Apex Specialist ->step2 ->Automate record creation

1. Maintenancerequest.apxt

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

   }
}
```

2. MaintenanceRequesthelper.apxc

```
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,
Vehiclec, Equipmentc, Equipmentr.Maintenance_Cyclec,
(SELECT Id,Equipmentc,Quantityc FROM Equipment_Maintenance_Itemsr)
FROM Case WHEREIN :validIds]); Map<Id,Decimal>
```

```
maintenanceCycles = new Map<ID,Decimal>();
//calculate the maintenance request due dates by using the maintenance cycle
defined on the related equipment records.
AggregateResult[] results = [SELECT
Maintenance_Requestc,
MIN(Equipmentr.Maintenance_Cyclec)cycle FROM
Equipment_Maintenance_Itemc
WHERE Maintenance_Requestc IN :ValidIds GROUP BY
Maintenance_Requestc];
 for (AggregateResult ar : results){maintenanceCycles.put((Id)
ar.get('Maintenance_Requestc'), (Decimal) ar.get('cycle'));
                        }

 List<Case> newCases = new List<Case>(); for(Case cc : closedCases.values()){
Case nc = newCase ( ParentId = cc.Id, Status = 'New',
Subject = 'RoutineMaintenance',
Type = 'Routine Maintenance',
Vehiclec = cc.Vehiclec,
Equipmentc =cc.Equipmentc,
Origin = 'Web',
Date_Reportedc = Date.Today()
                        );

//If multiple pieces of equipment are used in the maintenance request,
    //define the due date by applying the shortest maintenance cycle to today's date.

    //If (maintenanceCycles.containskey(cc.Id)){
       nc.Date_Duec =
       Date.today().addDays((Integer)
       maintenanceCycles.get(cc.Id));
                        //}
else {/     nc.Date_Duec = Date.today().addDays((Integer)
cc.Equipmentr.maintenance_Cycle_c);
                        //}
                        newCases.add(nc);
                      }
```

```
            insert newCases;

            List<Equipment_Maintenance_Itemc> clonedList
        = new List<Equipment_Maintenance_Itemc>();
            for (Case nc : newCases){
               for (Equipment_Maintenance_Itemc clonedListItem :
        closedCases.get(nc.ParentId).Equipment_Maintenance_Itemsr){
                  Equipment_Maintenance_Itemc item =
                  clonedListItem.clone();
                  item.Maintenance_Requestc = nc.Id

                  clonedList.add(item);
               }
            }
            insert clonedList;
         }
      }
    }
```

Apex Specialist ->step3-> Synchronize Salesforce data with an external system

1.WarehouseCalloutService.apxc

```
    public with sharing class WarehouseCalloutService {

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

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

        request.setEndpoint(WAREHOUSE_UR
        L); 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.getBo
    dy());
            System.debug(response.getBody());

            for (Object eq : jsonResponse){
               Map<String,Object> mapJson =
               (Map<String,Object>)eq; Product2
               myEq = new Product2();
               myEq.Replacement_Partc = (Boolean)
               mapJson.get('replacement'); myEq.Name =
               (String) mapJson.get('name');
               myEq.Maintenance_Cyclec = (Integer)
               mapJson.get('maintenanceperiod'); myEq.Lifespan_Monthsc =
               (Integer) mapJson.get('lifespan');
               myEq.Costc = (Decimal)
               mapJson.get('lifespan');
               myEq.Warehouse_SKUc = (String)
               mapJson.get('sku');
               myEq.Current_Inventoryc = (Double)
               mapJson.get('quantity');
               warehouseEq.add(myEq);
            }
if(warehouseEq.size() >0){
upsert warehouseEq;
System.debug('Your equipment was synced with the warehouse one');
System.debug(warehouseEq);
```

```
}

}
}
}
```

Apex Specialist ->step4-> Schedule synchronization

1)WarehouseSyncSchedule.apxc

```
global class WarehouseSyncSchedule implements Schedulable {

global void execute(SchedulableContext ctx) {

                    WarehouseCalloutService.runWarehouseEquipmentSync();
                }
            }
```

Apex Specialist-> step5 ->Test automation logic

1)MaintenanceRequesthelperTest.apxc

```
@isTest
public with sharing class MaintenanceRequestHelperTest {

/ createVehicle
private static Vehiclec createVehicle(){
Vehiclec vehicle = new VehicleC(name = 'Testing
Vehicle'); return vehicle;
                    }

/ createEquipment
private static Product2 createEquipment(){
product2 equipment = new product2(name = 'Testing equipment',
lifespan_monthsc = 10,
```

```
maintenance_cyclec = 10,
replacement_partc = 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',
Equipmentc=equipmentId,
Vehiclec=vehicleId);
                    return cse;
                }

/ createEquipmentMaintenanceItem

private static Equipment_Maintenance_Itemc
createEquipmentMaintenanceItem(id equipmentId,id requestId){
Equipment_Maintenance_Itemc equipmentMaintenanceItem = new
Equipment_Maintenance_Itemc(
Equipmentc = equipmentId,
Maintenance_Requestc = requestId);
return equipmentMaintenanceItem;
                }
@isTest
private static void
testPositive(){ Vehiclec
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_Itemc equipmentMaintenanceItem =
createEquipmentMaintenanceItem(equipmentId,createdCase.id);
 insert equipmentMaintenanceItem;
test.startTest();
createdCase.status = 'Closed';
update createdCase;
 test.stopTest();
Case newCase = [Select id,subject,type,
Equipment_c,Date_Reported_c,Vehicle_c,Date_Due_c from case
where status ='New'];

Equipment_Maintenance_Itemc workPart = [select id
from Equipment_Maintenance_Itemc
where Maintenance_Requestc =:newCase.Id]; list<case> allCase =
[select id from case];
system.assert(allCase.size() == 2);
 system.assert(newCase != null);
 system.assert(newCase.Subject != null); system.assertEquals(newCase.Type, 'Routine
Maintenance'); SYSTEM.assertEquals(newCase.Equipmentc, equipmentId);
SYSTEM.assertEquals(newCase.Vehiclec, vehicleId);
SYSTEM.assertEquals(newCase.Date_Reportedc, system.today());
                        }


@isTest
private static void testNegative(){ VehicleC 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_Itemc workP =
createEquipmentMaintenanceItem(equipmentId, createdCase.Id);
insert workP;
test.startTest();
 createdCase.Status = 'Working'; update createdCase;
test.stopTest();
 list<case> allCase = [select id from case];
Equipment_Maintenance_Itemc equipmentMaintenanceItem = [select id
from Equipment_Maintenance_Itemc
where Maintenance_Requestc = :createdCase.Id];
 system.assert(equipmentMaintenanceItem != null);
 system.assert(allCase.size() == 1);
                        }
@isTest
private static void testBulk(){
list<VehicleC> vehicleList = new
list<VehicleC>(); list<Product2> equipmentList =
new list<Product2>();
list<Equipment_Maintenance_Itemc> equipmentMaintenanceItemList = new
list<Equipment_Maintenance_Itemc>();
list<case> caseList = new
list<case>();
list<id> oldCaseIds = new
list<id>();
for(integer i = 0; i < 300; i++)
{ vehicleList.add(createVehicle());
equipmentList.add(createEquipment());
}
insert vehicleList;
 insert equipmentList;
 for(integer i = 0; i < 300; i++){
caseList.add(createMaintenanceRequest(vehicleList.get(i).id,
                    equipmentList.get(i).id));
                         }
insert caseList;
 for(integer i = 0; i < 300; i++){
```

```
 equipmentMaintenanceItemList.add(createEquipmentMaintenanceItem(equipmentList.
get(i).id, caseList.get(i).id));
                        }
                        insert equipmentMaintenanceItemList;
 test.startTest();
 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_Itemc> workParts = [select id
from Equipment_Maintenance_Itemc where Maintenance_Requestc in: oldCaseIds];
 system.assert(newCase.size() == 300);

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

}
                }
```

Apex Specialist ->step6-> Test callout logic

1)WarehouseCalloutServiceMock.apxc

```
 @isTest
global class WarehouseCalloutServiceMock implements HttpCalloutMock {
/ implement http mock callout global static HttpResponse respond(HttpRequest
request){
 System.assertEquals('https://th-superbadge-apex.herokuapp.com/equipment',
request.getEndpoint());
System.assertEquals('GET', request.getMethod());
/ Create a fake response HttpResponse response = new HttpResponse();
response.setHeader('Content-Type', 'application/json');
response.setBody('[{"_id":"55d66226726b611100aaf741","replacement":false,"quantity":5
```

,"name":"Generator 1000
kW","maintenanceperiod":365,"lifespan":120,"cost":5000,"sku":"100003"}]
');
response.setStatusCode(
200); return response;
}
}

1.WarehouseCalloutServiceTest.apxc

@isTestprivate
class WarehouseCalloutServiceTest {
@isTest
static void
testWareHouseCallout(){
Test.startTest();
/ implement mock callout test here
Test.setMock(HTTPCalloutMock.class, new WarehouseCalloutServiceMock());
WarehouseCalloutService.runWarehouseEquipmentSync();
Test.stopTest();
System.assertEquals(1, [SELECT count() FROM Product2]);
}
}

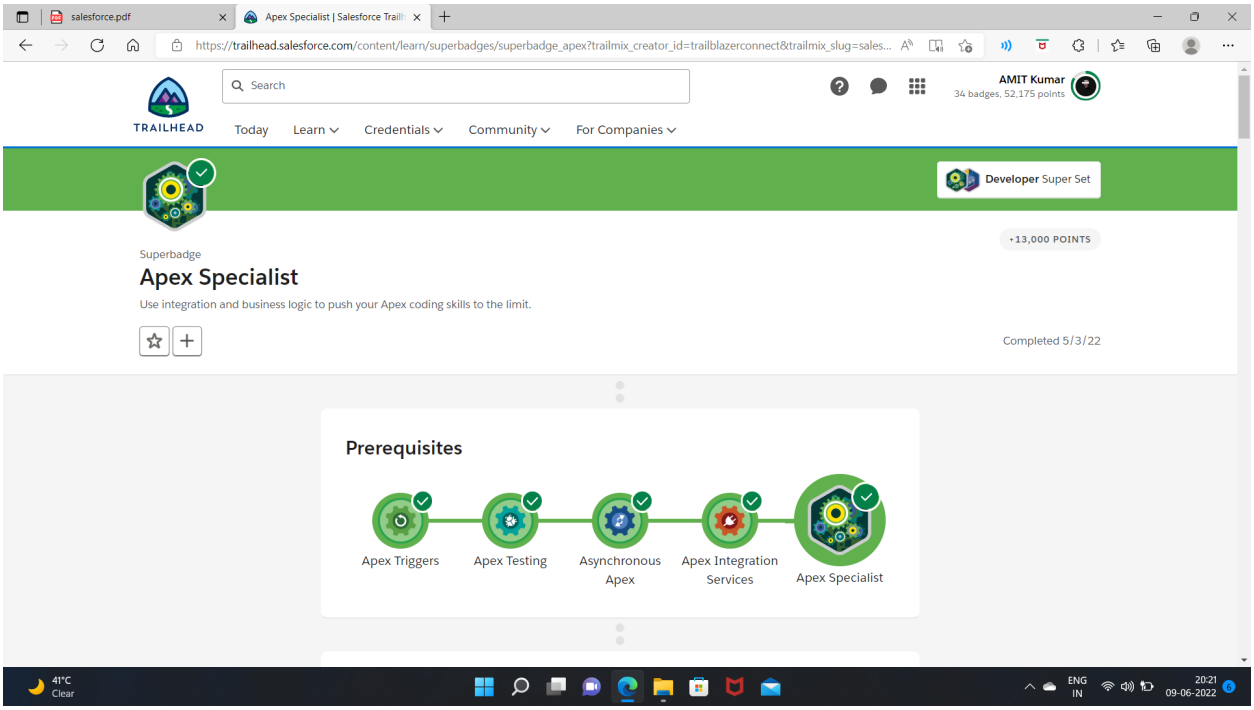
Apex Specialist-> step7-> test scheduling logic

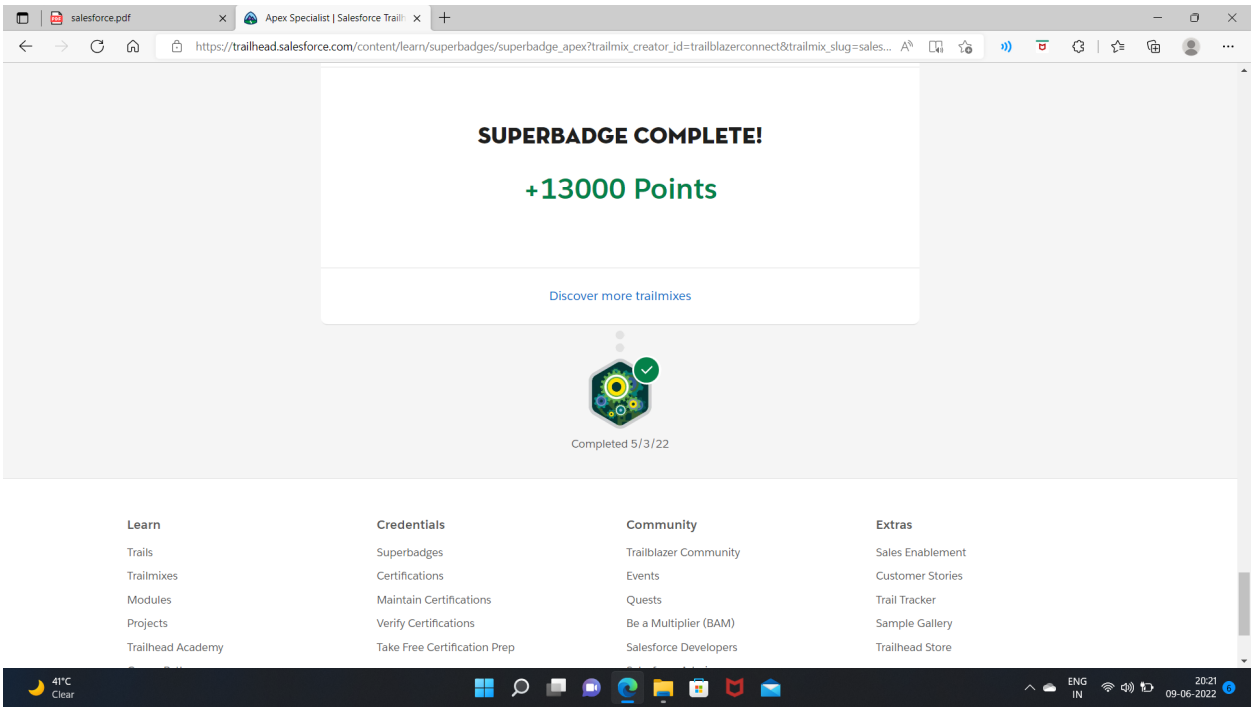
1)WarehouseSyncScheduleTest.apxc

@isTest
public class WarehouseSyncScheduleTest {
@isTest
static void WarehousescheduleTest(){
String scheduleTime = '00 00 01 * * ?';
Test.startTest();
Test.setMock(HttpCalloutMock.class
new
WarehouseCalloutServiceMock());

```
StringjobID=System.schedule('Wareho
use Time To Schedule to Test',
scheduleTime, new
WarehouseSyncSchedule());
Test.stopTest();
CronTrigger a=[SELECT Id FROM CronTrigger where NextFireTime > today];
System.assertEquals(jobID, a.Id,'Schedule ');


}
}
```




salesforce.pdf
Apex Specialist | Salesforce Trailh
https://trailhead.salesforce.com/content/learn/superbadges/superbadge_apex?trailmix_creator_id=trailblazerconnect&trailmix_slug=sales...
SUPERBADGE COMPLETE!
+13000 Points
Discover more trailmixes
Completed 5/3/22
Learn
Trails
Trailmixes
Modules
Projects
Trailhead Academy
Credentials
Superbadges
Certifications
Maintain Certifications
Verify Certifications
Take Free Certification Prep
Community
Trailblazer Community
Events
Quests
Be a Multiplier (BAM)
Salesforce Developers
Extras
Sales Enablement
Customer Stories
Trail Tracker
Sample Gallery
Trailhead Store
41°C
Clear
ENG
IN
20:21
09-06-2022