1. Created new trailhead playground
2. Install this unlocked package (package ID: 04t6g000008av9iAAA).
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

Automate record creation

created apex class as MaintainanceRequiredHelper

code

```
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;
      }
   }
}
```

created trigger for Maintainance required for case object

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

```
    }

}
```

- After saving the code go back the How We Roll Maintenance ,
- click on Maintenance Requests -> click on 2nd case -> click Details -> change the type Repair to Routine Maintenance -> select Origin = Phone -> Vehicle = select Teardrop Camper , save it.
- Feed -> Close Case = save it..

challenge 2

Implement an Apex class (called WarehouseCalloutService) that implements the queueable interface and makes a callout to the external service used for warehouse inventory management. This service receives updated values in the external system and updates the related records in Salesforce. Before checking this section,

- Setup -> Search in quick find box -> click Remote Site Settings -> Name = Warehouse URL , Remote Site URL = https://th-superbadge-apex.herokuapp.com , make sure active is selected.
- Go to the developer console use below code

```
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();
   }

}
```

Go to the developer console and use below code

Product2Extension.apxc

Product2Extension.cls :

```
public without sharing class Product2Extension {

 public List<ProductWrapper> productsToInsert {get; set;}

 public Product2Extension(ApexPages.StandardController controller){
   productsToInsert = new List<ProductWrapper>();
   AddRows();
 }

 public void AddRows(){
   for (Integer i=0; i<Constants.DEFAULT_ROWS; i++ ) {
     productsToInsert.add( new ProductWrapper() );
   }
 }
 public List<ChartHelper.ChartData> GetInventory(){
   return ChartHelper.GetInventory();
 }

 public List<SelectOption> GetFamilyOptions() {
   List<SelectOption> options = new List<SelectOption>();
   options.add(new SelectOption(Constants.SELECT_ONE, Constants.SELECT_ONE));
   for(PickListEntry eachPicklistValue : Constants.PRODUCT_FAMILY) {
     options.add(new SelectOption(eachPicklistValue.getValue(),
eachPicklistValue.getLabel()));
   }
   return options;
 }

 public PageReference Save(){
   SavePoint sp = Database.setSavepoint();
   Integer insertedCount = 0;
   try {
     List<Product2> newProducts = new List<Product2>();
     List<PriceBookEntry> pbeList = new List<PriceBookEntry>();
     List<ProductWrapper> filteredProductWrappers = new List<ProductWrapper>();

     for(ProductWrapper eachPW : productsToInsert) {
       if(!String.isBlank(eachPW.productRecord.Name) &&
!String.isBlank(eachPW.productRecord.Family) && eachPW.productRecord.Family !=
Constants.SELECT_ONE && eachPW.productRecord.isActive &&
eachPW.pricebookEntryRecord.UnitPrice != null &&
eachPW.pricebookEntryRecord.UnitPrice != 0 &&
eachPW.productRecord.Initial_Inventory__c != null &&
eachPW.productRecord.Initial_Inventory__c != 0) {
         filteredProductWrappers.add(eachPW);
       }
```

```
      }
      for(ProductWrapper eachPW : filteredProductWrappers) {
        newProducts.add(eachPW.productRecord);
      }

      Database.SaveResult[] productSaveResults = Database.insert(newProducts, false);

      for(Integer i = 0; i < productSaveResults.size(); i++) {
        if(productSaveResults[i].isSuccess()) {
          PriceBookEntry pbe = filteredProductWrappers[i].pricebookEntryRecord;
          pbe.Product2Id = productSaveResults[i].getId();
          pbe.IsActive = true;
          pbe.Pricebook2Id = Constants.STANDARD_PRICEBOOK_ID;
          pbeList.add(pbe);
          insertedCount++;
        }
      }

      Database.SaveResult[] pbeSaveResults = Database.insert(pbeList, false);

      apexPages.addMessage(new
ApexPages.message(ApexPages.Severity.INFO,insertedCount + ' Inserted'));
      productsToInsert.clear();
      addRows();
    }
    catch (Exception e){
      System.debug('Exception occured:'+e.getMessage());
      Database.rollback(sp);
      apexPages.addMessage(new ApexPages.message(ApexPages.Severity.ERROR,
Constants.ERROR_MESSAGE));
    }
    return null;
```

```
 }

 public class ProductWrapper {
   public Product2 productRecord {get; set;}
   public PriceBookEntry pricebookEntryRecord {get; set;}

   public ProductWrapper() {
     productRecord = new product2(Initial_Inventory__c =0);
     pricebookEntryRecord = new pricebookEntry(Unitprice=0.0);
   }
 }
}
```

Product2New.vfp

```
<apex:page standardcontroller="Product2" extensions="Product2Extension">
   <apex:sectionHeader title="New Product" subtitle="Add Inventory" />
   <apex:pageMessages id="pageMessages" />
   <apex:form id="form" >
      <apex:actionRegion >
         <apex:pageBlock title="Existing Inventory" id="existingInv">
            <apex:chart data="{!Inventory}" width="600" height="400">
               <apex:axis type="Category" fields="name" position="left" title="Product
Family"/>
               <apex:axis type="Numeric" fields="val" position="bottom" title="Quantity
Remaining"/>
               <apex:barSeries axis="bottom" orientation="horizontal" xField="val"
yField="name"/>
```

```
            </apex:chart>
        </apex:pageBlock>
        <apex:pageBlock title="New Products" >
            <apex:pageBlockButtons location="top">
                <apex:commandButton action="{!save}" value="Save" reRender="existingInv,
orderItemTable, pageMessages"/>
            </apex:pageBlockButtons>
            <apex:pageBlockButtons location="bottom">
                <apex:commandButton action="{!addRows}" value="Add"
reRender="orderItemTable, pageMessages" />
            </apex:pageBlockButtons>

            <apex:pageBlockTable value="{!productsToInsert}" var="p" id="orderItemTable"
>
                <apex:column headerValue="{!$ObjectType.Product2.Fields.Name.Label}" >
                    <apex:inputText value="{!p.productRecord.Name}" />
                </apex:column>
                <apex:column headerValue="{!$ObjectType.Product2.Fields.Family.Label}" >
                    <apex:selectList value="{!p.productRecord.Family}" size="1"
multiselect="false">
                        <apex:selectOptions value="{!FamilyOptions}"></apex:selectOptions>
                    </apex:selectList>
                </apex:column>
                <apex:column headerValue="{!$ObjectType.Product2.Fields.IsActive.Label}" >
                    <apex:inputField value="{!p.productRecord.isActive}" />
                </apex:column>
                <apex:column
headerValue="{!$ObjectType.PricebookEntry.Fields.UnitPrice.Label}" >
                    <apex:inputText value="{!p.pricebookEntryRecord.UnitPrice}" />
                </apex:column>
                <apex:column
headerValue="{!$ObjectType.Product2.Fields.Initial_Inventory__c.Label}" >
```

```
                    <apex:inputField value="{!p.productRecord.Initial_Inventory__c}" />
                </apex:column>
            </apex:pageBlockTable>
        </apex:pageBlock>
    </apex:actionRegion>
  </apex:form>
</apex:page>
```

product2Trigger.apxt

```
/**
 * @name product2Trigger
 * @description Trigger to notify staff of low levels of inventory
**/
trigger product2Trigger on Product2 (
    before insert,
    before update,
    before delete,
    after insert,
    after update,
    after delete,
    after undelete
) {
    try {
        for ( Product2 p : Trigger.New ){
            if (
                p.Id != null && (
                    ( p.Family == 'Entree' && p.Quantity_Remaining__c < 20 )||
                    ( p.Family == 'Side' && p.Quantity_Remaining__c < 10 )||
                    ( p.Family == 'Dessert' && p.Quantity_Remaining__c < 15 )||
                    ( p.Family == 'Beverage' && p.Quantity_Remaining__c < 5 )
                )
            ){
                insert new FeedItem(
                    Body=p.Name+' Quantity is down to '+p.Quantity_Remaining__c,
                    ParentId = p.Id
                );
            }
        }
    } catch ( Exception e ){
        //A good developer would do something with this Exception!
    }
}
```

Constants.apxc

```
public class Constants {
  public static final Integer DEFAULT_ROWS = 5;
  public static final String SELECT_ONE = Label.Select_One;
  public static final String INVENTORY_LEVEL_LOW = Label.Inventory_Level_Low;
  public static final List<Schema.PicklistEntry> PRODUCT_FAMILY =
Product2.Family.getDescribe().getPicklistValues();
```

```
    public static final String DRAFT_ORDER_STATUS = 'draft';
    public static final String ACTIVATED_ORDER_STATUS = 'activated';
    public static final String ERROR_MESSAGE = 'An error has occurred, please take a
screenshot with the URL and send it to IT.';
    public static final Id STANDARD_PRICEBOOK_ID = '01s5g00000HLBKKAA5';

}
```

ChartHelper.apxc

```
public without sharing class ChartHelper {

    @AuraEnabled
    public static List<chartData> GetInventory(){
        List<chartData> cht = new List<chartData>();
        //ToDo: Perform a calculation that aggregates active Products that have a positive
Quantity_Remaining__c
        //And return a list of chartData
        //Where the name is the Product Family and the Qty is the sum of the
Quantity_Remaining__c

        for(AggregateResult ar : [SELECT Family, SUM(Quantity_Remaining__c) FROM
Product2 WHERE Quantity_Remaining__c > 0 AND IsActive = true GROUP BY Family]) {
            cht.add(new ChartData(String.ValueOf(ar.get('Family')),
Integer.ValueOf(ar.get('expr0'))));
        }
        return cht;
        return cht;
    }

    public class ChartData {
        public String name {get;set;}
        public Decimal val {get;set;}

        public ChartData(String name, Decimal val){
            this.name = name;
            this.val = val;
        }
    }

}
```

TestDataDFactory.apxc

```
/**
* @name TestDataFactory
* @description Contains methods to construct and/or validate commonly used records
**/
public with sharing class TestDataFactory{

    //@name ConstructCollaborationGroup
    //@description
    public static CollaborationGroup ConstructCollaborationGroup(){
        //ToDo: Ensure this method returns a single Chatter CollaborationGroup
        //    whose Name starts with 'TEST' followed by the INVENTORY_ANNOUNCEMENTS
constant
        //    and configured so anyone can join, see and post updates.
        CollaborationGroup cgroup = new CollaborationGroup();
        cgroup.Name = 'TEST' + Constants.INVENTORY_ANNOUNCEMENTS;
        cgroup.CanHaveGuests = false;
        cgroup.CollaborationType = 'Public';
        cgroup.IsArchived = false;
        cgroup.IsAutoArchiveDisabled = false;
        return cgroup;
    }

    //@name CreateProducts
    //@description Constructs a list of Product2 records for unit tests
    public static List<Product2> ConstructProducts(Integer cnt){
        //ToDo: Ensure this method returns a list, of size cnt, of uniquely named Product2
records
        //  with all the required fields populated
        //  and IsActive = true
        //  an Initial Inventory set to 10
        //  and iterating through the product family picklist values throughout the list.

        List<Schema.PickListEntry> familyValueList =
Product2.Family.getDescribe().getPickListValues();
        Integer listSize = familyValueList.size();

        List<Product2> productList = new List<Product2>();
        for (Integer i = 0; i < cnt; i++) {
            Product2 p = new Product2();
            p.Name = 'Product ' + i;
            p.Family = familyValueList[Math.mod(i, listSize)].getValue();
            p.Initial_Inventory__c = 10;
            p.IsActive = true;
            productList.add(p);
        }
        return productList;
    }

    //@name CreatePricebookEntries
    //@description constructs a List of PricebookEntry records for unit tests
    public static List<PricebookEntry> constructPricebookEntries(List<Product2>
productList){
        //ToDo: Ensure this method returns a corresponding list of PricebookEntries records
        //  related to the provided Products
        //  with all the required fields populated
        //  and IsActive = true
        //  and belonging to the standard Pricebook
        List<PricebookEntry> pbes = new List<PricebookEntry>();
        for (Product2 product: productList) {
            PricebookEntry pbe = new PricebookEntry();
            pbe.Pricebook2Id = Constants.STANDARD_PRICEBOOK_ID;
            pbe.Product2Id = product.Id;
            pbe.IsActive = true;
            pbe.UnitPrice = 1;
            pbes.add(pbe);
        }
        return pbes;
    }

    //@name CreateAccounts
    //@description constructs a List of Account records for unit tests
    public static List<Account> constructAccounts(Integer cnt){
        //ToDo: Ensure this method returns a list of size cnt of uniquely named Account
records
        //  with all of the required fields populated.
        List<Account> accts = new List<Account>();
        for (Integer i = 0; i < cnt; i++) {
            Account acct = new Account();
            acct.Name = 'Account ' + i;
            accts.add(acct);
        }
        return accts;
    }

    //@name CreateContacts
    //@description constructs a List of Contacxt records for unit tests
    public static List<Contact> constructContacts(Integer cnt, List<Account> accts){
        //ToDo: Ensure this method returns a list, of size cnt, of uniquely named Contact
records
        //  related to the provided Accounts
        //  with all of the required fields populated.
        Integer listSize = accts.size();

        List<Contact> contactList = new List<Contact>();
        for (Integer i = 0; i < cnt; i++) {
            Contact c = new Contact();
            c.LastName = 'Contact ' + i;
            c.AccountId = accts[Math.mod(i, listSize)].Id;
            contactList.add(c);
        }

        return contactList;
    }

    //@name CreateOrders
    //@description constructs a List of Order records for unit tests
    public static List<Order> constructOrders(Integer cnt, List<Account> accts){
        //ToDo: Ensure this method returns a list of size cnt of uniquely named Order records
        //  related to the provided Accounts
        //  with all of the required fields populated.
        Integer listSize = accts.size();

        List<Order> orders = new List<Order>();

        for (Integer i = 0; i < cnt; i++) {
            Order o = new Order();
            o.Name = 'Order ' + i;
            o.AccountId = accts[Math.mod(i, listSize)].Id;
            o.EffectiveDate = Date.today();
            o.Pricebook2Id = Constants.STANDARD_PRICEBOOK_ID;
            o.Status = 'Draft';
            orders.add(o);
        }

        return orders;
    }

    //@name CreateOrderItems
    //@description constructs a List of OrderItem records for unit tests
    public static List<OrderItem> constructOrderItems(Integer cnt, List<Pricebookentry>
pbes, List<Order> ords){
        //ToDo: Ensure this method returns a list of size cnt of OrderItem records
        //  related to the provided Pricebook Entries
        //  and related to the provided Orders
        //  with all of the required fields populated.
        //  Hint: Use the DEFAULT_ROWS constant for Quantity as it will be used in the next
challenge
        Integer pbeListSize = pbes.size();
        Integer orderListSize = ords.size();

        List<OrderItem> orderItemList = new List<OrderItem>();
        for (Integer i = 0; i < cnt; i++) {
            OrderItem oi = new OrderItem();
            oi.OrderId = ords[Math.mod(i, orderListSize)].Id;
            oi.PriceBookEntryId = pbes[Math.mod(i, pbeListSize)].Id;
            oi.Quantity = Constants.DEFAULT_ROWS;
            oi.UnitPrice = 1;
            orderItemList.add(oi);
        }

        return orderItemList;
    }

    //@name SetupTestData
    //@description Inserts accounts, contacts, Products, PricebookEntries, Orders, and
OrderItems.
    public static void InsertTestData(Integer cnt){
        //ToDo: Ensure this method calls each of the construct methods
        //  and inserts the results for use as test data.

        insert constructCollaborationGroup();

        List<Product2> productList = constructProducts(cnt);
        insert productList;

        List<PricebookEntry> pbes = constructPricebookEntries(productList);
        insert pbes;

        List<Account> accts = constructAccounts(cnt);
        insert accts;
        insert constructContacts(cnt, accts);
        List<Order> ords = constructOrders(cnt, accts);
        insert ords;

        insert constructOrderItems(cnt, pbes, ords);
    }

    //Step5
    public static void VerifyQuantityOrdered(Product2 originalProduct, Product2
updatedProduct, Integer qtyOrdered){
        Integer sumQuantity = Integer.valueOf(originalProduct.Quantity_Ordered__c) +
qtyOrdered;
        System.assertEquals(updatedProduct.Quantity_Ordered__c, sumQuantity);
    }
}
```

Constants.apxc

```
public class Constants {
    public static final Integer DEFAULT_ROWS = 5;
    public static final String SELECT_ONE = Label.Select_One;
    public static final String INVENTORY_LEVEL_LOW = Label.Inventory_Level_Low;
    public static final List<Schema.PicklistEntry> PRODUCT_FAMILY =
Product2.Family.getDescribe().getPicklistValues();
    public static final String DRAFT_ORDER_STATUS = 'draft';
    public static final String ACTIVATED_ORDER_STATUS = 'activated';
    public static final String ERROR_MESSAGE = 'An error has occurred, please take a
screenshot with the URL and send it to IT.';
```

```
    public static final String INVENTORY_ANNOUNCEMENTS = 'Inventory
Announcements';
    public static final Id STANDARD_PRICEBOOK_ID = '01s5g00000HLBKKAA5';

}
```

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);
         }

      }
   }
}
```

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,"quantity":5
,"name":"Generator 1000
kW","maintenanceperiod":365,"lifespan":120,"cost":5000,"sku":"100003"}]');
        response.setStatusCode(200);
        return response;
    }
}
```

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 ');



  }
}
```