APEX TRIGGERS

1. Get Started with Apex Triggers

Name : AccountAddressTrigger

```
trigger AccountAddressTrigger on Account (beforeinsert,before update)

   {for(Account account:Trigger.New){

      if(account.Match_Billing_Address_c==True){
         account.ShippingPostalCode=account.BillingPostalCode;
      }


   }

}
```

2. Bulk Apex Triggers

Name : ClosedOpportunityTrigger

```
trigger ClosedOpportunityTrigger on Opportunity (after insert, after update)

   {List<Task> taskList= new List<Task>();

   for(Opportunity opp : Trigger.new) {

                  //Only create FollowUp Task only once when Opp StageName is to
'ClosedWon' on Create
```

```
                if(Trigger.isInsert) {
                        if(Opp.StageName == 'Closed Won'){
                                taskList.add(new Task(Subject = 'Follow Up Test Task',
WhatId = opp.Id));
                        }

                }

                //Only create FollowUp Task only once when Opp StageNamechanged to
'ClosedWon' on Update
                if(Trigger.isUpdate) {
                        if(Opp.StageName == 'Closed Won'
                        && Opp.StageName !=
                                Trigger.oldMap.get(opp.Id).StageName) {
                                taskList.add(new Task(Subject = 'FollowUp Test Task',
WhatId = opp.Id));
                        }
                }
    }

    if(taskList.size()>
        0) {insert
        taskList;
    }
}
```

APEX TESTING

1. Get StartedWith Apex Unit

TestsName : VerifyDate

```
public class VerifyDate {

        //method to handle potential checks against two
        datespublicstatic Date CheckDates(Date date1, Date
        date2) {
                //if date2 is within the next 30 days of date1, use date2.  Otherwise
  usethe end of the month
                if(DateWithin30Days(date1,date2
                        )) {return date2;

                } else {

                }
                                                }

return  SetEndOfMonthDate(date1);


        //method to check if date2 is within the next 30 days of date1
        @TestVisible privatestatic Boolean DateWithin30Days(Date date1,Date date2) {
                //check for date2 being in the
        pastif( date2 < date1) { returnfalse; }

        //check that date2 is within (>=)30 days of date1
        Date date30Days = date1.addDays(30); //createa date 30 days away from
                date1if( date2 >= date30Days ) { return false; }
                else { return true; }
        }
```

```
        //method to return the end of the month of a given date
        @TestVisible privatestatic Date SetEndOfMonthDate(Date
        date1){
                Integer totalDays = Date.daysInMonth(date1.year(), date1.month());
                Date lastDay = Date.newInstance(date1.year(), date1.month(),
                totalDays);return lastDay;
        }

}
```

Name : TestVerifyDate

```
@isTest
private class TestVerifyDate {
   @isTest static void
      Test_CheckDates_case1(){Date D
=VerifyDate.CheckDates(date.parse('01/01/2020'),date.parse('01/05/2020'));
      System.assertEquals(date.parse('01/05/2020'),D);
   }
   @isTest staticvoid Test_CheckDates_case2(){

      Date D
=VerifyDate.CheckDates(date.parse('01/01/2020'),date.parse('05/05/2020'));
      System.assertEquals(date.parse('01/31/2020'),D);
   }
   @isTest static void Test_DateWithin30Days_case1(){
      Boolean
flag=VerifyDate.DateWithin30Days(date.parse('01/01/2020'),date.parse('12/30/2019'));
      System.assertEquals(false,flag);
   }
   @isTest static void
      Test_DateWithin30Days_case2(){Boolean flag
```

```
=VerifyDate.DateWithin30Days(date.parse('01/01/2020'),date.parse('2/02/2019'));
        System.assertEquals(false,flag);
    }
    @isTest static void Test_DateWithin30Days_case3(){
        Boolean
flag=VerifyDate.DateWithin30Days(date.parse('01/15/2020'),date.parse('01/15/2019'));
        System.assertEquals(true,flag);
    }
    @isTest static void Test_SetEndOfMonthDate(){
        Date     returndate=VerifyDate.SetEndOfMonthDate(date.parse('01/01/2020'));


    }
}
```

2. Test Apex Triggers

Name : RestrictContactByName

```
trigger RestrictContactByName on Contact (beforeinsert, before update){

        //check contacts prior to insert or update for
        invaliddataFor (Contactc : Trigger.New) {



DML');

if(c.LastName == 'INVALIDNAME') {       //invalidname is invalid c.AddError('The
        Last Name "'+c.LastName+'" is not allowedfor
```

```
            }

        }


}
```

Name : TestRestrictContactByName

```
@isTest
public class TestRestrictContactByName {
   @isTest static void
   Test_insertupdateContact(){
      Contact cnt = new Contact();
      cnt.LastName = 'INVALIDNAME';
      Test.startTest();
      Database.SaveResult  result=  Database.insert(cnt,
      false);Test.stopTest();
      System.assert(!result.isSuccess());Syste
      m.assert(result.getErrors().size() > 0);
      System.assertEquals('The Last Name "INVALIDNAME" is not allowedfor
DML',result.getErrors()[0].getMessage());
}
}
```

3. Create Test Data for apex Tests

Name : RandomContactFactory

```
publicclass RandomContactFactory {
   public static List<Contact> generateRandomContacts(Integer num,String
```

```
        lastName){List<Contact> contactList=new List<Contact>();
    for(Integer i=1;i<=num;i++){
      Contact ct=new Contact(FirstName='Test'+i,LastName=lastName);

      contactList.add(ct);
    }
    return contactList;
  }
}
```

Asynchronous Apex

1. Use Future Methods

Name :

```
AccountProcessorpublic class
AccountProcessor {
  @future
  public static void countContacts(List<Id> accountIds){
    List<Account> accountsToUpdate = new List<Account>();
    List<Account> accounts = [Select Id, Name, (SelectId from Contacts)from Account Where
Id in :accountIds];
    For(Account acc:accounts){
      List<Contact> contactList = acc.Contacts;
      acc.Number_Of_Contacts_c = contactList.size();
      accountsToUpdate.add(acc);
    }
    update accountsToUpdate;

  }

}
```

Name : AccountProcessorTest

```
@isTest
private class AccountProcessorTest {
   @IsTest
   private static void testCountContacts(){
      Account newAccount = new Account(Name='Test Account');
      insert newAccount;

      Contact newContact1 = new Contact(FirstName='John',LastName='Doe',AccountId =
newAccount.Id);
      insert newContact1;

      Contact newContact2 = new Contact(FirstName='John',LastName='Doe',AccountId =
newAccount.Id);
      insert newContact2;

      List<Id> accountIds = new List<Id>();
      accountIds.add(newAccount.Id);
      Test.startTest();
      AccountProcessor.countContacts(accountIds
      );Test.stopTest();
   }

}
```

2. Use Batch Apex

Name : LeadProcessor

```
global class LeadProcessor implements Database.Batchable<sObject> {
   global Integer count=0;

   global Database.QueryLocator start(Database.BatchableContext bc){
      return Database.getQueryLocator('SELECT ID, LeadSource FROM Lead');
```

```
    }
    global void execute(Database.BatchableContext bc,List<Lead> L_list){
        List<lead> L_list_new=new List<lead>();
        for(lead L:L_list){
            L.leadsource='Dreamforc
            e'; L_list_new.add(L);
            count+=1;
        }
        update L_list_new;

        }
        global void finish(Database.BatchableContext bc){
            system.debug('count = '+count);
    }



}
```

Name : LeadProcessorTest

```
@isTest
public class LeadProcessorTest {

    @isTest
    public staticvoid testit(){
        List<lead> L_list = new List<lead>();

        for(Integer i=0; i<200; i++){
            Lead L = new lead();
            L.LastName = 'name' + i;
            L.Company = 'Company';
            L.Status = 'Random
            Status';L_list.add(L);
        }
        insert L_list;
        Test.startTest();
        LeadProcessor lp = new
        LeadProcessor();Id batchId =
```

```
        Database.executeBatch(lp);
        Test.stopTest();
    }

}
```

3. Contol Processor With Queueable Apex

Name : AddPrimaryContact

```
public class AddPrimaryContact implements Queueable{

    privateContact con;
    privateString state;

    public AddPrimaryContact(Contact con, String state){
        this.con= con;
        this.state = state;
    }


    public void execute(QueueableContext context){
        List<Account> accounts = [Select Id, Name, (SelectFirstName, LastName, Id from contacts)
                                  from Account whereBillingState = :stateLimit 200];
        List<Contact>primaryContacts = new List<Contact>();

        for(Account
            acc:accounts){contact
            c = con.clone();
            c.AccountId = acc.Id;
            primaryContacts.add(c
            );
        }
        if(primaryContacts.size() >
            0){insert
            primaryContacts;
        }
    }
```

```
}
```

Name : AddPrimaryContactTest

```
@isTest
public class AddPrimaryContactTest {

   static testmethod void testQueueable(){
      List<Account> testAccounts = new
      List<Account>();for(Integer i=0;i<50;i++){
         testAccounts.add(new  Account(Name='Account  '+i,BillingState='CA'));
      }
      for(Integer j=0;j<50;j++){
         testAccounts.add(new  Account(Name='Account  '+j,BillingState='NY'));
      }
      insert testAccounts;

      Contact testContact = new Contact(FirstName = 'John', LastName ='Doe');
      insert testContact;
      AddPrimaryContact addit = new AddPrimaryContact(testContact, 'CA');

      Test.startTest();

      system.enqueueJob(addit);
      Test.stopTest();

      system.assertEquals(50,[Select count() from Contact whereaccountId in (SelectID from
Accountwhere BillingState='CA')]);
   }
}
```

4)Schedle Jobs Using the Apex scheduler

Name :

DailyLeadProcessorglobal class DailyLeadProcessor

implements Schedulable{

```
   global void execute(SchedulableContext ctx){
      List<lead> leadstoupdate = new List<lead>();
      List<Lead> leads= [Select id from Lead Where LeadSource = NULL Limit 200];

      for(Lead   l:leads){
         l.LeadSource  =
         'Dreamforce';
         leadstoupdate.add(l);
   }
   update leadstoupdate;
   }
```

}

Name : DailyLeadProcessorTest

```
@isTest
private class DailyLeadProcessorTest {
        static testMethod void testDailyLeadProcessor() {
                StringCRON_EXP = '0 0 1 * * ?';
                List<Lead> lList = new List<Lead>();
           for (Integeri = 0; i < 200; i++) {
                        lList.add(new Lead(LastName='Dreamforce'+i,Company='Test1 Inc.',
Status='Open - Not Contacted'));
                }
                insert lList;

                Test.startTest();
                String jobId = System.schedule('DailyLeadProcessor', CRON_EXP,new
DailyLeadProcessor());
        }
```

}

APEX INTEGRATION OVERVIEW

1. Apex REST Callouts

Name : AnimalLocator

```
public class AnimalLocator{
   public static String getAnimalNameById(Integer
      x){Http http = new Http();
      HttpRequest req = new HttpRequest();
      req.setEndpoint('https://th-apex-http-callout.herokuapp.com/animals/' + x);
      req.setMethod('GET');
      Map<String, Object> animal= new Map<String,
      Object>();HttpResponse res = http.send(req);
         if (res.getStatusCode() == 200) {
      Map<String, Object> results =
      (Map<String,
Object>)JSON.deserializeUntyped(res.getBody());
     animal = (Map<String, Object>)
     results.get('animal');
      }
return (String)animal.get('name');
}
}
```

Name : AnimalLocatorTest

```
@isTest
private class AnimalLocatorTest{
   @isTest static void AnimalLocatorMock1() {
      Test.setMock(HttpCalloutMock.class, new
      AnimalLocatorMock());string result=
```

```
        AnimalLocator.getAnimalNameById(3);
        String expectedResult =
        'chicken';System.assertEquals(result,expectedResult );
    }
}
```

Name : AnimalLocatorMock

```
@isTest
global class AnimalLocatorMock implements HttpCalloutMock {
    / Implementthis interface method
    global HTTPResponse respond(HTTPRequest request) {
        / Create a fake response
        HttpResponse response = new HttpResponse();
        response.setHeader('Content-Type', 'application/json');
        response.setBody('{"animals": ["majestic badger", "fluffy bunny", "scary
        bear",
"chicken", "mighty moose"]}');
        response.setStatusCode(200);
        return response;
    }
}
```

2. Apex SOAP Callouts

Name :

```
ParkLocatorpublic classParkLocator

{

    public staticstring[] country(string theCountry) {
        ParkService.ParksImplPort  parkSvc = new  ParkService.ParksImplPort(); /
removespace
        return  parkSvc.byCountry(theCountry);
```

```
    }
}
```

Name : ParkLocatorTest

```
@isTest
private class ParkLocatorTest {
    @isTest static void testCallout() {
        Test.setMock(WebServiceMock.class, new ParkServiceMock
        ());String country= 'United States';
        List<String> result = ParkLocator.country(country);
        List<String>parks = new List<String>{'Yellowstone', 'Mackinac National Park',

'Yosemite'};
         System.assertEquals(parks, result);
    }
}
```

Name : ParkServiceMock

```
@isTest
global class ParkServiceMock implements
  WebServiceMock {global void doInvoke(
          Object stub,
          Object request,
          Map<String, Object> response,
          String endpoint,
          String soapAction,
          String
          requestName,
          String
          responseNS,
```

```
String responseName,
        String responseType) {
      / start - specify the response you want to send
      ParkService.byCountryResponse response_x = new
ParkService.byCountryResponse();
      response_x.return_x = new List<String>{'Yellowstone', 'Mackinac National Park',
'Yosemite'};
      / end
      response.put('response_x', response_x);
  }
}
```

Name : AsyncParkService

```
//Generated by wsdl2apex

publicclass AsyncParkService

{


   public class byCountryResponseFuture extends System.WebServiceCalloutFuture {
      public String[] getValue() {
         ParkService.byCountryResponse response =
(ParkService.byCountryResponse)System.WebServiceCallout.endInvoke(this);
         return response.return_x;
      }
   }
   public class AsyncParksImplPort {
      public String endpoint_x = 'https://th-apex-soap-
service.herokuapp.com/service/parks';
      publicMap<String,String> inputHttpHeaders_x;
```

```
        public StringclientCertName_x;
        public Integer timeout_x;
        private String[]ns_map_type_info = new String[]{'http://parks.services/',
'ParkService'};
        public AsyncParkService.byCountryResponseFuture
beginByCountry(System.Continuation continuation,String arg0) {
            ParkService.byCountry request_x= new ParkService.byCountry();
            request_x.arg0 = arg0;
            return (AsyncParkService.byCountryResponseFuture)
System.WebServiceCallout.beginInvoke(
              this,
              request_x,
              AsyncParkService.byCountryResponseFuture.class,
              continuation,
              new
              String[]{endpoint_x,'',
              'http://parks.services/',
              'byCountry',
              'http://parks.services/',
              'byCountryResponse',
              'ParkService.byCountryRespon
              se'}
            );
        }
    }
}
```

3. Apex Web Services

Name : AccountManager

```
@RestResource(urlMapping='/Accounts/*/contacts
') global class AccountManager {
    @HttpGet
    global static Account getAccount() {
        RestRequest req = RestContext.request;
```

```
        StringaccId = req.requestURI.substringBetween('Accounts/',
        '/contacts');Account acc = [SELECTId, Name, (SELECTId, Name FROM
        Contacts)
                FROM Account WHERE Id =
        :accId];returnacc;
    }
}
```

Name : AccountManagerTest

```
@isTest
private classAccountManagerTest {

    private static testMethod void
        getAccountTest1() {Id recordId=
        createTestRecord();
        / Set up a test request
        RestRequest request= new RestRequest();
        request.requestUri =
'https://na1.salesforce.com/services/apexrest/Accounts/'+ recordId +'/contacts' ;
        request.httpMethod = 'GET';
        RestContext.request =
        request;
        / Call the method to test
 Account thisAccount = AccountManager.getAccount();
        / Verify
        resultsSystem.assert(thisAccount
        != null);
        System.assertEquals('Test record', thisAccount.Name);


    }

    / Helpermethod
```

```
    static Id createTestRecord() {
    / Createtest record
    Account TestAcc = new
      Account(Name='Test record');
    insert TestAcc;
    Contact TestCon= new
    Contact(LastName='Test',
    AccountId =
    TestAcc.id);return
    TestAcc.Id;
  }
}
```

APEX SPECIALIST

1. Automated Record Creation

Name : MaintenceRequestHelper

```
public with sharing classMaintenanceRequestHelper {
  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__cFROM Equipment_Maintenance_Items__r)
                                        FROM Case WHERE Id IN :validIds]);
        Map<Id,Decimal> maintenanceCycles = new
        Map<ID,Decimal>();AggregateResult[] results =
        [SELECTMaintenance_Request_c,
MIN(Equipment_r.Maintenance_Cycle__c)cycle FROM
Equipment_Maintenance_Item_c WHERE Maintenance_Request_c IN :ValidIdsGROUP
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));
            } else {
                nc.Date_Due_c = Date.today().addDays((Integer)
cc.Equipment_r.maintenance_Cycle_c);
            }


            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;
    }
  }
}
```

Name : MaintananceRequest

```
trigger MaintenanceRequest on Case (beforeupdate, after

   update){if(Trigger.isUpdate && Trigger.isAfter){

      MaintenanceRequestHelper.updateWorkOrders(Trigger.New, Trigger.OldMap);

   }

}
```

2. Synchronize Saleesforce data with an External System

Name : WarehouseCalloutService

```
public with sharing class WarehouseCalloutService implements Queueable {

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

   //class that makes a REST callout to an externalwarehouse system to get a list
ofequipment that needs to be updated.
   //The callout's JSON responsereturns the equipmentrecords that you upsert in
Salesforce.

   @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());

            //class maps the following fields:replacement part (alwaystrue), cost,
currentinventory, lifespan, maintenance cycle, and warehouse SKU
            //warehouse SKU will be externalID for identifying which equipmentrecords to
updatewithin Salesforce
            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 = (Integer) mapJson.get('cost');
                myEq.Warehouse_SKU_c = (String)mapJson.get('sku');

                myEq.Current_Inventory_c = (Double) mapJson.get('quantity');
                myEq.ProductCode = (String) mapJson.get('_id');
                warehouseEq.add(myEq);
            }

            if (warehouseEq.size() >
                0){
                upsertwarehouseEq;
                System.debug('Your equipment was syncedwith the warehouseone');
            }
        }
    }
```

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

}
```

3. Schedule Synchronization UsingApex Code

Name : WarehouseSyncSchedule

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

4. Test automation

Name : MaintenanceRequestHelperTest

```
@istest
public with sharing class MaintenanceRequestHelperTest {

    private static final string STATUS_NEW =
    'New'; private static final string WORKING=
    'Working'; privatestatic final stringCLOSED =
    'Closed';

    private static final string REPAIR = 'Repair';
    private static final stringREQUEST_ORIGIN = 'Web';
    privatestatic final string REQUEST_TYPE = 'Routine
    Maintenance';private static final stringREQUEST_SUBJECT =
```

```
    'Testingsubject';

    PRIVATE STATIC Vehicle_c createVehicle(){
        Vehicle_c Vehicle= new Vehicle_C(name = 'SuperTruck');
        returnVehicle;
    }

    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_cvehicle = createVehicle();
      insert vehicle;
      id vehicleId = vehicle.Id;


      Product2 equipment =
      createEq();insertequipment;
      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_cworkPart = [selectid
                                    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.Equipmentc, equipmentId);
```

```
    SYSTEM.assertEquals(newReq.Vehicle_c,vehicleId);
    SYSTEM.assertEquals(newReq.Date_Reported_c,system.today());

  }

  @istest
  private static void
    testMaintenanceRequestNegative(){Vehicle_
    Cvehicle = createVehicle();
    insert vehicle;
    id vehicleId = vehicle.Id;

    product2 equipment =
    createEq();insertequipment;
    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_cworkPart = [selectid
                        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);
        }
        updaterequestList;
        test.stopTest();

        list<case> allRequests = [select id
                                  from case
                                  where status =: STATUS_NEW];

        list<Equipment_Maintenance_Item_c>workParts = [selectid
                                                       from Equipment_Maintenance_Item_c

                                                       where Maintenance_Request_c in: oldRequestIds];

        system.assert(allRequests.size()  ==  300);
    }
}
```

Name : MaintenanceRequestHelper

```
public with sharing classMaintenanceRequestHelper {
    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__cFROM Equipment_Maintenance_Items__r)
                                        FROM Case WHERE Id IN :validIds]);
        Map<Id,Decimal> maintenanceCycles = new
        Map<ID,Decimal>();AggregateResult[] results =
        [SELECTMaintenance_Request_c,
MIN(Equipment_r.Maintenance_Cycle__c)cycle FROM
Equipment_Maintenance_Item_c WHERE Maintenance_Request_c IN :ValidIdsGROUP
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;
    }
  }
}
```

Name : MaintenanceRequest

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

5. Test Callout Logic

Name :

```
WarehouseCalloutServicepublic with sharing class
WarehouseCalloutService {

   private staticfinal 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 (Objecteq : 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){
            upsertwarehouseEq;
            System.debug('Your equipment was syncedwith the warehouse one');
            System.debug(warehouseEq);
        }


    }
  }
}
Name : WarehouseCalloutServiceTest
@isTest

private 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, [SELECTcount() FROM Product2]);

    }
}
```

Name : WarehouseCalloutServiceMock

```
@isTest
global class WarehouseCalloutServiceMock implements HttpCalloutMock {
    / implementhttp 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;
    }
}
```

6. Test Schedling Logic

Name : WarehouseSyncSchedule

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

    WarehouseCalloutService.runWarehouseEquipmentSync();
  }
}
```

Name : WarehouseSyncScheduleTest

```
@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 Scheduleto Test',
scheduleTime, new
    WarehouseSyncSchedule());
    Test.stopTest();
    //Contains schedule information for a scheduledjob. CronTrigger is similar to
acron job on UNIX systems.
    / This objectis available in API version17.0 and later.
    CronTrigger a=[SELECT Id FROM CronTrigger where NextFireTime >
    today];System.assertEquals(jobID, a.Id,'Schedule ');


  }
}
```