SUPER BADGE - APEX SPECIALIST

1. Automate Record Creation

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

{

        if(Trigger.isUpdate  && Trigger.isAfter)

        {

                MaintenanceRequestHelper.updateWorkOrders(Trigger.new);

        }

}


public with sharing class MaintenanceRequestHelper

{

        public static void updateWorkOrders(List<Case> newCases)

        {

                Set<Id> closedCases = new Set<Id>();

                for(Case c : newCases)

                {

                        if(c.Status == 'Closed' && (c.Type == 'Repair' || c.Type == 'Routine Maintenance'))

                        {

                                closedCases.add(c.Id);

                        }

                }


                Map<Id, Decimal> reqToMaintenanceCycleMap = new Map<Id, Decimal>();

                List<AggregateResult> result =

                        [SELECT Maintenance_Request__c, MIN(Equipment__r.Maintenance_Cycle__c) cycle

                        FROM Equipment_Maintenance_Item__c

                        WHERE Maintenance_Request__c IN :closedCases

                        GROUP BY Maintenance_Request__c];
```

```
for(AggregateResult agg : result)
{
        reqToMaintenanceCycleMap.put(
                (Id)agg.get('Maintenance_Request__c'),
                (Decimal)agg.get('cycle'));
}


Map<Id, List<Equipment_Maintenance_Item__c>> requestToEquipmentsMap =
                        new Map<Id, List<Equipment_Maintenance_Item__c>>();
List<Case> newMaintenanceRequests = new List<Case>();

for(Case c : [SELECT Id, Vehicle__c, (SELECT Id, Equipment__c, Quantity__c FROM
Equipment_Maintenance_Items__r) FROM Case WHERE Id IN :closedCases])
{
        requestToEquipmentsMap.put(c.Id, c.Equipment_Maintenance_Items__r);

        Case req = new Case();
        req.ParentId = c.Id;
        req.Vehicle__c = c.Vehicle__c;
        req.Origin = 'Web';
        req.Type = 'Routine Maintenance';
        req.Subject = 'New Routine Maintenance Request';
        req.Date_Reported__c = Date.Today();
        Integer addDays = 0;

        if(reqToMaintenanceCycleMap.containsKey(c.Id))
        {
                addDays = Integer.valueOf(reqToMaintenanceCycleMap.get(c.Id));
        }
        req.Date_Due__c = Date.Today().addDays(addDays);
        newMaintenanceRequests.add(req);
}
```

```
                    insert newMaintenanceRequests;

                    List<Equipment_Maintenance_Item__c> equipmentMaintenanceItems =
                                                                  new List<Equipment_Maintenance_Item__c>();
                    for(Case c : newMaintenanceRequests)
                    {
                              for(Equipment_Maintenance_Item__c item : requestToEquipmentsMap.get(c.ParentId))
                              {
                                        Equipment_Maintenance_Item__c equipment = item.clone();
                                        equipment.Maintenance_Request__c = c.Id;
                                        equipmentMaintenanceItems.add(equipment);
                              }
                    }
                    insert equipmentMaintenanceItems;
          }
}
```

2. Synchronize Salesforce Data with an External System

```
public with sharing class WarehouseCalloutService implements Queueable
{
	private static final String WAREHOUSE_URL = 'https://th-superbadge-apex.herokuapp.com/equipment';

	@future(callout=true)
	public static void syncWarehouseEquipments()
	{
		Http http = new Http();
		HttpRequest request = new HttpRequest();

		request.setEndpoint(WAREHOUSE_URL);
		request.setMethod('GET');
		HttpResponse response = http.send(request);

		List<Product2> equipments = new List<Product2>();

		if (response.getStatusCode() == 200)
		{
			List<Object> jsonResponse =
						(List<Object>)JSON.deserializeUntyped(response.getBody());

			for (Object eq : jsonResponse)
			{
				Map<String,Object> mapJson = (Map<String,Object>)eq;
				Product2 equipment = new Product2();
				equipment.Replacement_Part__c = (Boolean) mapJson.get('replacement');
				equipment.Current_Inventory__c = (Double) mapJson.get('quantity');
				equipment.Name = (String) mapJson.get('name');
				equipment.Maintenance_Cycle__c =
							(Integer) mapJson.get('maintenanceperiod');
				 equipment.Lifespan_Months__c = (Integer) mapJson.get('lifespan');
```

```
                equipment.Cost__c = (Decimal) mapJson.get('cost');

                equipment.Warehouse_SKU__c = (String) mapJson.get('sku');

                equipments.add(equipment);

            }

            if (equipments.size() > 0)

            {

                upsert equipments;

            }

        }

    }

    public static void execute (QueueableContext context)

    {

        syncWarehouseEquipments();

     }

}
```

3. Schedule Synchronization

```
public with sharing class WarehouseSyncSchedule implements Schedulable

{

        public void execute(SchedulableContext context)

        {

                System.enqueueJob(new WarehouseCalloutService());

        }

}
```

4. Test Automation Logic

```
@IsTest

public with sharing class MaintenanceRequestHelperTest

{
        @isTest

        private static void testMaintenanceRequestBulk()

        {
                //Test Vehicle

                Vehicle__c vehicle = new Vehicle__c(Name = 'Super Truck');

                insert vehicle;


                //Test Equipment

                Product2 equipment1 =

                                new Product2(Name = 'Test Equipment', Lifespan_Months__c = 5,

                                Maintenance_Cycle__c = 30, Replacement_Part__c = true);

                Product2 equipment2 =

                                new Product2(Name = 'Test Equipment',Lifespan_Months__c = 3,

                                Maintenance_Cycle__c = 15, Replacement_Part__c = true);

                insert equipment1;

                insert equipment2;


                //Test Maintenance Requests

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


                for(Integer i=0; i<10; i++)

                {
                        Case maintenanceRequest = new Case(Type = 'Repair',

                                                        Status = 'New',

                                                        Subject = 'Routine Maintenance Request' + i,

                                                        Origin = 'Web',

                                                        Vehicle__c = vehicle.id);

                        maintenanceRequests.add(maintenanceRequest);
                }
```

```
insert maintenanceRequests;

List<Equipment_Maintenance_Item__c> items =
                        new List<Equipment_Maintenance_Item__c>();

for(Case c : maintenanceRequests)
{
        Equipment_Maintenance_Item__c item1 =
                        new Equipment_Maintenance_Item__c(
                                Equipment__c = equipment1.Id,
                                Maintenance_Request__c = c.Id,
                                Quantity__c = 10);
        Equipment_Maintenance_Item__c item2 =
                        new Equipment_Maintenance_Item__c(
                        Equipment__c = equipment2.Id,
                        Maintenance_Request__c = c.Id,
                        Quantity__c = 10);
        items.add(item1);
        items.add(item2);
}

insert items;

test.startTest();
for(case req : maintenanceRequests)
{
        req.Status = 'Closed';
}
update maintenanceRequests;
test.stopTest();

List<Case> updatedRequests = [SELECT id FROM Case WHERE Status ='New'];
```

```
                    system.assert(updatedRequests.size() == 10);
          }
}
```

5. Test Callout Logic

```
@IsTest

private class WarehouseCalloutServiceTest

{

	@isTest

	private static void testWareHouseCalloutService()

	{

		test.startTest();

		Test.setMock(HTTPCalloutMock.class, new WarehouseCalloutServiceMock());

		WarehouseCalloutService.syncWarehouseEquipments();

		System.enqueueJob(new WarehouseCalloutService());

		test.stopTest();

		System.assertEquals(1, [SELECT count() FROM Product2]);

	}

}


@IsTest

global class WarehouseCalloutServiceMock implements HttpCalloutMock

{
	global static HttpResponse respond(HttpRequest request)

	{

		HttpResponse response = new HttpResponse();

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

		response.setBody(

		'[{"_id":"55d66226726b611100aaf741","replacement":false,"quantity":5,"name":"Generator
		1000kW","maintenanceperiod":365,"lifespan":120,"cost":5000,"sku":"100003"}]');

		response.setStatusCode(200);

		return response;

	}

}
```

6. Test Scheduling Logic

```
@isTest

public with sharing class WarehouseSyncScheduleTest

{

        @isTest

        public static void testWarehouseSyncSchedule()

        {

                String CRON_EXP  = '00 00 00 * * ? *';

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

                Test.startTest();

                System.schedule('WarehouseSyncScheduleTest', CRON_EXP, new WarehouseSyncSchedule());

                Test.stopTest();

                System.assertEquals(1, [SELECT count() FROM CronTrigger WHERE CronJobDetail.Name = 
'WarehouseSyncScheduleTest']);

        }

}
```