Apex Specialist Superbadge:

In this superbadge, inifial step is to create a new playground. Now the steps which are menfioned in 'set up development org' has to be done. Then according to the given process, write the code for each step menfioned below:

Step 1 : Answering the mulfiple choice quesfions.

Step 2 - Automate Record Creafion :

Automate record creafion using apex triggers.

Go to developer console and edit the apex class and the triggers for below:

MaintenanceRequestHelper

```
public static string strTestFlag = '';

  public static void updateWorkOrders(MAP<ID, Case> triggerOldMAP,
MAP<ID, Case> triggerNewMAP){

    try {
      if (strTestFlag == 'Try') integer nThrowTestError = 1 / 0;

      // Get ID's in scope
      LIST<Case> caseLIST = new LIST<Case>();
      system.debug('triggerOldMAP: ' + triggerOldMAP);
      system.debug('triggerNewMAP: ' + triggerNewMAP);
      for (ID caseID : triggerNewMAP.keySet()) {
        if ((triggerOldMAP.get(caseID).Type <> triggerNewMAP.get(caseID).Type
          || triggerOldMAP.get(caseID).Status <> triggerNewMAP.get(caseID).Status)
          && triggerNewMAP.get(caseID).Status == 'Closed'
          && (triggerNewMAP.get(caseID).Type == 'Routine Maintenance' ||
triggerNewMAP.get(caseID).Type == 'Repair')) {

          caseLIST.add(triggerNewMAP.get(caseID));
          system.debug('caseLIST: ' + caseLIST);
        } // END if (triggerOldMAP.get(caseID).Type <> triggerNewMAP.get(caseID).Type...
```

Apex Specialist Superbadge:

```
        } // END for (Case caseItem : triggerNewLIST)

        // Get full records needed in scope
        LIST<Case> caseScopeLIST = new LIST<Case>();
        if (caseLIST.size() > 0) {
            caseScopeLIST =
                [select ID, Vehicle__c, Type, Status,
                    (select ID, Name, Equipment__c, Quantity__c, Maintenance_Request__c,
Equipment__r.Maintenance_Cycle__c from Work_Parts__r)
                    from Case
                    where ID in :caseLIST];
            system.debug('caseScopeLIST: ' + caseScopeLIST);
        } // END if (caseLIST.size() > 0)

        // Create new Cases and Work Parts for records in scope
        if (caseScopeLIST.size() > 0) {
            MAP<integer, string> errToReqMAP = new MAP<integer, string>();
            LIST<Case> caseInsLIST = new LIST<Case>();
            LIST<Work_Part__c> wpInsLIST = new LIST<Work_Part__c>();
            MAP<integer, Work_Part__c> newCaseWPMAP = new MAP<integer, Work_Part__c>();

                                        integer nRcdNbr = 0;
            for (Case caseItem : caseScopeLIST) {
                for (Work_Part__c wpItem : caseItem.Work_Parts__r) {
                    caseInsLIST.add(new Case(
                        Subject = caseItem.ID + ' Scheduled Checkup',
                        Description = caseItem.ID + ' Scheduled Checkup for related Work Part',
                        Vehicle__c = caseItem.Vehicle__c,
                        Type = (strTestFlag <> 'Case' ? 'Routine Maintenance' : 'Routine Maintenance Routine
Maintenance Routine Maintenance Routine Maintenance Routine Maintenance'),
                        Date_Reported__c = system.Today(),
                        Date_Due__c = system.Today() + (wpItem.Equipment__r.Maintenance_Cycle__c == null ? 0
: (integer) wpItem.Equipment__r.Maintenance_Cycle__c)));
                    newCaseWPMAP.put(nRcdNbr, new Work_Part__c(Equipment__c = wpItem.Equipment__c,
Quantity__c = wpItem.Quantity__c, Maintenance_Request__c = wpItem.Maintenance_Request__c)); //
Maintenance_Request__c will be replaced if successful
                    nRcdNbr ++;
                } // END for (Work_Part__c wpItem : caseItem.Work_Parts__r)
            } // END for (Case caseItem : caseScopeLIST)
            system.debug('caseInsLIST: ' + caseInsLIST);
            system.debug('newCaseWPMAP: ' + newCaseWPMAP);
```

Apex Specialist Superbadge:

```
if (caseInsLIST.size() > 0) {
                nRcdNbr = 0;
                Work_Part__c wpItem = new Work_Part__c();
                MAP<integer, integer> wpToNewCaseMAP = new MAP<integer, integer>();

                for (database.SaveResult srItem : database.insert(caseInsLIST, false)) {
                   system.debug('srItem: ' + srItem);
                   if (!srItem.IsSuccess()) {
                      system.debug('Error Detected on Case Insert for Record ' + string.valueOf(nRcdNbr) + ': ' +
srItem);
                      triggerNewMAP.get(caseInsLIST[nRcdNbr].Subject.left(18)).addError('Error creating
Scheduled Case for Case ID: ' + caseInsLIST[nRcdNbr].Subject.left(18) + ', Error: ' +
srItem.getErrors()[0].getMessage());
                   } else {
                      wpItem = new Work_Part__c();
                      wpItem = newCaseWPMAP.get(nRcdNbr);
                      wpItem.Maintenance_Request__c = srItem.getID();
                      wpInsLIST.add(wpItem);
                      wpToNewCaseMAP.put(wpInsLIST.size() - 1, nRcdNbr);
                   } // END if (!srItem.IsSuccess())
                   nRcdNbr ++;
                } // END for (database.SaveResult srItem : database.insert(caseInsLIST, false))
                system.debug('wpToNewCaseMAP: ' + wpToNewCaseMAP);

                if (wpInsLIST.size() > 0) {
                   nRcdNbr = 0;
                   for (database.SaveResult srItem : database.insert(wpInsLIST, false)) {
                      system.debug('srItem: ' + srItem);
                      if (!srItem.IsSuccess() || strTestFlag == 'WorkPart') {
                         system.debug('Error Detected on Work Part Insert for Record ' + string.valueOf(nRcdNbr)
+ ': ' + srItem);
                         string strError = (strTestFlag == 'WorkPart' ? 'Testing Error from Test Class' :
srItem.getErrors()[0].getMessage());

triggerNewMAP.get(caseInsLIST[wpToNewCaseMAP.get(nRcdNbr)].Subject.left(18)).addError('Error
```

Apex Specialist Superbadge:

```
creating Scheduled Work Part for Case ID: ' +
caseInsLIST[wpToNewCaseMAP.get(nRcdNbr)].Subject.left(18) + ', Error: ' + strError);
                    } // END if (!srItem.IsSuccess())
                    nRcdNbr ++;
                  } // END for (database.SaveResult srItem : database.insert(wpInsLIST, false))
                } // END if (wpInsLIST.size() > 0)

              } // END if (caseInsLIST.size() > 0)

            } // END if (caseScopeLIST.size() > 0)
        } catch (exception e) {
            for (string caseID : triggerNewMAP.keySet()) {
              triggerNewMAP.get(caseID).addError('Unexpected Error creating Scheduled Case for Case ID: ' +
e.getMessage());
            } // END for (string caseID : triggerNewMAP.keySet())
        } // END try

    } // END updateWorkOrders

} // END MaintenanceRequestHelper
```

MaintenanceRequestHelperTest

```
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);
```

Apex Specialist Superbadge:

```
            }
        }
    }

    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);
            }
```

Apex Specialist Superbadge:

```
            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;
    }
  }
}
```

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 {
    2       private static final String WAREHOUSE_URL = 'https://th-

    3

1.      //class that makes a REST callout to an externalwarehouse system to get a
```

Apex Specialist Superbadge:

```
list of equipment that needs to beupdated.
2.          //The callout's JSON response returns the equipmentrecords that you upsert
in Salesforce.
6


1. @future(callout=true)

2. public static void runWarehouseEquipmentSync(){

3. Http http = new Http();

4. HttpRequest request = newHttpRequest(); 11

1. request.setEndpoint(WAREHOUSE_URL);

2. request.setMethod('GET');

3. HttpResponse response =http.send(request); 15
16                  List<Product2> warehouseEq = newList<Product2>(); 17

1. if (response.getStatusCode() == 200){

2.                  List<Object> jsonResponse =
(List<Object>)JSON.deserializeUntyped(response.getBody());
3. System.debug(response.getBody()); 21

1.                  //class maps the following fields: replacementpart (always
true), cost, current inventory, lifespan, maintenance cycle, and warehouse SKU

2.                  //warehouse SKU will be external ID foridentifying which
equipment records to update within Salesforce

3. for (Object eq : jsonResponse){

4.                      Map<String,Object> mapJson =
(Map<String,Object>)eq;

5. Product2 myEq = new Product2();

6.                      myEq.Replacement_Part_____c
=(Boolean) mapJson.get('replacement');

7. myEq.Name = (String) mapJson.get('name');

8.                      myEq.Maintenance_Cycle_____c
=(Integer) mapJson.get('maintenanceperiod');

9.                      myEq.Lifespan_Months_____c
```

Apex Specialist Superbadge:

```
        =(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);
                }
```

Step 4 - Schedule Synchronizafion:

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

WarehouseSyncSchdeule

```
1. global with sharing class WarehouseSyncSchedule implements
   Schedulable{
2. global void execute(SchedulableContext ctx){
3. System.enqueueJob(new WarehouseCalloutService()); 4
4.     }
5  }
```

Step 5 - Test automafion logic :

Modify the Apex Classes as below, save and runall.

MaintenanceRequestHelper

```
public class MaintenanceRequestHelper {



   public static void updateWorkOrders(List<Case>

updatedCases, Map<Id,Case> oldCaseMap){
```

Apex Specialist Superbadge:

```
        // updatedCases <= Trigger.New
        // oldCaseMap <= Trigger.oldMap


        // we only need to create new Routine Maintenance
cases if
        // 1. the case has been updated to 'Closed'
        // 2. the case Type is either 'Repair' or 'Routine
Maintenance'
        Set<Id> validCaseIds = new Set<Id>(); // holds all Case
Ids that need to be touched
        for (Case c: updatedCases) {
            // continue only if the Case status has just been
updated to 'Closed'
            if (oldCaseMap.get(c.Id).Status != 'Closed' &&
c.Status == 'Closed') {
                // continue only if the Type is 'Repair' or 'Routine
Maintenance'
                if (c.Type == 'Repair' || c.Type == 'Routine
Maintenance') {
```

Apex Specialist Superbadge:

```
                    // add the Case to the map of valid cases that
    need to be touched
                    validCaseIds.add(c.Id);
                }
            }
        }
        // continue only if there's something to do
        if (!validCaseIds.isEmpty()) {
            // create a list to hold all the new Cases
            List<Case> newCases = new List<Case>();
Date_Reported__c = Date.today());

            // if there are no Work Parts, there won't be a minimum value to assign to the new Case
            // the req'ts don't specify what to do in this case; it seems reasonable to use the cycle
from Equipment__c
            if (maintCycleMap.containsKey(cc.Id) ) {
                nc.Date_Due__c = Date.today().addDays((Integer) maintCycleMap.get(cc.Id));
            } else {
                nc.Date_Due__c = Date.today().addDays((Integer)
cc.Equipment__r.Maintenance_Cycle__c);
            }

            newCases.add(nc);
        }

        // insert the new Cases
        insert newCases;
```

Apex Specialist Superbadge:

```
        // clone the Work Parts and assign them to the new Case
        List<Work_Part__c> clonedWorkParts = new List<Work_Part__c>();
        for (Case nc: newCases) {
          // for each new Case, clone all the Work Parts from the parent case, and assign them to
the new Case
          for (Work_Part__c wp: closedCaseMap.get(nc.ParentId).Work_Parts__r) {
            Work_Part__c wpClone = wp.clone();
            wpClone.Maintenance_Request__c = nc.Id;
            clonedWorkParts.add(wpClone);
          }
        }

        // insert the cloned work parts
        insert clonedWorkParts;
      }
    }
}
```

MaintenanceRequestTest.apxc

```
@isTest
private class MaintenanceRequestTest{
  @testSetup
  static void setup(){
    //Equipment SETUP
    List<Product2> lstOfEqpmnts = new List<Product2>();

    Product2 eqip = new Product2(Name = 'Test Equipment',
                                 Maintenance_Cycle__c = 10,
                                 Cost__c = 100,
                                 Current_Inventory__c = 10,
                                 Lifespan_Months__c = 10,
                                 Replacement_Part__c = true,
```

Apex Specialist Superbadge:

```
                        Warehouse_SKU__c = 'abc');
    lstOfEqpmnts.add(eqip);
    INSERT lstOfEqpmnts;
}

@isTest
static void testMaintenanceRequest(){
    List<Case> lstOfInsertMRs = new List<Case>();

    List<Case> lstOfUpdtMRs = new List<Case>();

    Id equipId = [SELECT Id FROM Product2 LIMIT 1].get(0).Id;

    Case newCase = new Case(Type = 'Routine Maintenance',Status = 'New', Origin = 'Phone');
    newCase.Equipment__c = equipId;
    lstOfInsertMRs.add(newCase);

   Case mrInsert = new Case(Type = 'Routine Maintenance2', Status = 'New', Origin = 'Phone');
   mrInsert.Equipment__c = equipId;
   lstOfInsertMRs.add(mrInsert);


    Test.startTest();
        INSERT lstOfInsertMRs;

        System.assertEquals(2, lstOfInsertMRs.size());

        for(Case mrUpdt : lstOfInsertMRs){
            mrUpdt.Status = 'Closed';
            lstOfUpdtMRs.add(mrUpdt);
        }

        UPDATE lstOfUpdtMRs;
```

```
        System.assertEquals(2, lstOfUpdtMRs.size());
      Test.stopTest();
   }
}
```

Step 6 - Test callout logic :

Modify the Apex Classes as below, save

and run all.

WarehouseCalloutServiceTest

```
@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

Apex Specialist Superbadge:

```
@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
```

Apex Specialist Superbadge:

```
kW","maintenanceperiod":365,"lifespan":120,"cost":50

00,"sku":"100003"}]');

        response.setStatusCode(200);

        return response;

    }

}
```

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(){

        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 =
```

Apex Specialist Superbadge:

```
(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);
            }

        }
    }
}
```

Step 7 - Test scheduling logic :

Apex Specialist Superbadge:

Modify the Apex Classes as below, save

and run all.

WarehouseSyncSchedule:

```
global class WarehouseSyncSchedule implements
Schedulable {
  global void execute(SchedulableContext ctx) {


WarehouseCalloutService.runWarehouseEquipmentS
ync();
  }
}
```

WarehouseSyncScheduleTest

```
@isTest
public class WarehouseSyncScheduleTest {
  @isTest static void WarehousescheduleTest(){
    String scheduleTime = '00 00 01 * * ?';
```

Apex Specialist Superbadge:

```
        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 ');


    }
```

Apex Specialist Superbadge:

```
    }
```