)fld;
            equip.add(new product2(
                Warehouse_SKU__c = String.valueOf(entry.get('_id')+''),
            Cost__c = Decimal.valueOf(entry.get('cost')+''),
            Lifespan_Months__c = Decimal.valueOf(entry.get('lifespan')+'') ,
            Maintenance_Cycle__c = Decimal.valueOf(entry.get('maintenanceperiod')+''),
            Name = String.valueOf(entry.get('name')+''),
            QuantityUnitOfMeasure = String.valueOf(entry.get('quantity')+'') ,
            Replacement_Part__c = Boolean.valueOf(entry.get('replacement') +''),
            StockKeepingUnit = String.valueOf(entry.get('sku')+'')
         ));
      }
      if(!equip.isEmpty())
      {
         upsert equip Warehouse_SKU__c;
         system.debug('list got updated. Size: '+equip.size());
      }
   }
}
```

Step 4 - Schedule Synchronization:
Modify the Apex Classes as below, save and run all.

WarehouseSyncSchdeule

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

Step 5 - Test automation logic :
Modify the Apex Classes as below, save and run all.

MaintenanceRequestHelper

```
public with sharing class MaintenanceRequestHelper {
  public static void updateWorkOrders() {
    List<case> newCaseList = new List<case>();
    Integer avgAmount=10000;

    List<Equipment_Maintenance_Item__c> newEMI = new
List<Equipment_Maintenance_Item__c>();
    List<case> caseList = [SELECT id,Vehicle__c,Subject,ProductID,Product__c, (SELECT
id from Equipment_Maintenance_Items__r) from case where status='closed' and Type
IN ('Repair', 'Routine Maintenance') and ID IN :Trigger.new LIMIT 200];
    Map<id,Equipment_Maintenance_Item__c> equip = new
map<id,Equipment_Maintenance_Item__c>([Select ID, Equipment__c,
Quantity__c,Equipment__r.id,Equipment__r.Maintenance_Cycle__c from
Equipment_Maintenance_Item__c ]);
    for(case c: caseList){
      case newCase = new Case();
      newCase.Type = 'Routine Maintenance';
      newCase.Status = 'New';
      newCase.Vehicle__c = c.Vehicle__c;
      newCase.Subject =  String.isBlank(c.Subject) ? 'Routine Maintenance Request' :
c.Subject;
      newCase.Date_Reported__c = Date.today();
      newCase.ProductId = c.ProductId;
      newCase.Product__c = c.Product__c;
      newCase.parentID = c.Id;


      for(Equipment_Maintenance_Item__c emi : c.Equipment_Maintenance_Items__r ){
            avgAmount =
Math.min(avgAmount,Integer.valueOf(equip.get(emi.id).Equipment__r.Maintenance_Cyc
le__c));
            newEMI.add(new Equipment_Maintenance_Item__c(
               Equipment__c = equip.get(emi.id).Equipment__c,
               Maintenance_Request__c = c.id,
               Quantity__c = equip.get(emi.id).Quantity__c));
         }
         Date dueDate = date.TODAY().adddays(avgAmount);
         newCase.Date_Due__c =dueDate;
         newCaseList.add(newCase);

      }
      if(newCaseList.size()>0){
         Database.insert(newCaseList);
      }

      for(Case c2: newCaseList){
         for(Equipment_Maintenance_Item__c emi2 : newEmi){
            if(c2.parentID == emi2.Maintenance_Request__c){
               emi2.Maintenance_Request__c = c2.id;
            }
         }
      }

      if(newEmi.size()>0){
         Database.insert(newEmi);
      }
   }
}
```

MaintenanceRequestHelperTest

```
@istest
public with sharing class MaintenanceRequestHelperTest {
   @istest
   public static void BulkTesting(){
        product2 pt2 = new product2(Name = 'tester',Maintenance_Cycle__c = 10,
Replacement_Part__c = true);

        Database.insert(pt2);


        List<case> caseList = new List<case>();
        for(Integer i=0;i<300;i++){
            caseList.add(new case(
                Type = 'Routine Maintenance',
                Status = 'Closed',
                Subject = 'testing',
                Date_Reported__c = Date.today(),
                ProductId = pt2.id
            ));
        }
        if(caseList.size()>0){
            Database.insert(caseList);
            System.debug(pt2.id);
            System.debug(caseList.size());
        }


        List<Equipment_Maintenance_Item__c> newEMI = new
List<Equipment_Maintenance_Item__c>();
        for(Integer i=0;i<5;i++){
            newEMI.add(new Equipment_Maintenance_Item__c(
                Equipment__c = pt2.id,
                Maintenance_Request__c = caseList[1].id,
                Quantity__c = 10));
        }
        if(newEmi.size()>0){
            Database.insert(newEmi);
        }
        for(case c :caseList){
            c.Subject = 'For Testing';
        }
        Database.update(caseList);
        Integer newcase = [Select count() from case where ParentId = :caseList[0].id];
        System.assertEquals(1, newcase);
    }

    @istest
    public static void positive(){
        product2 pt2 = new product2(Name = 'tester',Maintenance_Cycle__c = 10);
        insert pt2;

        Case cParent = new Case(Type = 'Repair',status = 'Closed',Date_Reported__c =
Date.today(),
                            ProductId = pt2.id);
        insert cParent;
        Case cChild = new Case(Type = 'Repair',status = 'Closed',Date_Reported__c =
Date.today(),
                            ProductId = pt2.id,parentID = cParent.ParentId);
        insert cChild;
        cParent.subject = 'child refrecer record';
        update cParent;
        Integer newcase = [Select count() from case where ParentId = :cParent.id];
        System.assertEquals(1, newcase);

    }
    @istest public static void negetive(){
        product2 pt2 = new product2(Name = 'tester',Maintenance_Cycle__c = 10);
        insert pt2;

        Case c = new Case(Type = 'Repair',status = 'New',Date_Reported__c = Date.today(),
                    ProductId = pt2.id);
        insert c;

        c.Status = 'Working';
        update c;

        Integer newcase = [Select count() from case where ParentId = :c.id];
        System.assertEquals(0, newcase);
```

```
   }
}
```

Step 6 - Test callout logic :
Modify the Apex Classes as below, save and run all.

WarehouseCalloutServiceTest

```
@IsTest
private class WarehouseCalloutServiceTest {
   // implement your mock callout test here
   @isTest static void mainTest(){
      Test.setMock(HttpCalloutMock.class, new WarehouseCalloutServiceMock());
      Test.startTest();
      Id jobID = System.enqueueJob(new WarehouseCalloutService());
      //System.assertEquals('Queued',aaj.status);
      Test.stopTest();
      AsyncApexJob aaj = [SELECT Id, Status, NumberOfErrors FROM AsyncApexJob
WHERE Id = :jobID];
      System.assertEquals('Completed',aaj.status);
      System.assertEquals(0, aaj.NumberOfErrors);
   }
}
```

WarehouseCalloutServiceMock

```
@istest
global class WarehouseCalloutServiceMock implements HttpCalloutMock{
   // implement http mock callout
   global HttpResponse respond(HttpRequest request){
      HttpResponse response = new HttpResponse();
      response.setHeader('Content-Type', 'application/json');

response.setBody('[{"_id":"55d66226726b611100aaf741","replacement":true,"quantity":5,"
name":"Generator 1000
kW","maintenanceperiod":365,"lifespan":120,"cost":5000,"sku":"220000"}]');
      response.setStatusCode(200);
      return response;
   }
}
```

Step 7 - Test scheduling logic :
Modify the Apex Classes as below, save and run all.

WarehouseSyncSchedule

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

    }
}
```

WarehouseSyncScheduleTest

```
@isTest
public class WarehouseSyncScheduleTest {
    public static String CRON_EXP = '0 0 1 * * ?';

    @isTest
    static void testExecute(){
        Test.setMock(HttpCalloutMock.class, new WarehouseCalloutServiceMock());

        Test.startTest();
        String jobId = System.schedule('WarehouseSyncScheduleTest', CRON_EXP, new
WarehouseSyncSchedule());
        Test.stopTest();

        System.assertEquals(1, [SELECT count() FROM CronTrigger WHERE
CronJobDetail.Name = 'WarehouseSyncScheduleTest']);
    }
}
```