Apex Specialist

1. Install the unmanaged package for the schema and stubs for Apex classes and triggers. Rename cases and products to match the HowWeRoll schema, and assign all profiles to the custom HowWeRoll page layouts for those objects. Use the included package content to automatically create a Routine Maintenance request every time a maintenance request of type Repair or Routine Maintenance is updated to Closed. Follow the specifications and naming conventions outlined in the business requirements.
    a. Install the unmanaged Package
    b. Rename following objects
        i. Case to Maintenance Request
        ii. Product to Equipment
    c. Set "HowWeRoll" Page Layout as default layout
    d. Create/update "MaintenanceRequest" trigger on Case object. Follow the following guidance
        i. Trigger should run for all cases where type = "Repair" & "Routine Maintenance" & has been marked closed
        ii. Create "MaintenanceRequestHelper" class and "updateWorkOrders" method in the class
        iii. For every case that satisfy condition in point i, Create another case with following Mapping
            1. Subject - Any Subject
            2. Type - 'Routine Maintenance'
            3. Status - 'New'
            4. Date_Due__c - current date + minimum maintenance cycle day
            5. the rest fields stay the same as previous closed case
        iv. To calculate Date_Due__c you will have to Find Maintenance_Cycle__c of the product on that case. Query all the the Products associated with all Cases and Create a Map between product id and Maintenance_Cycle__c
        v. Insert all the newly created Cases.

2. Synchronize Salesforce data with an external system
    a. Setup Remote Site Settings with endpoint: https://th-superbadge-apex.herokuapp.com.
    b. create/update "WarehouseCalloutService" class
        i. Create following inner class,
        //Wrapper class for Response details.
        //You can create this class using JSOn2Apex utility available online, by parsing the JSON provide in requirements.

```
private class Json2Apex{
    private String id;
    private Integer cost;
    private Integer lifespan;
    private Integer maintenanceperiod;
    private String name;
    private Integer quantity;
    private boolean replacement;
    private String sku;
}
```

        ii. Create a method called "runWarehouseEquipmentSync" in the parent class with @future(callout=true) annotation. This annotation will allow callouts(rest calls) in future Methods.
        iii. Create an Http Request
            1. Set endpoint to "'https://th-superbadge-apex.herokuapp.com/equipment'"
            2. Method = "Get"
        iv. Send Http request
        v. Deserialize the response using inner Class, This is give you a list of inner class instances.
        vi. Use the List to create Product records, with following mapping:
            1. Cost__c = cost;
            2. Lifespan_Months__c = lifespan;
            3. Maintenance_Cycle__c = maintenanceperiod;
            4. Name = name;
            5. Current_Inventory__c = quantity;
            6. Replacement_Part__c = replacement;
            7. Warehouse_SKU__c = sku;
        vii. Upsert the product list use Warehouse_SKU__c as external key
        use following syntax :

UPSERT ListName Warehouse_SKU__c;

3. Schedule synchronization
    a. Create a schedulable class called "WarehouseSyncSchedule" for scheduling WarehouseCalloutService
    b. Schedule the schedulable class
4. Test automation logic
    a. Create Test Class for MaintenanceRequest trigger, you need to have 100% coverage and test class should be well bulkified.
        i. Create Test Setup method (@testsetup)
        ii. Insert Some product records in TestSetup
        iii. Create testMaintenanceRequest test method (@isTest)
            1. Get Id of the products inserted
            2. Create cases for all products with following fields Type = 'Repair', Status = 'New', Origin = 'Phone' and use product id in Equipment__c. Create at least 300 cases.
            3. Start Test.StartTest()
            4. Insert Cases
            5. Iterate over all the cases and set Status = closed.
            6. Update Status.
            7. Re-query all the Cases, 300 additional Status should be created.
            8. Assert to ensure there are 600 Cases in the org with 300 cases of type "Routine Maintenance".
5. Test callout logic
    a. Create "WarehouseCalloutServiceMock" which implements "HttpCalloutMock"
        i. Implement public HTTPResponse respond(HTTPRequest request){} method in mocking class

String responseBody = '[{"_id":"55d66226726b611100aaf741","replacement":false,"quantity":5,"name":"Generator 1000 kW","maintenanceperiod":365,"lifespan":120,"cost":5000,"sku":"100003"},{"_id":"55d66226726b611100aaf742","replacement":true,"quantity":183,"name":"Cooling Fan","maintenanceperiod":0,"lifespan":0,"cost":300,"sku":"100004"},{"_id":"55d66226726b611100aaf743","replacement":true,"quantity":143,"name":"Fuse 20A","maintenanceperiod":0,"lifespan":0,"cost":22,"sku":"100005"},{


```
"_id":"55d66226726b611100aaf744","replacement":false,"quantity":
5,"name":"Generator 2000
kw","maintenanceperiod":365,"lifespan":120,"cost":6000,"sku":"100
006"},{"_id":"55d66226726b611100aaf745","replacement":true,"qu
antity":142,"name":"Fuse
25A","maintenanceperiod":0,"lifespan":0,"cost":28,"sku":"100007"},{
"_id":"55d66226726b611100aaf746","replacement":true,"quantity":
122,"name":"Fuse
13A","maintenanceperiod":0,"lifespan":0,"cost":10,"sku":"100008"},{
"_id":"55d66226726b611100aaf747","replacement":true,"quantity":
90,"name":"Ball Valve 10
cm","maintenanceperiod":0,"lifespan":0,"cost":50,"sku":"100009"},{"
_id":"55d66226726b611100aaf748","replacement":false,"quantity":
2,"name":"Converter","maintenanceperiod":180,"lifespan":120,"cost
":3000,"sku":"100010"},{"_id":"55d66226726b611100aaf749","repl
acement":true,"quantity":75,"name":"Ball Valve 8
cm","maintenanceperiod":0,"lifespan":0,"cost":42,"sku":"100011"},{"
_id":"55d66226726b611100aaf74a","replacement":true,"quantity":1
00,"name":"Breaker
25A","maintenanceperiod":0,"lifespan":0,"cost":30,"sku":"100012"}]'
;
HttpResponse response = new HttpResponse();
response.setHeader('Content-Type', 'application/json');
response.setBody(responseBody);
response.setStatusCode(200);

return response;
```

b. Create WarehouseCalloutServiceTest test class for testing WarehouseCalloutService
    i. Write test method testWarehouseCalloutService()
        1. Start Test
        2. Set MockTest Response (Test.SetMock())
        3. Run the method from WarehouseCalloutService class
        4. Stop Test
        5. Assert to ensure ten Products exist.

6. Test scheduling logic
    a. Create WarehouseSyncScheduleTest test class to test WarehouseSyncSchedule
    b. Start Test
    c. Set mock

d. Create scheduling cron expression variable (String sch = '0 0 0 * * ?';)
e. Use System.schedule to schedule WarehouseSyncSchedule class , ensure you save the job id in a variable.
f. Use the job id to query cron trigger
g. Assert to ensure CronExpression of the cron trigger is same as the cron schedule set while scheduling.
h. Stop Test

//CreateDefaultData.cls

```
public with sharing class CreateDefaultData{
   Static Final String TYPE_ROUTINE_MAINTENANCE = 'Routine Maintenance';
   //gets value from custom metadata How_We_Roll_Settings__mdt to know if Default data was
created
   @AuraEnabled
   public static Boolean isDataCreated() {
      How_We_Roll_Settings__c          customSetting =
How_We_Roll_Settings__c.getOrgDefaults();
      return customSetting.Is_Data_Created__c;
   }
   //creates Default Data for How We Roll application
   @AuraEnabled
   public static void createDefaultData(){
      List<Vehicle__c> vehicles = createVehicles();
      List<Product2> equipment = createEquipment();
      List<Case> maintenanceRequest = createMaintenanceRequest(vehicles);
      List<Equipment_Maintenance_Item__c> joinRecords = createJoinRecords(equipment,
maintenanceRequest);

      updateCustomSetting(true);
   }



   public static void updateCustomSetting(Boolean isDataCreated){
      How_We_Roll_Settings__c          customSetting =
How_We_Roll_Settings__c.getOrgDefaults();
      customSetting.Is_Data_Created__c = isDataCreated;
      upsert customSetting;
   }

   public static List<Vehicle__c> createVehicles(){
      List<Vehicle__c> vehicles = new List<Vehicle__c>();
      vehicles.add(new Vehicle__c(Name = 'Toy Hauler RV', Air_Conditioner__c = true,
Bathrooms__c = 1, Bedrooms__c = 1, Model__c = 'Toy Hauler RV'));
      vehicles.add(new Vehicle__c(Name = 'Travel Trailer RV', Air_Conditioner__c = true,
Bathrooms__c = 2, Bedrooms__c = 2, Model__c = 'Travel Trailer RV'));
      vehicles.add(new Vehicle__c(Name = 'Teardrop Camper', Air_Conditioner__c = true,
Bathrooms__c = 1, Bedrooms__c = 1, Model__c = 'Teardrop Camper'));
      vehicles.add(new Vehicle__c(Name = 'Pop-Up Camper', Air_Conditioner__c = true,
```

```
Bathrooms__c = 1, Bedrooms__c = 1, Model__c = 'Pop-Up Camper'));
        insert vehicles;
        return vehicles;
    }

    public static List<Product2> createEquipment(){
        List<Product2> equipments = new List<Product2>();
        equipments.add(new Product2(Warehouse_SKU__c = '55d66226726b611100aaf741',name
= 'Generator 1000 kW', Replacement_Part__c = true,Cost__c = 100 ,Maintenance_Cycle__c =
100));
        equipments.add(new Product2(name = 'Fuse 20B',Replacement_Part__c = true,Cost__c =
1000, Maintenance_Cycle__c = 30  ));
        equipments.add(new Product2(name = 'Breaker 13C',Replacement_Part__c = true,Cost__c =
100  , Maintenance_Cycle__c = 15));
        equipments.add(new Product2(name = 'UPS 20 VA',Replacement_Part__c = true,Cost__c =
200  , Maintenance_Cycle__c = 60));
        insert equipments;
        return equipments;
    }

    public static List<Case> createMaintenanceRequest(List<Vehicle__c> vehicles){
        List<Case> maintenanceRequests = new List<Case>();
        maintenanceRequests.add(new Case(Vehicle__c = vehicles.get(1).Id, Type =
TYPE_ROUTINE_MAINTENANCE, Date_Reported__c = Date.today()));
        maintenanceRequests.add(new Case(Vehicle__c = vehicles.get(2).Id, Type =
TYPE_ROUTINE_MAINTENANCE, Date_Reported__c = Date.today()));
        insert maintenanceRequests;
        return maintenanceRequests;
    }

    public static List<Equipment_Maintenance_Item__c> createJoinRecords(List<Product2>
equipment, List<Case> maintenanceRequest){
        List<Equipment_Maintenance_Item__c> joinRecords = new
List<Equipment_Maintenance_Item__c>();
        joinRecords.add(new Equipment_Maintenance_Item__c(Equipment__c =
equipment.get(0).Id, Maintenance_Request__c = maintenanceRequest.get(0).Id));
        joinRecords.add(new Equipment_Maintenance_Item__c(Equipment__c =
equipment.get(1).Id, Maintenance_Request__c = maintenanceRequest.get(0).Id));
        joinRecords.add(new Equipment_Maintenance_Item__c(Equipment__c =
equipment.get(2).Id, Maintenance_Request__c = maintenanceRequest.get(0).Id));
```

```
        joinRecords.add(new Equipment_Maintenance_Item__c(Equipment__c =
equipment.get(0).Id, Maintenance_Request__c = maintenanceRequest.get(1).Id));
        joinRecords.add(new Equipment_Maintenance_Item__c(Equipment__c =
equipment.get(1).Id, Maintenance_Request__c = maintenanceRequest.get(1).Id));
        joinRecords.add(new Equipment_Maintenance_Item__c(Equipment__c =
equipment.get(2).Id, Maintenance_Request__c = maintenanceRequest.get(1).Id));
        insert joinRecords;
        return joinRecords;

    }
}
```

///CreateDefaultDataTest.cls

```
@isTest
private class CreateDefaultDataTest {
    @isTest
    static void createData_test(){
        Test.startTest();
        CreateDefaultData.createDefaultData();
        List<Vehicle__c> vehicles = [SELECT Id FROM Vehicle__c];
        List<Product2> equipment = [SELECT Id FROM Product2];
        List<Case> maintenanceRequest = [SELECT Id FROM Case];
        List<Equipment_Maintenance_Item__c> joinRecords = [SELECT Id FROM
Equipment_Maintenance_Item__c];

        System.assertEquals(4, vehicles.size(), 'There should have been 4 vehicles created');
        System.assertEquals(4, equipment.size(), 'There should have been 4 equipment created');
        System.assertEquals(2, maintenanceRequest.size(), 'There should have been 2
maintenance request created');
        System.assertEquals(6, joinRecords.size(), 'There should have been 6 equipment
maintenance items created');

    }

    @isTest
    static void updateCustomSetting_test(){
```

```
        How_We_Roll_Settings__c          customSetting =
How_We_Roll_Settings__c.getOrgDefaults();
        customSetting.Is_Data_Created__c = false;
        upsert customSetting;

        System.assertEquals(false, CreateDefaultData.isDataCreated(), 'The custom setting
How_We_Roll_Settings__c.Is_Data_Created__c should be false');

        customSetting.Is_Data_Created__c = true;
        upsert customSetting;
        System.assertEquals(true, CreateDefaultData.isDataCreated(), 'The custom setting
How_We_Roll_Settings__c.Is_Data_Created__c should be true');

    }
}



////MaintenanceRequestHelper.cls
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);
                }
            }
        }
        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()
                );
                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;
        }
    }
}

/// MaintenanceRequestHelperTest .cls
@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 Maintenance';
    private static final string REQUEST_SUBJECT = 'Testing subject';
    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];
        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);
    }
```

```
}



//WarehouseCalloutService.cls
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_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());
         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 = (Decimal) mapJson.get('lifespan');
            myEq.Warehouse_SKU__c = (String) mapJson.get('sku');
            myEq.Current_Inventory__c = (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);
         }
      }
   }
```

```
}



// WarehouseCalloutServiceMock.cls
@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;
    }
}

//WarehouseCalloutServiceTest.cls

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

```
//
global class WarehouseSyncSchedule implements Schedulable {
   global void execute(SchedulableContext ctx) {
      WarehouseCalloutService.runWarehouseEquipmentSync();
   }
}

//WarehouseSyncScheduleTest.cls
@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 Test', scheduleTime, new
WarehouseSyncSchedule());
      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.assertEquals(jobID, a.Id,'Schedule ');
   }
}
```