Apex Specialist Superbadge:

In this superbadge, 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 : 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> 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
                                  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   Id

```
    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
```

```
        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;
}
}
```

MaintenanceRequestHelperTest

```
@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
private static final string REQUEST_SUBJECT = 'Testing
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_partc =
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
```

```

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
Equic
 where
Maintenance_Request_c =:newReq.Id];

 system.assert(workPart != null);
```

assert 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_
workPart = [select id
from Equipment_Maintenance_Itemc
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);
}
}
```

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-
        //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

}
}
}

public static void execute (QueueableContext context){
runWarehouseEquipmentSync();
}
```

Step 4 - Schedule Synchronization:

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

WarehouseSyncSchdeule

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

```

Step 5 - Test automationlogic :

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
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() 37
 );

 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
```

```
wp.clone();
wpClone.Maintenance_Request_c = nc.Id; ClonedWPs.add(wpClone);

}
}
insert ClonedWPs;
}
```

MaintenanceRequestHelperTest

```
@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
private static final string REQUEST_SUBJECT = 'Testing
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; 47
 Product2 equipment = createEq();
 insert equipment;
 id equipmentId = equipment.Id; 51
       case somethingToUpdate =
createMaintenanceRequest(vehicleId,equipmentId);
 insert somethingToUpdate; 54
       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_
 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_
workPart = [select id
from Equipment_Maintenance_Item_
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_
where Maintenance_Request_c in: oldRequestIds];
system.assert(allRequests.size()== 300);
}
}

```

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.execute(null);
test.stopTest();
System.assertEquals(1, [SELECT count() FROM Product2]);
```

WarehouseCalloutServiceMock

```
 @isTest
2global class WarehouseCalloutServiceMock implements
HttpCalloutMock {
 // implement http mock callout
 global static HttpResponse respond(HttpRequest request){
 System.assertEquals('https://th-superbadge-
));
 System.assertEquals('GET', request.getMethod());
 // Create a fake response
 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{
 global void execute(SchedulableContext ctx){
System.enqueueJob(new WarehouseCalloutService());
}
}
```

WarehouseSyncScheduleTest

```
 @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
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 Id 'Schedule '

```
}
System.assertEquals(jobID, a.Id,'Schedule ');

```