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> caseList) {
List<case> newCases = new List<Case>();
Map<String,Integer> result=getDueDate(caseList);
for(Case c : caseList){
if(c.status=='closed')
if(c.type=='Repair' || c.type=='Routine Maintenance'){
Case newCase = new Case();
newCase.Status='New';
newCase.Origin='web';
```

```
11 newCase.Type='Routine Maintenance';

12 newCase.Subject='Routine Maintenance of Vehicle';

13 newCase.Vehicle__c=c.Vehicle__c;

14 newCase.Equipment__c=c.Equipment__c;

15 newCase.Date_Reported__c=Date.today();

16 if(result.get(c.Id)!=null)

17 newCase.Date_Due__c=Date.today()+result.get(c.Id);

18 else

19 newCase.Date_Due__c=Date.today();

20 newCases.add(newCase);

21 }

22 }

23 insert newCases;

24 }

25 //

26 public static  Map<String,Integer> getDueDate(List<case>
   CaseIDs){

27 Map<String,Integer> result = new Map<String,Integer>();
```

```
28 Map<Id, case> caseKeys = new Map<Id, case> (CaseIDs);

29 List<AggregateResult> wpc=[select Maintenance_Request__r.ID
   cID,min(Equipment__r.Maintenance_Cycle__c)cycle

30 from Work_Part__c where  Maintenance_Request__r.ID in
   :caseKeys.keySet() group by
   Maintenance_Request__r.ID ];

31 for(AggregateResult res :wpc){

32 Integer addDays=0;

33 if(res.get('cycle')!=null)

34 addDays+=Integer.valueOf(res.get('cycle'));

35 result.put((String)res.get('cID'),addDays);

36 }

37 return result;

38 }

39 }
```

MaintenanceRequestHelperTest

```
1 @IsTest

2 private class InstallationTests {
```

```
3 private static final String STRING_TEST = 'TEST';

4 private static final String NEW_STATUS = 'New';

5 private static final String WORKING = 'Working';

6 private static final String CLOSED = 'Closed';

7 private static final String REPAIR = 'Repair';

8 private static final String REQUEST_ORIGIN = 'Web';

9 private static final String REQUEST_TYPE = 'Routine


10private static final String REQUEST_SUBJECT = 'AMC


11public static String CRON_EXP = '0 0 1 * * ?';

12static testmethod void
  testMaintenanceRequestNegative() {

13Vehicle__c vehicle = createVehicle();

14insert vehicle;

15Id vehicleId = vehicle.Id;

16Product2 equipment = createEquipment();
```

```
insert equipment;

Id equipmentId = equipment.Id;

Case r = createMaintenanceRequest(vehicleId,
  equipmentId);

insert r;

Work_Part__c w = createWorkPart(equipmentId, r.Id);

insert w;

Test.startTest();

r.Status = WORKING;

update r;

Test.stopTest();

List<case> allRequest = [SELECT Id

FROM Case];

Work_Part__c workPart = [SELECT Id

FROM Work_Part__c

WHERE Maintenance_Request__c =: r.Id];
```

```
32System.assert(workPart != null);

33System.assert(allRequest.size() == 1);

34}

35static testmethod void testWarehouseSync() {

36Test.setMock(HttpCalloutMock.class, new
  WarehouseCalloutServiceMock());

37Test.startTest();

38String jobId =
  System.schedule('WarehouseSyncSchedule',

39CRON_EXP,

40new WarehouseSyncSchedule());

41CronTrigger ct = [SELECT Id, CronExpression,
  TimesTriggered, NextFireTime

42FROM CronTrigger

43WHERE id = :jobId];

44System.assertEquals(CRON_EXP, ct.CronExpression);

45System.assertEquals(0, ct.TimesTriggered);
```

```
Test.stopTest();
}
private static Vehicle__c createVehicle() {
Vehicle__c v = new Vehicle__c(Name = STRING_TEST);
return v;
}
private static Product2 createEquipment() {
Product2 p = new Product2(Name = STRING_TEST,
Lifespan_Months__c = 10,
Maintenance_Cycle__c = 10,
Replacement_Part__c = true);
return p;
}
private static Case createMaintenanceRequest(Id
  vehicleId, Id equipmentId) {
Case c = new Case(Type = REPAIR,
```

```
61Status = NEW_STATUS,

62Origin = REQUEST_ORIGIN,

63Subject = REQUEST_SUBJECT,

64Equipment__c = equipmentId,

65Vehicle__c = vehicleId);

66return c;

67}

68private static Work_Part__c createWorkPart(Id
  equipmentId, Id requestId) {

69Work_Part__c wp = new Work_Part__c(Equipment__c =
  equipmentId,

70Maintenance_Request__c = requestId);

71return wp;

72}

73}
```

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

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

WarehouseCalloutService

```
1 public with sharing class WarehouseCalloutService
  implements Queueable {
2     private static final String WAREHOUSE_URL =
  'https://th-superbadge-

3
4     //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         Http http = new Http();
10        HttpRequest request = new HttpRequest();
11
12        request.setEndpoint(WAREHOUSE_URL);
13        request.setMethod('GET');
14        HttpResponse response = http.send(request);
15
16        List<Product2> warehouseEq = new
  List<Product2>();
17
18        if (response.getStatusCode() == 200){
19            List<Object> jsonResponse =
  (List<Object>)JSON.deserializeUntyped(response.getB

20            System.debug(response.getBody());
```

```
21
22              //class maps the following fields:
  replacement part (always true), cost, current
  inventory, lifespan, maintenance cycle, and
  warehouse SKU
23              //warehouse SKU will be external ID for
  identifying which equipment records to update
  within Salesforce
24              for (Object eq : jsonResponse){
25                  Map<String,Object> mapJson =
  (Map<String,Object>)eq;
26                  Product2 myEq = new Product2();
27                  myEq.Replacement_Part__c =
  (Boolean) mapJson.get('replacement');
28                  myEq.Name = (String)
  mapJson.get('name');
29                  myEq.Maintenance_Cycle__c =
  (Integer) mapJson.get('maintenanceperiod');
30                  myEq.Lifespan_Months__c = (Integer)
  mapJson.get('lifespan');
31                  myEq.Cost__c = (Integer)
  mapJson.get('cost');
32                  myEq.Warehouse_SKU__c = (String)
  mapJson.get('sku');
33                  myEq.Current_Inventory__c =
  (Double) mapJson.get('quantity');
34                  myEq.ProductCode = (String)
  mapJson.get('_id');
35                  warehouseEq.add(myEq);
36              }
37
```

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

            }
        }
    }

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

}


```

Step 4 - Schedule Synchronization:

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

WarehouseSyncSchdeule

```
global  class WarehouseSyncSchedule implements
  Schedulable{
// implement scheduled code here
global  void execute (SchedulableContext sc){
WarehouseCalloutService.runWarehouseEquipmentSync(
  );
//optional this can be done by debug mode
String sch = '00 00 01 * * ?';//on 1 pm
System.schedule('WarehouseSyncScheduleTest', sch,
```

```
    new WarehouseSyncSchedule());
8 }
9 }
```

Step 5 - Test automation logic :

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

MaintenanceRequestHelper

```
1 public with sharing class MaintenanceRequestHelper {

2 public static void updateWorkOrders(List<Case>
  caseList) {

3 List<case> newCases = new List<Case>();

4 Map<String,Integer> result=getDueDate(caseList);

5 for(Case c : caseList){

6 if(c.status=='closed')

7 if(c.type=='Repair' || c.type=='Routine


8 Case newCase = new Case();

9 newCase.Status='New';
```

```
10newCase.Origin='web';

11newCase.Type='Routine Maintenance';

12newCase.Subject='Routine Maintenance of Vehicle';

13newCase.Vehicle__c=c.Vehicle__c;

14newCase.Equipment__c=c.Equipment__c;

15newCase.Date_Reported__c=Date.today();

16if(result.get(c.Id)!=null)

17newCase.Date_Due__c=Date.today()+result.get(c.Id);

18else

19newCase.Date_Due__c=Date.today();

20newCases.add(newCase);

21}

22}

23insert newCases;

24}
```

```
25//

26public static  Map<String,Integer>
  getDueDate(List<case> CaseIDs){

27Map<String,Integer> result = new
  Map<String,Integer>();

28Map<Id, case> caseKeys = new Map<Id, case>
  (CaseIDs);

29List<AggregateResult> wpc=[select
  Maintenance_Request__r.ID
  cID,min(Equipment__r.Maintenance_Cycle__c)cycle

30from Work_Part__c where  Maintenance_Request__r.ID
  in :caseKeys.keySet() group by
  Maintenance_Request__r.ID ];

31for(AggregateResult res :wpc){

32Integer addDays=0;

33if(res.get('cycle')!=null)

34addDays+=Integer.valueOf(res.get('cycle'));

35result.put((String)res.get('cID'),addDays);

36}
```

```
37return result;

38}

39}
```

MaintenanceRequestHelperTest

```
1 @IsTest
2 private class InstallationTests {
3 private static final String STRING_TEST = 'TEST';
4 private static final String NEW_STATUS = 'New';
5 private static final String WORKING = 'Working';
6 private static final String CLOSED = 'Closed';
7 private static final String REPAIR = 'Repair';
8 private static final String REQUEST_ORIGIN = 'Web';
9 private static final String REQUEST_TYPE = 'Routine

10private static final String REQUEST_SUBJECT = 'AMC

11public static String CRON_EXP = '0 0 1 * * ?';
12static testmethod void
  testMaintenanceRequestNegative() {
13Vehicle__c vehicle = createVehicle();
14insert vehicle;
15Id vehicleId = vehicle.Id;
16Product2 equipment = createEquipment();
17insert equipment;
18Id equipmentId = equipment.Id;
19Case r = createMaintenanceRequest(vehicleId,
```

```
  equipmentId);
insert r;
Work_Part__c w = createWorkPart(equipmentId, r.Id);
insert w;
Test.startTest();
r.Status = WORKING;
update r;
Test.stopTest();
List<case> allRequest = [SELECT Id
FROM Case];
Work_Part__c workPart = [SELECT Id
FROM Work_Part__c
WHERE Maintenance_Request__c =: r.Id];
System.assert(workPart != null);
System.assert(allRequest.size() == 1);
}
static testmethod void testWarehouseSync() {
Test.setMock(HttpCalloutMock.class,                new
  WarehouseCalloutServiceMock());
Test.startTest();
String                      jobId                      =
  System.schedule('WarehouseSyncSchedule',
CRON_EXP,
new WarehouseSyncSchedule());
CronTrigger   ct   =   [SELECT   Id,   CronExpression,
  TimesTriggered, NextFireTime
FROM CronTrigger
WHERE id = :jobId];
System.assertEquals(CRON_EXP, ct.CronExpression);
System.assertEquals(0, ct.TimesTriggered);
Test.stopTest();
```

```
47}
48private static Vehicle__c createVehicle() {
49Vehicle__c v = new Vehicle__c(Name = STRING_TEST);
50return v;
51}
52private static Product2 createEquipment() {
53Product2 p = new Product2(Name = STRING_TEST,
54Lifespan_Months__c = 10,
55Maintenance_Cycle__c = 10,
56Replacement_Part__c = true);
57return p;
58}
59private static Case createMaintenanceRequest(Id
  vehicleId, Id equipmentId) {
60Case c = new Case(Type = REPAIR,
61Status = NEW_STATUS,
62Origin = REQUEST_ORIGIN,
63Subject = REQUEST_SUBJECT,
64Equipment__c = equipmentId,
65Vehicle__c = vehicleId);
66return c;
67}
68private static Work_Part__c createWorkPart(Id
  equipmentId, Id requestId) {
69Work_Part__c wp = new Work_Part__c(Equipment__c =
  equipmentId,
70Maintenance_Request__c = requestId);
71return wp;
72}
73}
```

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.setMock(HttpCalloutMock.class, new
WarehouseCalloutServiceMock());
WarehouseCalloutService.runWarehouseEquipmentSync(
);
}
}
```

WarehouseCalloutServiceMock

```
@isTest
public class WarehouseCalloutServiceMock implements
HTTPCalloutMock {
// implement http mock callout
public HTTPResponse respond (HttpRequest request){
HttpResponse response = new HTTPResponse();
response.setHeader('Content-type','application/json');
response.setBody('[{"_id":"55d66226726b611100aaf741","repla
```

```
 8  response.setStatusCode(200);
 9  return response;
 10 }
 11 }
```

Step 7 - Test scheduling logic :

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

WarehouseSyncSchedule

```
1  global with sharing class WarehouseSyncSchedule implements
   Schedulable{

2   global void execute(SchedulableContext ctx){

3   System.enqueueJob(new WarehouseCalloutService());

4  }

5  }

6
```

WarehouseSyncScheduleTest

```
 1 @isTest
```

```
2 private class WarehouseSyncScheduleTest {
3 public static String CRON_EXP = '0 0 0 15 3 ?

4 static testmethod void testjob(){
5 MaintenanceRequestTest.CreateData( 5,2,2,'Repair');
6 Test.startTest();
7 Test.setMock(HttpCalloutMock.class, new
  WarehouseCalloutServiceMock());
8 String joBID= System.schedule('TestScheduleJob',
  CRON_EXP, new WarehouseSyncSchedule());
9 // List<Case> caselist = [Select count(id) from
  case where case]
10Test.stopTest();
11}
12}
```