Apex Specialist Superbadge:

In this Apex Specialist superbadge firstly we need to complete apex triggers in order to access this superbadge, After that the initial step is to create a new playground. Now the steps which are mentioned in set up development org has to be done. Then according to the given process, write the code for each step mentioned below:

Step 1 : Answer the given multiple choice questions correctly.

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:

Maintanence Request Helper:

```
1  public with sharing class MaintenanceRequestHelper {
2      public static void updateworkOrders(List<Case>
   updWorkOrders, Map<Id,Case> nonUpdCaseMap) {
3          Set<Id> validIds = new Set<Id>();
4          For (Case c : updWorkOrders){
5            if (nonUpdCaseMap.get(c.Id).Status != 'Closed'
   c.Status == 'Closed'){
6                  if (c.Type == 'Repair' || c.Type ==
   'Routine Maintenance'){
7                      validIds.add(c.Id);
8                  }
9              }
10         }
11
12         //When an existing maintenance request of type
   Repair or Routine Maintenance is closed,
13         //create a new maintenance request for a future
   routine checkup.
14         if (!validIds.isEmpty()){
15             Map<Id,Case> closedCases = new
   Map<Id,Case>([SELECT Id, Vehicle__c, Equipment__c,
   Equipment__r.Maintenance_Cycle__c,
16
   (SELECT Id,Equipment__c,Quantity__c FROM
```

```
Equipment_Maintenance_Items__r)
17
FROM Case WHERE Id IN :validIds]);
18              Map<Id,Decimal> maintenanceCycles = new
Map<ID,Decimal>();
19
20              //calculate the maintenance request due dates
by using the maintenance cycle defined on the related
equipment records.
21              AggregateResult[] results = [SELECT
Maintenance_Request__c,
22
MIN(Equipment__r.Maintenance_Cycle__c)cycle
23                                              FROM
Equipment_Maintenance_Item__c
24                                              WHERE
Maintenance_Request__c IN :ValidIds GROUP BY
Maintenance_Request__c];
25
26              for (AggregateResult ar : results){
27                  maintenanceCycles.put((Id)
ar.get('Maintenance_Request__c'), (Decimal)
ar.get('cycle'));
28              }
29
30              List<Case> newCases = new List<Case>();
31              for(Case cc : closedCases.values()){
32                  Case nc = new Case (
33                      ParentId = cc.Id,
34                      Status = 'New',
35                      Subject = 'Routine Maintenance',
36                      Type = 'Routine Maintenance',
37                      Vehicle__c = cc.Vehicle__c,
38                      Equipment__c =cc.Equipment__c,
39                      Origin = 'Web',
40                      Date_Reported__c = Date.Today()
41                  );
```

```
42
43                 //If multiple pieces of equipment are used
   in the maintenance request,
44                 //define the due date by applying the
   shortest maintenance cycle to today's date.
45                 //If
   (maintenanceCycles.containskey(cc.Id)){
46                     nc.Date_Due__c =
   Date.today().addDays((Integer)
   maintenanceCycles.get(cc.Id));
47                 //} else {
48                 //    nc.Date_Due__c =
   Date.today().addDays((Integer)
   cc.Equipment__r.maintenance_Cycle__c);
49                 //}
50
51                 newCases.add(nc);
52             }
53
54             insert newCases;
55
56             List<Equipment_Maintenance_Item__c> clonedList
   = new List<Equipment_Maintenance_Item__c>();
57             for (Case nc : newCases){
58                 for (Equipment_Maintenance_Item__c
   clonedListItem :
   closedCases.get(nc.ParentId).Equipment_Maintenance_Items__r
   ){
59                     Equipment_Maintenance_Item__c item =
   clonedListItem.clone();
60                     item.Maintenance_Request__c = nc.Id;
61                     clonedList.add(item);
62                 }
63             }
64             insert clonedList;
65         }
```

```
    }
}
```

MaintenanceRequestHelperTest:

```
@isTest
public with sharing class MaintenanceRequestHelperTest {

    // createVehicle
    private static Vehicle__c createVehicle(){
        Vehicle__c vehicle = new Vehicle__C(name = 'Testing

        return vehicle;
    }

    // createEquipment
    private static Product2 createEquipment(){
        product2 equipment = new product2(name = 'Testing

lifespan_months__c = 10,
maintenance_cycle__c = 10,
replacement_part__c = true);
        return equipment;
    }

    // createMaintenanceRequest
    private static Case createMaintenanceRequest(id
vehicleId, id equipmentId){
        case cse = new case(Type='Repair',
                            Status='New',
                            Origin='Web',
                            Subject='Testing subject',
```

```
                            Equipment__c=equipmentId,
                            Vehicle__c=vehicleId);
        return cse;
    }

    // createEquipmentMaintenanceItem
    private static Equipment_Maintenance_Item__c
createEquipmentMaintenanceItem(id equipmentId,id
requestId){
        Equipment_Maintenance_Item__c
equipmentMaintenanceItem = new
Equipment_Maintenance_Item__c(
            Equipment__c = equipmentId,
            Maintenance_Request__c = requestId);
        return equipmentMaintenanceItem;
    }

    @isTest
    private static void testPositive(){
        Vehicle__c vehicle = createVehicle();
        insert vehicle;
        id vehicleId = vehicle.Id;

        Product2 equipment = createEquipment();
        insert equipment;
        id equipmentId = equipment.Id;

        case createdCase =
createMaintenanceRequest(vehicleId,equipmentId);
        insert createdCase;

        Equipment_Maintenance_Item__c
equipmentMaintenanceItem =
createEquipmentMaintenanceItem(equipmentId,createdCase.id);
        insert equipmentMaintenanceItem;

```

```
54          test.startTest();
55          createdCase.status = 'Closed';
56          update createdCase;
57          test.stopTest();
58
59          Case newCase = [Select id,
60                          subject,
61                          type,
62                          Equipment__c,
63                          Date_Reported__c,
64                          Vehicle__c,
65                          Date_Due__c
66                         from case
67                         where status ='New'];
68
69          Equipment_Maintenance_Item__c workPart = [select id
70                                                    from
    Equipment_Maintenance_Item__c
71                                                    where
    Maintenance_Request__c =:newCase.Id];
72          list<case> allCase = [select id from case];
73          system.assert(allCase.size() == 2);
74
75          system.assert(newCase != null);
76          system.assert(newCase.Subject != null);
77          system.assertEquals(newCase.Type, 'Routine

78          SYSTEM.assertEquals(newCase.Equipment__c,
    equipmentId);
79          SYSTEM.assertEquals(newCase.Vehicle__c, vehicleId);
80          SYSTEM.assertEquals(newCase.Date_Reported__c,
    system.today());
81      }
82
83      @isTest
84      private static void testNegative(){
```

```
85          Vehicle__C vehicle = createVehicle();
86          insert vehicle;
87          id vehicleId = vehicle.Id;
88
89          product2 equipment = createEquipment();
90          insert equipment;
91          id equipmentId = equipment.Id;
92
93          case createdCase =
  createMaintenanceRequest(vehicleId,equipmentId);
94          insert createdCase;
95
96          Equipment_Maintenance_Item__c workP =
  createEquipmentMaintenanceItem(equipmentId,
  createdCase.Id);
97          insert workP;
98
99          test.startTest();
100           createdCase.Status = 'Working';
101           update createdCase;
102           test.stopTest();
103
104           list<case> allCase = [select id from case];
105
106           Equipment_Maintenance_Item__c
  equipmentMaintenanceItem = [select id
107                                                   from
  Equipment_Maintenance_Item__c
108                                                   where
  Maintenance_Request__c = :createdCase.Id];
109
110           system.assert(equipmentMaintenanceItem != null);
111           system.assert(allCase.size() == 1);
112       }
113
114       @isTest
```

```
115      private static void testBulk(){
116          list<Vehicle__C> vehicleList = new
   list<Vehicle__C>();
117          list<Product2> equipmentList = new
   list<Product2>();
118          list<Equipment_Maintenance_Item__c>
   equipmentMaintenanceItemList = new
   list<Equipment_Maintenance_Item__c>();
119          list<case> caseList = new list<case>();
120          list<id> oldCaseIds = new list<id>();
121
122          for(integer i = 0; i < 300; i++){
123              vehicleList.add(createVehicle());
124              equipmentList.add(createEquipment());
125          }
126          insert vehicleList;
127          insert equipmentList;
128
129          for(integer i = 0; i < 300; i++){
130
   caseList.add(createMaintenanceRequest(vehicleList.get(i).i

131          }
132          insert caseList;
133
134          for(integer i = 0; i < 300; i++){
135
   equipmentMaintenanceItemList.add(createEquipmentMaintenance

136          }
137          insert equipmentMaintenanceItemList;
138
139          test.startTest();
140          for(case cs : caseList){
141              cs.Status = 'Closed';
142              oldCaseIds.add(cs.Id);
```

```
        }
        update caseList;
        test.stopTest();

        list<case> newCase = [select id
                              from case
                              where status ='New'];


        list<Equipment_Maintenance_Item__c> workParts =
  [select id
  from Equipment_Maintenance_Item__c
  where Maintenance_Request__c in: oldCaseIds];

        system.assert(newCase.size() == 300);

        list<case> allCase = [select id from case];
        system.assert(allCase.size() == 600);
    }
}
```

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 {
    private static final String WAREHOUSE_URL =
'https://th-superbadge-apex.herokuapp.com/equipment';

    //Write a class that makes a REST callout to an
external warehouse system to get a list of equipment that
needs to be updated.
```

```
5      //The callout's JSON response returns the equipment
   records that you upsert in Salesforce.
6
7      @future(callout=true)
8      public static void runWarehouseEquipmentSync(){
9          System.debug('go into runWarehouseEquipmentSync');
10         Http http = new Http();
11         HttpRequest request = new HttpRequest();
12
13         request.setEndpoint(WAREHOUSE_URL);
14         request.setMethod('GET');
15         HttpResponse response = http.send(request);
16
17         List<Product2> product2List = new List<Product2>();
18         System.debug(response.getStatusCode());
19         if (response.getStatusCode() == 200){
20             List<Object> jsonResponse =
   (List<Object>)JSON.deserializeUntyped(response.getBody());
21             System.debug(response.getBody());
22
23             //class maps the following fields:
24             //warehouse SKU will be external ID for
   identifying which equipment records to update within
   Salesforce
25             for (Object jR : jsonResponse){
26                 Map<String,Object> mapJson =
   (Map<String,Object>)jR;
27                 Product2 product2 = new Product2();
28                 //replacement part (always true),
29                 product2.Replacement_Part__c = (Boolean)
   mapJson.get('replacement');
30                 //cost
31                 product2.Cost__c = (Integer)
   mapJson.get('cost');
32                 //current inventory
33                 product2.Current_Inventory__c = (Double)
   mapJson.get('quantity');
```

```
                //lifespan
                product2.Lifespan_Months__c = (Integer)
  mapJson.get('lifespan');
                //maintenance cycle
                product2.Maintenance_Cycle__c = (Integer)
  mapJson.get('maintenanceperiod');
                //warehouse SKU
                product2.Warehouse_SKU__c = (String)
  mapJson.get('sku');

                product2.Name = (String)
  mapJson.get('name');
                product2.ProductCode = (String)
  mapJson.get('_id');
                product2List.add(product2);
            }

            if (product2List.size() > 0){
                upsert product2List;
                System.debug('Your equipment was synced

            }
        }
    }

    public static void execute (QueueableContext context){
        System.debug('start runWarehouseEquipmentSync');
        runWarehouseEquipmentSync();
        System.debug('end runWarehouseEquipmentSync');
    }

}
```

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 ctx) {
        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> 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);
                }
            }
        }

        //When an existing maintenance request of type Repair or Routine Maintenance is closed,
        //create a new maintenance request for a future routine checkup.
        if (!validIds.isEmpty()){
            Map<Id,Case> closedCases = 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)
17
   FROM Case WHERE Id IN :validIds]);
18             Map<Id,Decimal> maintenanceCycles = new
   Map<ID,Decimal>();
19
20             //calculate the maintenance request due dates
   by using the maintenance cycle defined on the related
   equipment records.
21             AggregateResult[] results = [SELECT
   Maintenance_Request__c,
22
   MIN(Equipment__r.Maintenance_Cycle__c)cycle
23                                          FROM
   Equipment_Maintenance_Item__c
24                                          WHERE
   Maintenance_Request__c IN :ValidIds GROUP BY
   Maintenance_Request__c];
25
26             for (AggregateResult ar : results){
27                 maintenanceCycles.put((Id)
   ar.get('Maintenance_Request__c'), (Decimal)
   ar.get('cycle'));
28             }
29
30             List<Case> newCases = new List<Case>();
31             for(Case cc : closedCases.values()){
32                 Case nc = new Case (
33                     ParentId = cc.Id,
34                     Status = 'New',
35                     Subject = 'Routine Maintenance',
36                     Type = 'Routine Maintenance',
37                     Vehicle__c = cc.Vehicle__c,
38                     Equipment__c =cc.Equipment__c,
39                     Origin = 'Web',
40                    Date_Reported__c = Date.Today()
```

```
41                );
42
43                //If multiple pieces of equipment are used
  in the maintenance request,
44                //define the due date by applying the
  shortest maintenance cycle to today’s date.
45                //If
  (maintenanceCycles.containskey(cc.Id)){
46                    nc.Date_Due__c =
  Date.today().addDays((Integer)
  maintenanceCycles.get(cc.Id));
47                //} else {
48                //    nc.Date_Due__c =
  Date.today().addDays((Integer)
  cc.Equipment__r.maintenance_Cycle__c);
49                //}
50
51                newCases.add(nc);
52            }
53
54            insert newCases;
55
56            List<Equipment_Maintenance_Item__c> clonedList
  = new List<Equipment_Maintenance_Item__c>();
57            for (Case nc : newCases){
58                for (Equipment_Maintenance_Item__c
  clonedListItem :
  closedCases.get(nc.ParentId).Equipment_Maintenance_Items__r
  ){
59                    Equipment_Maintenance_Item__c item =
  clonedListItem.clone();
60                    item.Maintenance_Request__c = nc.Id;
61                    clonedList.add(item);
62                }
63            }
64            insert clonedList;
```

```
        }
    }
}
```

MaintenanceRequestHelperTest:

```
@isTest
public with sharing class MaintenanceRequestHelperTest {

    // createVehicle
    private static Vehicle__c createVehicle(){
        Vehicle__c vehicle = new Vehicle__C(name = 'Testing

        return vehicle;
    }

    // createEquipment
    private static Product2 createEquipment(){
        product2 equipment = new product2(name = 'Testing


lifespan_months__c = 10,

maintenance_cycle__c = 10,

replacement_part__c = true);
        return equipment;
    }

    // createMaintenanceRequest
    private static Case createMaintenanceRequest(id
vehicleId, id equipmentId){
        case cse = new case(Type='Repair',
```

```
                                Status='New',
                                Origin='Web',
                                Subject='Testing subject',
                                Equipment__c=equipmentId,
                                Vehicle__c=vehicleId);
        return cse;
    }

    // createEquipmentMaintenanceItem
    private static Equipment_Maintenance_Item__c
createEquipmentMaintenanceItem(id equipmentId,id
requestId){
        Equipment_Maintenance_Item__c
equipmentMaintenanceItem = new
Equipment_Maintenance_Item__c(
            Equipment__c = equipmentId,
            Maintenance_Request__c = requestId);
        return equipmentMaintenanceItem;
    }

    @isTest
    private static void testPositive(){
        Vehicle__c vehicle = createVehicle();
        insert vehicle;
        id vehicleId = vehicle.Id;

        Product2 equipment = createEquipment();
        insert equipment;
        id equipmentId = equipment.Id;

        case createdCase =
createMaintenanceRequest(vehicleId,equipmentId);
        insert createdCase;

        Equipment_Maintenance_Item__c
equipmentMaintenanceItem =
createEquipmentMaintenanceItem(equipmentId,createdCase.id);
```

```
52          insert equipmentMaintenanceItem;
53
54          test.startTest();
55          createdCase.status = 'Closed';
56          update createdCase;
57          test.stopTest();
58
59          Case newCase = [Select id,
60                          subject,
61                          type,
62                          Equipment__c,
63                          Date_Reported__c,
64                          Vehicle__c,
65                          Date_Due__c
66                         from case
67                         where status ='New'];
68
69          Equipment_Maintenance_Item__c workPart = [select id
70                                                    from
    Equipment_Maintenance_Item__c
71                                                    where
    Maintenance_Request__c =:newCase.Id];
72          list<case> allCase = [select id from case];
73          system.assert(allCase.size() == 2);
74
75          system.assert(newCase != null);
76          system.assert(newCase.Subject != null);
77          system.assertEquals(newCase.Type, 'Routine

78          SYSTEM.assertEquals(newCase.Equipment__c,
    equipmentId);
79          SYSTEM.assertEquals(newCase.Vehicle__c, vehicleId);
80          SYSTEM.assertEquals(newCase.Date_Reported__c,
    system.today());
81      }
82
```

```
83      @isTest
84      private static void testNegative(){
85          Vehicle__C vehicle = createVehicle();
86          insert vehicle;
87          id vehicleId = vehicle.Id;
88
89          product2 equipment = createEquipment();
90          insert equipment;
91          id equipmentId = equipment.Id;
92
93          case createdCase =
    createMaintenanceRequest(vehicleId,equipmentId);
94          insert createdCase;
95
96          Equipment_Maintenance_Item__c workP =
    createEquipmentMaintenanceItem(equipmentId,
    createdCase.Id);
97          insert workP;
98
99          test.startTest();
100          createdCase.Status = 'Working';
101          update createdCase;
102          test.stopTest();
103
104          list<case> allCase = [select id from case];
105
106          Equipment_Maintenance_Item__c
    equipmentMaintenanceItem = [select id
107                                                  from
    Equipment_Maintenance_Item__c
108                                                  where
    Maintenance_Request__c = :createdCase.Id];
109
110          system.assert(equipmentMaintenanceItem != null);
111          system.assert(allCase.size() == 1);
112      }
```

```
113
114     @isTest
115     private static void testBulk(){
116         list<Vehicle__C> vehicleList = new
    list<Vehicle__C>();
117         list<Product2> equipmentList = new
    list<Product2>();
118         list<Equipment_Maintenance_Item__c>
    equipmentMaintenanceItemList = new
    list<Equipment_Maintenance_Item__c>();
119         list<case> caseList = new list<case>();
120         list<id> oldCaseIds = new list<id>();
121
122         for(integer i = 0; i < 300; i++){
123             vehicleList.add(createVehicle());
124             equipmentList.add(createEquipment());
125         }
126         insert vehicleList;
127         insert equipmentList;
128
129         for(integer i = 0; i < 300; i++){
130
    caseList.add(createMaintenanceRequest(vehicleList.get(i).i

131         }
132         insert caseList;
133
134         for(integer i = 0; i < 300; i++){
135
    equipmentMaintenanceItemList.add(createEquipmentMaintenance

136 _       }
137         insert equipmentMaintenanceItemList;
138
139         test.startTest();
140         for(case cs : caseList){
```

```
            cs.Status = 'Closed';
            oldCaseIds.add(cs.Id);
        }
        update caseList;
        test.stopTest();

        list<case> newCase = [select id
                              from case
                              where status ='New'];


        list<Equipment_Maintenance_Item__c> workParts =
  [select id
  from Equipment_Maintenance_Item__c
  where Maintenance_Request__c in: oldCaseIds];

        system.assert(newCase.size() == 300);

        list<case> allCase = [select id from case];
        system.assert(allCase.size() == 600);
    }
}
```

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 testWarehouseCallout() {
```

```
        test.startTest();
        test.setMock(HttpCalloutMock.class, new
  WarehouseCalloutServiceMock());
        WarehouseCalloutService.execute(null);
        test.stopTest();

        List<Product2> product2List = new List<Product2>();
        product2List = [SELECT ProductCode FROM Product2];

        System.assertEquals(3, product2List.size());
        System.assertEquals('55d66226726b611100aaf741',
  product2List.get(0).ProductCode);
        System.assertEquals('55d66226726b611100aaf742',
  product2List.get(1).ProductCode);
        System.assertEquals('55d66226726b611100aaf743',
  product2List.get(2).ProductCode);
    }
}
```

WarehouseCalloutServiceMock:

```
@isTest
global class WarehouseCalloutServiceMock implements
  HttpCalloutMock {
    // implement http mock callout
    global static HttpResponse respond(HttpRequest request) {

        HttpResponse response = new HttpResponse();
        response.setHeader('Content-Type', 'application/json');

  response.setBody('[{"_id":"55d66226726b611100aaf741","replacement
```

```
            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 ctx) {
        System.enqueueJob(new WarehouseCalloutService());
    }
}
```

WarehouseSyncScheduleTest:

```

```