Apex Specialist

1. **Automate Record Creation**

MaintenanceRequest

```
trigger MaintenanceRequest on Case (before update, after update) {

    Map<Id,Case> validCaseMap = new Map<Id,Case>();

    if(Trigger.isUpdate && Trigger.isAfter){
            for(Case caseHere: Trigger.new){
                    if (caseHere.IsClosed && (caseHere.Type.equals('Repair') ||
                    caseHere.Type.equals('Routine Maintenance'))){
                    validCaseMap.put(caseHere.Id, caseHere);
                    }
            }

            if(!validCaseMap.values().isEmpty()) {
            MaintenanceRequestHelper.createNewRequest(validCaseMap);
            }
    }
}
```

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);
                }
            }
        }

```

```
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      }
66    }
67 }
```

2. **Synchronize Salesforce data with an external system**

```
1 public with sharing class WarehouseCalloutService {
2
3       private static final String WAREHOUSE_URL =
    'https://th-superbadge-apex.herokuapp.com/equipment';
4
5       @future(callout=true)
6       public static void runWarehouseEquipmentSync(){
```

```
7           Http http = new Http();
8           HttpRequest request = new HttpRequest();
9           request.setEndpoint(WAREHOUSE_URL);
10          request.setMethod('GET');1
  HttpResponse response = http.send(request);
11
12          if (response.getStatusCode() == 200) {
13              List<Object> results = (List<Object>)
  JSON.deserializeUntyped(response.getBody());
14              List<Product2> equipmentList = new
  List<Product2>();
15              for (Object record: results) {
16                  Map<String, Object> recordMap =
  (Map<String, Object>)record;
17                  Product2 equipment = new Product2();
18
19                  equipment.Name =
  (String)recordMap.get('name');
20                  equipment.Cost__c =
  (Decimal)recordMap.get('cost');
21                  equipment.ProductCode =
  (String)recordMap.get('_id');
22                  equipment.Current_Inventory__c =
  (Integer)recordMap.get('quantity');
23                  equipment.Maintenance_Cycle__c =
  (Integer)recordMap.get('maintenanceperiod');
24                  equipment.Replacement_Part__c =
  (Boolean)recordMap.get('replacement');
  equipment.Lifespan_Months__c =
  (Integer)recordMap.get('lifespan');
25                  equipment.Warehouse_SKU__c =
  (String)recordMap.get('sku');
```

```
26
27                  equipmentList.add(equipment);
28              }
29
30              if(equipmentList.size() > 0){
31                  upsert equipmentList;
32              }
33          }
34
35      }
36  }
```

Anonymous Window

```
1 WarehouseCalloutService.runWarehouseEquipmentSync();
```

3. Schedule synchronization

Class

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

Anonymous Window

```
1  System.schedule('WarehouseSyncScheduleTest', '0 0 1 * * ?', new
2  WarehouseSyncSchedule());
```

4. Test automation logic

Helper Test Class

```
@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 subject';
1
 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;

 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 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__c workPart = [select id from
 Equipment_Maintenance_Item__c where Maintenance_Request__c =
  :emptyReq.Id];
```

```
112
113  system.assert(workPart != null);
114  system.assert(allRequest.size() == 1);
115  }
116
117  @istest
118  private static void testMaintenanceRequestBulk(){
119  list<Vehicle__C> vehicleList = new list<Vehicle__C>();
120  list<Product2> equipmentList = new list<Product2>();
121  list<Equipment_Maintenance_Item__c> workPartList = new
122 list<Equipment_Maintenance_Item__c>();
123  list<case> requestList = new list<case>();
124  list<id> oldRequestIds = new list<id>();
125
126  for(integer i = 0; i < 300; i++){
127  vehicleList.add(createVehicle());
128  equipmentList.add(createEq());
129  }
130  insert vehicleList;
131  insert equipmentList;
132
133  for(integer i = 0; i < 300; i++){
134  requestList.add(createMaintenanceRequest(vehicleList.get(i).id,
135 equipmentList.get(i).id));
136  }
137  insert requestList;
138
139  for(integer i = 0; i < 300; i++){
140  workPartList.add(createWorkPart(equipmentList.get(i).id,
141 requestList.get(i).id));
142  }
143  insert workPartList;
144
145  test.startTest();
146  for(case req : requestList){
147  req.Status = CLOSED;
148  oldRequestIds.add(req.Id);
149  }
150  update requestList;
151  test.stopTest();
```

```
152
153  list<case> allRequests = [select id
154  from case
155  where status =: STATUS_NEW];
156
157  list<Equipment_Maintenance_Item__c> workParts = [select id
158  from
159 Equipment_Maintenance_Item__c
160  where
161 Maintenance_Request__c in: oldRequestIds];
162  system.assert(allRequests.size() == 300);
163  }
164  }
165
```

test Class

```
1  public with sharing class MaintenanceRequestHelper {
2   public static void updateworkOrders(List<Case> updWorkOrders,
   Map<Id,Case>
3  nonUpdCaseMap) {
4  3 Set<Id> validIds = new Set<Id>();
5
6
7   For (Case c : updWorkOrders){
8   if (nonUpdCaseMap.get(c.Id).Status != 'Closed' && c.Status ==
9  'Closed'){
10  if (c.Type == 'Repair' || c.Type == 'Routine Maintenance'){
11  validIds.add(c.Id);
12
13
14  }
15  }
16  }
17
18  if (!validIds.isEmpty()){
19  List<Case> newCases = new List<Case>();
20  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));
 }

 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;
 }
 }
 }
```

Apex Trigger

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

```

5. Test call out logic

Apex Class

```
public with sharing class WarehouseCalloutService {

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

 public static void runWarehouseEquipmentSync(){

 Http http = new Http();
 HttpRequest request = new HttpRequest();
```

```
8
9   request.setEndpoint(WAREHOUSE_URL);
10  request.setMethod('GET');
11  HttpResponse response = http.send(request);
12
13
14  List<Product2> warehouseEq = new List<Product2>();
15
16  if (response.getStatusCode() == 200){
17  List<Object> jsonResponse =
18 (List<Object>)JSON.deserializeUntyped(response.getBody());
19  System.debug(response.getBody());
20  for (Object eq : jsonResponse){
21  Map<String,Object> mapJson = (Map<String,Object>)eq;
22  Product2 myEq = new Product2();
23  myEq.Replacement_Part__c = (Boolean) mapJson.get('replacement');
24  myEq.Name = (String) mapJson.get('name');
25  myEq.Maintenance_Cycle__c = (Integer)
26 mapJson.get('maintenanceperiod');
27  myEq.Lifespan_Months__c = (Integer) mapJson.get('lifespan');
28  myEq.Cost__c = (Decimal) mapJson.get('lifespan');
29  myEq.Warehouse_SKU__c = (String) mapJson.get('sku');
30  myEq.Current_Inventory__c = (Double) mapJson.get('quantity');
31  warehouseEq.add(myEq);
32  }
33
34  if (warehouseEq.size() > 0){
35  upsert warehouseEq;
36  System.debug('Your equipment was synced with the warehouse

37  System.debug(warehouseEq);
38  }
39  }
40  }
41  }
```

Test Class

```
1  @isTest
```

```
2
3   private class WarehouseCalloutServiceTest {
4   @isTest
5   static void testWareHouseCallout(){
6   Test.startTest();
7   // implement mock callout test here
8   Test.setMock(HTTPCalloutMock.class, new
    WarehouseCalloutServiceMock());
9   WarehouseCalloutService.runWarehouseEquipmentSync();
10  Test.stopTest();
11  System.assertEquals(1, [SELECT count() FROM Product2]);
12  }
13  }
```

Mock Test Class

```
1  @isTest
2   global class WarehouseCalloutServiceMock implements
    HttpCalloutMock {
3   global static HttpResponse respond(HttpRequest request){
4   System.assertEquals('https://th-superbadge-

5  request.getEndpoint());
6   System.assertEquals('GET', request.getMethod());
7   HttpResponse response = new HttpResponse();
8   response.setHeader('Content-Type', 'application/json');
9
10 response.setBody('[{"_id":"55d66226726b611100aaf741","replacement
    ":false,"quantity"
11  response.setStatusCode(200);
12  return response;
13  }
14  }
```

6. Test scheduling logic

Apex Class

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

 WarehouseCalloutService.runWarehouseEquipmentSync();
 }
 }
```

test class

```
@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

scheduleTime, new WarehouseSyncSchedule());
 Test.stopTest();

 CronTrigger a=[SELECT Id FROM CronTrigger where NextFireTime >
today]; System.assertEquals(jobID, a.Id,'Schedule ');


 }
 }
```