Apex Specialist Superbadge

Challenge 1:

This the first challenge where we attenda quiz answering some generalquestions regarding the superbadge challengethat we are doing.

Challenge 2:

It is all about preparingmy organization with the necessary pakage installations and customizations as per given in the Prepare Your Oraganization sectionto completethe Apex Specialist Superbadge.

Challenge 3:

In this challenge we automate recordcreation using apex class and apex triggerby creating a apex class called MaintainanceRequestHelper and a apex trigger calledMaintenanceRequest.

Apex Class code:

```
public with sharing class MaintenanceRequestHelper {
public staticvoid updateWorkOrders(List<Case> caseList)
{List<case> newCases = new List<Case>();
Map<String,Integer> result=getDueDate(caseList);
for(Case c :
caseList){
if(c.status=='close
d')
if(c.type=='Repair'|| c.type=='Routine
Maintenance'){ Case newCase = new Case();
newCase.Status='New';
newCase.Origin='web';
newCase.Type='Routine Maintenance';
newCase.Subject='Routine Maintenance of
Vehicle';newCase.Vehicle_c=c.Vehicle_c;
newCase.Equipment_c=c.Equipment_c;
```

```
newCase.Date_Reported_c=Date.today();
if(result.get(c.Id)!=null)
newCase.Date_Duec=Date.today()+result.get(c.I
d);else
newCase.Date_Due_c=Date.today();
newCases.add(newCase);
}


}
insert newCases;
}
/
public static Map<String,Integer> getDueDate(List<case>
CaseIDs){Map<String,Integer> result = new
Map<String,Integer>();
Map<Id, case> caseKeys = new Map<Id, case> (CaseIDs);
List<AggregateResult> wpc=[select Maintenance_Requestr.ID
cID,min(Equipment_r.Maintenance_Cycle_c)cycle
from Work_Part_c where Maintenance_Request_r.ID in :caseKeys.keySet() group by
Maintenance_Request_r.ID];
for(AggregateResult res :wpc){
Integer addDays=0;
if(res.get('cycle')!=null)
addDays+=Integer.valueOf(res.get('cycle'));
result.put((String)res.get('cID'),addDays);
}
return result;
}
}
```

Apex Trigger code:

```
trigger MaintenanceRequest on Case (before update,after update) {
/ToDo: Call
MaintenanceRequestHelper.updateWorkOrders
if(Trigger.isAfter)
MaintenanceRequestHelper.updateWorkOrders(Trigger.Ne
w);
}
```

Challenge 4:

In challenge 3 we synchronize salesforce data with an externalsystem using apex classof name WarehouseCalloutService which is already given and after writing code in it andexecuting it anonymously in a separatewindow, the processwill be successful.

Apex class code:

```
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() {
/ ToDo: completethis method to make the callout (using@future) to the

/      REST endpoint and update equipmenton
hand.HttpResponse response = getResponse();
if(response.getStatusCode() == 200)
{
List<Product2> results = getProductList(response); / get list of productsfrom Http
calloutresponse
if(results.size() >0)
upsert resultsWarehouse_SKU_c; / Upsert the products in your org based on the externalID
SKU
}
}
/ Get the product list from the external link
public static List<Product2> getProductList(HttpResponse response)
{
List<Object> externalProducts = (List<Object>) JSON.deserializeUntyped(response.getBody());
/ desrialize the json response
List<Product2> newProducts = new
List<Product2>();for(Object p : externalProducts)
{
Map<String, Object> productMap = (Map<String, Object>)
p;Product2 pr = new Product2();
/ Map the fields in the response to the appropriate fields in the Equipment
```

```
objectpr.Replacement_Part_c = (Boolean)productMap.get('replacement');
pr.Cost_c = (Integer)productMap.get('cost');
pr.Current_Inventory_c =
(Integer)productMap.get('quantity');pr.Lifespan_Months_c
= (Integer)productMap.get('lifespan') ;
pr.Maintenance_Cycle_c = (Integer)productMap.get('maintenanceperiod');
pr.Warehouse_SKU_c = (String)productMap.get('sku');
pr.ProductCode = (String)productMap.get('_id');
pr.Name = (String)productMap.get('name');
newProducts.add(pr);
}
return newProducts;
}
/ Send Http GET request and receive Http
responsepublic staticHttpResponse
getResponse() {
Http http = new Http();
HttpRequest request = new HttpRequest();
request.setEndpoint(WAREHOUSE_URL);
request.setMethod('GET');
HttpResponse response = http.send(request);

return response;
}
}
```

Execute anonymous window:

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

Challenge 5:

In challenge 4 we will be scheduling our synchronization using WarehouseSyncSchedule in th apex class and execute a code in an anonymouswindow.

Apex Class code:

```
global  class WarehouseSyncSchedule implements Schedulable{
/  implement scheduledcode here
```

```
global void execute (SchedulableContext sc){
WarehouseCalloutService.runWarehouseEquipmentSync();
/ optionalthis can be done by debug
modeString sch = '00 00 01 * * ?';/ on 1
pm
System.schedule('WarehouseSyncScheduleTest', sch, new WarehouseSyncSchedule());
}
}
```

Execute anonymous window:

```
WarehouseSyncSchedule scheduleInventoryCheck();
```

Challenge 6:

In this challenge we are testing our automation logic using apex trigger class MaintenanceRequest and three apex classes where two are used for testing and one is used for sharing and those classesare given below.

Apex trigger:

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

Apex class:

```
@IsTest
private class InstallationTests {
private static final String STRING_TEST = 'TEST';

private static final String NEW_STATUS =
'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 =
```

```
'AMC Spirit'; public staticString CRON_EXP = '0 0 1 * * ?';
static testmethod void
testMaintenanceRequestNegative() {Vehicle_c vehicle =
createVehicle();
insert vehicle;
Id vehicleId = vehicle.Id;
Product2equipment =
createEquipment();insertequipment;
Id equipmentId = equipment.Id;
Case r = createMaintenanceRequest(vehicleId,
equipmentId);insert r;
Work_Part_c w = createWorkPart(equipmentId,
r.Id);insert w;
Test.startTest();
r.Status =
WORKING;update
r; Test.stopTest();
List<case> allRequest = [SELECT
IdFROM Case];
Work_Part_c workPart = [SELECT
IdFROM Work_Part_c
WHERE Maintenance_Request_c =:
r.Id];System.assert(workPart != null);
System.assert(allRequest.size() == 1);
}
static testmethod void testWarehouseSync() {
Test.setMock(HttpCalloutMock.class, new WarehouseCalloutServiceMock());
Test.startTest();
String jobId =
System.schedule('WarehouseSyncSchedule',CRON_EXP,
new WarehouseSyncSchedule());
CronTrigger ct = [SELECT Id, CronExpression, TimesTriggered, NextFireTime
FROM CronTrigger
WHERE id = :jobId];
System.assertEquals(CRON_EXP,  ct.CronExpression);

System.assertEquals(0, ct.TimesTriggered);
Test.stopTest();
}
```

```
private static Vehicle_c createVehicle() {
Vehicle_c v = new Vehicle_c(Name =
STRING_TEST);returnv;
}
private static Product2 createEquipment() {
Product2p = new Product2(Name =
STRING_TEST,Lifespan_Months_c = 10,
Maintenance_Cycle_c =
10, Replacement_Part_c
= true);return p;
}
privatestatic Case createMaintenanceRequest(Id vehicleId, Id equipmentId) {
Case c = new Case(Type = REPAIR,
Status =
NEW_STATUS, Origin =
REQUEST_ORIGIN,
Subject = REQUEST_SUBJECT,
Equipment_c =
equipmentId,Vehicle__c
= vehicleId); return c;
}
private staticWork_Part_c createWorkPart(Id equipmentId, Id requestId) {
Work_Part_c wp = new Work_Part_c(Equipment_c = equipmentId,
Maintenance_Request_c= requestId);
return wp;
}
}
```

Apex class:

```
public with sharing class MaintenanceRequestHelper {
public static void updateWorkOrders(List<case>
caseList){List<case> newCases = new List<case>();
Map<String,Integer> result=getDueDate(caseList);
for(Case c : caseList){
if(c.status=='closed')
if(c.type=='Repair'|| c.type=='Routine
Maintenance'){ Case newCase = new Case();
newCase.Status='New';
```

```
newCase.Origin='web';


newCase.Type='Routine Maintenance';
newCase.Subject='Routine Maintenance of
Vehicle';newCase.Vehicle_c=c.Vehicle_c;
newCase.Equipment_c=c.Equipment_c;
newCase.Date_Reported_c=Date.today();
if(result.get(c.Id)!=null)
newCase.Date_Duec=Date.today()+result.get(c.I
d);else
newCase.Date_Due_c=Date.today();
newCases.add(newCase);
}
}
insert newCases;
}
/
public static Map<String,Integer> getDueDate(List<case>
CaseIDs){Map<String,Integer> result = new
Map<String,Integer>();
Map<Id, case> caseKeys = new Map<Id, case> (CaseIDs);
List<aggregateresult> wpc=[select Maintenance_Requestr.ID
cID,min(Equipment_r.Maintenance_Cycle_c)cycle
from Work_Part_c where Maintenance_Request_r.ID in :caseKeys.keySet() group by
Maintenance_Request_r.ID];
for(AggregateResult res :wpc){
Integer addDays=0;
if(res.get('cycle')!=null)
addDays+=Integer.valueOf(res.get('cycle'));
result.put((String)res.get('cID'),addDays);
}
return result;
}
}
```

Apex class:

```
@isTest
public  class MaintenanceRequestTest {
```

```
static  List<case> caseList1 = new List<case>();
static List<product2> prodList = new List<product2>();
static List<work_part_c> wpList = new List<work_part_
c>();@testSetup
static void getData(){
caseList1=  CreateData(  300,3,3,'Repair');

}
publicstatic List<case>  CreateData( Integer numOfcase, IntegernumofProd,
IntegernumofVehicle,
String type){
List<case> caseList= new List<case>();
/ Create Vehicle
Vehicle_c vc = new Vehicle_c();
vc.name='Test Vehicle';
upsert vc;
/ Create Equiment
for(Integer
i=0;i<numofProd;i++){
Product2 prod = new
Product2();prod.Name='Test
Product'+i; if(i!=0)
prod.Maintenance_Cycle_
c=i;prod.Replacement_Part_c=true;
prodList.add(prod);
}
upsert  prodlist;
/ Create Case
for(Integer i=0;i<
numOfcase;i++){Case
newCase = new Case();
newCase.Status='New';
newCase.Origin='web';
if( math.mod(i, 2) ==0)
newCase.Type='Routine
Maintenance'; else
newCase.Type='Repair';
newCase.Subject='Routine  Maintenance  of  Vehicle'+i;
```

```
newCase.Vehicle_c=vc.Id;
if(i<numofProd)
newCase.Equipmentc=prodList.get(i).I
D;else
newCase.Equipmentc=prodList.get(0).ID;
caseList.add(newCase);
}
upsert caseList;
for(Integer i=0;i<numofProd;i++){
Work_Part_cwp = new Work_Part_
c();wp.Equipment_c  =prodlist.get(i).Id  ;
wp.Maintenance_Request_c=caseList.get(i).id;

wplist.add(wp) ;
}
upsert wplist;
return
caseList;
}
publicstatic testmethod void testMaintenanceHelper(){
Test.startTest();
getData();
for(Case cas:
caseList1)
cas.Status ='Closed';
update caseList1;
Test.stopTest();
}
}
```

Challenge 7:

In challenge 6 we are testing our callout logicby using two apex classeswhich are usedfor testingwhere one of the classesimplements HTTPCalloutMock.

Apex class:

```
@IsTest
```

```
private class WarehouseCalloutServiceTest {
/ implement your mock callout test
here@isTest
static void testWareHouseCallout(){
Test.setMock(HttpCalloutMock.class, new WarehouseCalloutServiceMock());
WarehouseCalloutService.runWarehouseEquipmentSync();
}
}
```

Apex class:

```
@isTest
public class WarehouseCalloutServiceMock implements HTTPCalloutMock {
/  implement http mock callout
publicHTTPResponse respond (HttpRequest
request){HttpResponse response = new
HTTPResponse(); response.setHeader('Content-
type','application/json');
response.setBody('[{"_id":"55d66226726b611100aaf741","replacement":false,"quantity":5,"nam
e": "Generator 1000
kW","maintenanceperiod":365,"lifespan":120,"cost":5000,"sku":"100003"},{"_id":"55d66226726b6
11


100aaf742","replacement":true,"quantity":183,"name":"Cooling
Fan","maintenanceperiod":0,"lifespan":0,"cost":300,"sku":"100004"},{"_id":"55d66226726b61110
0a af743","replacement":true,"quantity":143,"name":"Fuse
20A","maintenanceperiod":0,"lifespan":0,"cost":22,"sku":"100005"}]');
response.setStatusCode(200);
return response;
}
}
```

Challenge 8:

In this challenge we are testing our Scheduling logic by usinga apex test class to testour scheduling logic and the code is given below.

Apex class:

```
@isTest
private class WarehouseSyncScheduleTest {
public static String CRON_EXP = '0 0 0 15 3 ?
2022';static testmethod void testjob(){
MaintenanceRequestTest.CreateData(
5,2,2,'Repair'); Test.startTest();
Test.setMock(HttpCalloutMock.class, new WarehouseCalloutServiceMock());
String joBID=System.schedule('TestScheduleJob', CRON_EXP, new
WarehouseSyncSchedule());
/ List<Case> caselist = [Select count(id) from case where
case]Test.stopTest();
}
}
```

with this the Apex Specialist Superbadge is completed succesfully.

Process AutomationSpecialist Superbadge

Challenge 1:

It is the same as the previoussuperbadge challenge 1 where we answer a quiz beforemoving into the actual Superbadge challenges.

Challenge 2:

This challenge is all about automating leads where we create a Validation rule under leads and you can give any Rule Name and the Error condition fomulawill be given below for validating leads.After this we have to create two Queues with the given name as per in the instruction of the challenge and then create a assignment rule.If all these things are done properly, the challenge will be completed without any problems.

Error Condition Formula:

```
OR(AND(LEN(State) > 2,
NOT(CONTAINS("AL:AK:AZ:AR:CA:CO:CT:DE:DC:FL:GA:HI:ID:IL:IN:IA:KS:KY:LA:ME:MD:MA:MI
:M
N:MS:MO:MT:NE:NV:NH:NJ:NM:NY:NC:ND:OH:OK:OR:PA:RI:SC:SD:TN:TX:UT:VT:VA:WA:WV:WI:
WY", State )) ), NOT(OR(Country ="US",Country ="USA",Country ="United States",
```

ISBLANK(Country))))

Challenge 3:

In this challenge we are given the task of automating accounts by creating Roll Up Summmary fileds as it is given in the instructions and after that by creatingtwo Error ConditionFormulas we automateour accounts and the code will be given below for these two formulas

Error Condition Formula1:

```
OR(AND(LEN(BillingState) > 2,
NOT(CONTAINS("AL:AK:AZ:AR:CA:CO:CT:DE:DC:FL:GA:HI:ID:IL:IN:IA:KS:KY:LA:ME:MD:MA:MI
:M
N:MS:MO:MT:NE:NV:NH:NJ:NM:NY:NC:ND:OH:OK:OR:PA:RI:SC:SD:TN:TX:UT:VT:VA:WA:WV:WI:
WY", BillingState ))
),AND(LEN(ShippingState) > 2,
NOT(CONTAINS("AL:AK:AZ:AR:CA:CO:CT:DE:DC:FL:GA:HI:ID:IL:IN:IA:KS:KY:LA:ME:MD:MA:MI
:M
N:MS:MO:MT:NE:NV:NH:NJ:NM:NY:NC:ND:OH:OK:OR:PA:RI:SC:SD:TN:TX:UT:VT:VA:WA:WV:WI:
WY", ShippingState))
),NOT(OR(BillingCountry ="US",BillingCountry ="USA",BillingCountry ="United States",
```

ISBLANK(BillingCountry))),
NOT(OR(ShippingCountry ="US",ShippingCountry="USA",ShippingCountry ="United States",
ISBLANK(ShippingCountry))))

Error Condition Formula2:

ISCHANGED(Name) && (OR(ISPICKVAL(Type ,'Customer - Direct') ,ISPICKVAL(Type ,'Customer - Channel')))

Challenge 4:

It is the easiest challenge in this superbadge where we dont have to do a lot of things, we only have to createRobot Setup object with a master-detail relationship with the opportunity and the createa few fields as per given in the challenge instructions.

Challenge 5:

In this challenge we are creating a Sales Processand Validating its opportuities, First wehave to create a field with checkboxtype with the name Approvalwhere it can only be viewed by System Administrators and Sales Managers.Then we have add a picklist value as Awating Approval to the filed Stage.Lastly we have to add the desired fields and then add a Validation rule in the Opportunity object.

Validation Rule:

IF((Amount > 100000 && Approved_c <> True && ISPICKVAL(StageName,'Closed Won')),True,False)

Challenge 6:

In this challenge we are Automating Opportunities, First we have to create three Email Templates upon reading instructions and create a approval process by selecting opportunity object in the approval processwith the necessaryfield updates in the processand set a criteria where this processwill only run if the criteria is met.

Then go to the processbuilder and startbuilding a processby selecting a object firstand by setting four criterias where each criteriawill do a action upon meeting the criterias.

Challenge 7:

In this challenge we are creatingFlow for Opportunities, First with a Start elementthen Screen

element where it then gets Records and there's a loop to get each record and after that the process ends with a screenelement where it shows the products.The productsare created as per given in the challenge instructions to successfully completethe challenge.

Challenge 8:

It is the last challenge of the superbadge where we Automate Setups, First we have to change the formula in one of the fields of the Robot object where the Formula will be given below and then we have go to the flows process that we createdpreviously and clone it to makechanges where we change the formula for the last criteria to Automate setups according to dates.

Formula 1:

```
Case ( WEEKDAY( Datec ),
1,"Sunday",
2,"Monday",
3,"Tuesday",
4,"Wednesday",
5,"Thursday",
6,"Friday",
7,"Saturday",
Text(WEEKDay(Date_c)))
```

Formula 2:

```
CASE(MOD([Opportunity].CloseDate + 180 - DATE(1900, 1, 7),7), 0, [Opportunity].CloseDate +
181, 6, [Opportunity].CloseDate + 182, [Opportunity].CloseDate + 180)
```

And with this you will have successfully completedthis Superbadge.

Apex Triggers

Get Started with Apex Triggers:

Apex trigger:

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



List<Account> acclst=new List<Account>();
  for(account a:trigger.new){
    if(a.Match_Billing_Addressc==true && a.BillingPostalCode!=null){
    a.ShippingPostalCode=a.BillingPostalCode;

    }

}
}
```

Bulk Apex Triggers:

Apex Trigger:

```
trigger ClosedOpportunityTrigger on Opportunity (afterinsert, after update){

    List <Task> tasks = new List<Task>();
    for(Opportunity opp : [SELECTId, StageName FROM Opportunity WHERE StageName='Closed
Won'
                           AND Id IN :Trigger.new]){
        tasks.add(new Task(Subject = 'Follow Up Test Task' , WhatId = opp.Id));
    }

    if(tasks.size()
```

```
        > 0){insert
        tasks;
    }
}
```

Apex Testing

Get Started with Apex Unit Tests:

Apex class:

```
@isTest
private class TestVerifyDate {
    @isTest staticvoid testWithin30Days() {
        Date Datetest = VerifyDate.CheckDates(System.today(), System.today()+10);
        System.assertEquals(System.today()+10, Datetest);
    }

     @isTest static void testSetEndOfMonth() {
        Date Datetest = VerifyDate.CheckDates(System.today(), System.today()+52);
        System.assertEquals(System.today()+27, Datetest); <!--27days until last day of
        Current
Month-->
     }

    }
```

Test Apex Triggers:

Apex Class:

```
@isTest
private class

    TestRestrictContactByName {
```

```
    statictestMethod void  metodoTest()


    {

        List<Contact> listContact= new List<Contact>();
        Contact c1 = new Contact(FirstName='Francesco', LastName='Riggio' ,
email='Test@test.com');
        Contact c2 = new Contact(FirstName='Francesco1', LastName=
'INVALIDNAME',email='Test@test.com');
        listContact.add(c1);
        listContact.add(c2);

        Test.startTes
           t();try
           {
              insert  listContact;
           }
           catch(Exception ee)
           {
           }

        Test.stopTest();

    }

}
```

Create Test Data for Apex Tests:

Apex Class:

```
public with sharing class RandomContactFactory
{
          public static List<Contact> generateRandomContacts( Integer noOfContacts, String
lastName)
          {
```

```
        List<Contact> contacts= new List<Contact>();

        for( Integer i = 0; i < noOfContacts; i++ )
        {
            Contact con = new Contact( FirstName = 'Test '+i, LastName = lastName
);
            contacts.add( con );

        }

        return contacts;
    }
}
```

Asynchronous Apex

Use Future Methods:

Apex class:

```
public class AccountProcessor{
  @future
  public static void countContacts(List<Id> accountIds){ List<Account> vAccountList = new List<Account>(); List<Account> acc = [SELECTId,Name,
                (SELECT Id,NameFROM Contacts)
                FROM Account WHEREId IN :accountIds]; System.debug('total contactin Account: ' + acc);

    if(acc.size() > 0){
      for(Account a:
      acc){
```

```
            List<Contact> con = [SELECTId,Name FROM ContactWHERE accountId = :a.Id];
            a.Number_of_Contacts_c = con.size();
            vAccountList.add(a);
        }
        if(vAccountList.size()>0)
        {
            update vAccountList;
        }
    }
  }
}
```

Test Class:

==========

```
@isTest
public class AccountProcessorTest {
  @isTest public static void
    testNoOfContacts(){ Account a = new
    Account(Name = 'Acme1');Insert a;
    Account b = new Account(Name = 'Acme2');
    insertb;
    Contactc = new Contact(FirstName = 'Gk', LastName= 'Gupta', accountId=
    a.Id);insertc;
    Contactc1 = new Contact(FirstName = 'Gk1', LastName= 'Gupta1', accountId= b.Id);
    insertc1;

    List<account> acnt = [SELECTId FROM Account WHERE Name = :a.NameOR Name =
:b.Name];
    System.debug('size of acnt: ' +
    acnt); List<ID> acntIDLST = new
    List<Id>();for(Account ac: acnt){
      acntIDLST.add(ac.Id);
    }
    Test.startTest();
    AccountProcessor.countContacts(acntIDLST);
```

```
        Test.stopTest();
    }
}
```

Use Batch Apex:

Apex Class:

```
global class LeadProcessor implements        Database.Batchable<Sobject>
{
    global  Database.QueryLocator  start(Database.BatchableContext  bc)
    {
        return Database.getQueryLocator([Select LeadSource From Lead ]);
    }

    global void execute(Database.BatchableContext bc, List<Lead> scope)
    {
          for (Lead Leads : scope)
          {
              Leads.LeadSource = 'Dreamforce';

          }
        update scope;
    }

    global void finish(Database.BatchableContext bc){ }
}

@isTest
public class LeadProcessorTest
{
    static testMethod void testMethod1()
    {
        List<Lead> lstLead = new List<Lead>();
        for(Integer i=0 ;i <200;i++)
        {
```

```
        Lead led = new Lead();
        led.FirstName
        ='FirstName';led.LastName
        ='LastName'+i;led.Company
        ='demo'+i;lstLead.add(led);
      }

      insert

      lstLead;

      Test.startTes

      t();

        LeadProcessor obj = new LeadProcessor();
        DataBase.executeBatch(obj);

      Test.stopTest();
    }
}
```

Control Processes with Queueable Apex:

Apex Class:

```
public class AddPrimaryContact implements Queueable
{
    private Contact
    c; private String
    state;

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

```
    public void execute(QueueableContext context)
    {
        List<Account> ListAccount = [SELECT ID, Name ,(Selectid,FirstName,LastName from
contacts ) FROM ACCOUNTWHERE BillingState = :state LIMIT200];
        List<Contact> lstContact = new
        List<Contact>();for (Account
        acc:ListAccount)
        {
            Contact cont = c.clone(false,false,false,false);
            cont.AccountId =  acc.id;
            lstContact.add( cont );
        }

        if(lstContact.size() >0)
        {
           insert lstContact;
        }

    }

}

@isTest
public class AddPrimaryContactTest
{
    @isTest staticvoid TestList()
    {
        List<Account> Teste = new List
        <Account>();for(Integer i=0;i<50;i++)
        {
           Teste.add(new Account(BillingState = 'CA', name = 'Test'+i));
        }
        for(Integer j=0;j<50;j++)
        {
           Teste.add(new Account(BillingState = 'NY', name = 'Test'+j));
        }
```

```
        insert Teste;


        Contact co = new
        Contact();
        co.FirstName='demo';
        co.LastName ='demo';
        insert co;
        String state = 'CA';

         AddPrimaryContact apc = new AddPrimaryContact(co,
         state);Test.startTest();
           System.enqueueJob(apc);
         Test.stopTest();
      }
  }
```

Schedule Jobs Using the Apex Scheduler:

Apex Class:

```
  global class DailyLeadProcessor implements Schedulable

     {global void execute(SchedulableContext ctx) {

        List<Lead> lList = [SelectId, LeadSource from Lead where LeadSource = null];

        if(!lList.isEmpty()) {
                                for(Lead l: lList) {
                                        l.LeadSource = 'Dreamforce';
                                }
                                update lList;
                        }
     }
  }
```

```
@isTest
public class DailyLeadProcessorTest {
   public static String CRON_EXP = '0 0 0 15 3 ? 2022';
   static testMethodvoid testDailyLeadProcessorTest() {
      List<Lead> listLead = new
      List<Lead>();for (Integer i=0; i<200;
      i++) {

         Lead ll = new Lead();
         ll.LastName = 'Test' + i;
         ll.Company = 'Company'
         + i;
         ll.Status = 'Open - Not Contacted';
         listLead.add(ll);
      }
      insert listLead;

      Test.startTest();
         DailyLeadProcessor daily = new DailyLeadProcessor();
         String jobId = System.schedule('Update LeadSource to Dreamforce', CRON_EXP, daily);

         List<Lead> liss = new List<Lead>([SELECT Id, LeadSource FROM Lead
WHERELeadSource != 'Dreamforce']);
      Test.stopTest();
   }
}
```

Apex Integration Services

Apex Rest Callouts:

Apex Class:

```
public class AnimalLocator {
   public static String getAnimalNameById(Integer
      id) {Http http = new Http();
      HttpRequest request= new HttpRequest();
      request.setEndpoint('https:/ th-apex-http-callout.herokuapp.com/animals/'+id);
      request.setMethod('GET');
      HttpResponse response= http.send(request);
         /*Map<String,Object> results =
(Map<String,Object>)JSON.deserializeUntyped(response.getBody());
      system.debug('---- >results'+results);
      List<Object>animals = (List<Object>)
      results.get('animal'); system.debug('----------------------
      >animal'+animals);*/
      Map<Integer,String> mapAnimal= new Map<Integer,String>();

      Integer varId;
      String
      varName;
      JSONParserparser1= JSON.createParser(response.getBody());
      while(parser1.nextToken() != null) {
         if ((parser1.getCurrentToken() == JSONToken.FIELD_NAME) && (parser1.getText() ==

'id')) {









                                                                                    }

/ Get the value.parser1.nextToken();
/ Fetch the ids for all animals in JSON Response.
varId=parser1.getIntegerValue();
System.debug('---- >varId-->'+varID);
```

```
parser1.nextToken();

        if ((parser1.getCurrentToken() == JSONToken.FIELD_NAME) && (parser1.getText() ==
'name')) {
            parser1.nextToken();
            / Fetch the names for all animals in JSON
            Response.varName=parser1.getText();
            System.debug('---- >varName-->'+varName);
        }
        mapAnimal.put(varId,varName);
      }
      system.debug('---- >mapAnimal-->'+mapAnimal);
      return  mapAnimal.get(id);

   }
}
```

Mock Test Class:

```
@isTest
global class AnimalLocatorMock implements HttpCalloutMock {
   /  Implement this interface method
   global HTTPResponse respond(HTTPRequest request){
      / Create a fake response
      HttpResponse response = new HttpResponse();
      response.setHeader('Content-Type',
      'application/json');
      response.setBody('{"animal":[{"id":1,"name":"chicken","eats":"chicken food","says":"cluck
cluck"},{"id":2,"name":"duck","eats":"worms","says":"pek pek"}]}');
      response.setStatusCode(200);
      return response;
   }

}
```

Test Class:

```
@isTest
```

```
private class AnimalLocatorTest {
@isTeststatic void
testGetCallout() {
   /  Set mock calloutclass
   Test.setMock(HttpCalloutMock.class, new AnimalLocatorMock());
   /  This causes a fake responseto be sent
   / from the class that implements HttpCalloutMock.
   String response =
   AnimalLocator.getAnimalNameById(1);
   system.debug('Test Response1--->'+response);
   String expectedValue = 'chicken';
   System.assertEquals(expectedValue,response);
   String response2 =
   AnimalLocator.getAnimalNameById(2);
   system.debug('Test Response2--->'+response2);
   String expectedValue2 = 'duck';
   System.assertEquals(expectedValue2,response2);
}
}
```

Apex SOAP Callouts:

Apex Class:

```
Service:
/ Generated by wsdl2apex

public class ParkService {
   publicclass
      byCountryResponse {public
      String[]return_x;
      private String[] return_x_type_info = new String[]{'return','http:/ parks.services/',null,'0','-
1','false'};
      private String[] apex_schema_type_info = new String[]{'http:/
      parks.services/','false','false'}; private String[] field_order_type_info = new
      String[]{'return_x'};
   }
```

```
public class
  byCountry {public
  String arg0;
  private String[] arg0_type_info = new String[]{'arg0','http:/
  parks.services/',null,'0','1','false'}; privateString[] apex_schema_type_info = new
  String[]{'http:/ parks.services/','false','false'};


  private String[] field_order_type_info = new String[]{'arg0'};
}
public class ParksImplPort {
  public String endpoint_x= 'https:/ th-apex-soap-service.herokuapp.com/service/parks';
  public Map<String,String> inputHttpHeaders_x;
  public Map<String,String>
  outputHttpHeaders_x; public String
  clientCertName_x;
  public String clientCert_x;
  public String
  clientCertPasswd_x;public
  Integer timeout_x;
  private String[] ns_map_type_info = new String[]{'http:/ parks.services/',
  'ParkService'}; public String[] byCountry(String arg0) {
    ParkService.byCountry request_x = new ParkService.byCountry();
    request_x.arg0 = arg0;
    ParkService.byCountryResponse  response_x;
    Map<String, ParkService.byCountryResponse> response_map_x = new Map<String,
ParkService.byCountryResponse>();
    response_map_x.put('response_x', response_x);
    WebServiceCallout.invoke(
     this,
     reque
     st_x,
     response_map_x,
     new
     String[]{endpoint_x,'',
     'http:/ parks.services/',
     'byCountry',
     'http:/ parks.services/',
     'byCountryResponse',
     'ParkService.byCountryResponse'}
```

```
        );
        response_x = response_map_x.get('response_x');
        returnresponse_x.return_x;
      }
   }
}
```

Class:

```
public class ParkLocator {
   public static String[]country(String country){
      ParkService.ParksImplPort parks = new
      ParkService.ParksImplPort();String[] parksname =
      parks.byCountry(country);

      return parksname;
   }
}
```

Test:

```
@isTest
privateclass ParkLocatorTest{
   @isTest
   static void testParkLocator() {
      Test.setMock(WebServiceMock.class, new
      ParkServiceMock());String[]arrayOfParks =
      ParkLocator.country('India');

      System.assertEquals('Park1', arrayOfParks[0]);
   }
}
```

Mock Test:

```
@isTest
global class ParkServiceMock implements WebServiceMock {
   globalvoid doInvoke(
        Object
```

```
        stub,
        Object
        request,
        Map<String, Object>
        response,String endpoint,
        String soapAction,
        String
        requestName,
        String
        responseNS,
        String
        responseName,
        String
        responseType) {
      ParkService.byCountryResponse response_x = new ParkService.byCountryResponse();
      List<String> lstOfDummyParks = new List<String> {'Park1','Park2','Park3'};
      response_x.return_x = lstOfDummyParks;

      response.put('response_x', response_x);
   }
}
```

Apex Web Services:

Apex Class:

```
@RestResource(urlMapping='/Accounts/*/cont
acts') global with sharing
classAccountManager{
   @HttpGet
   global static Account getAccount(){
      RestRequest request =
      RestContext.request;
      String accountId = request.requestURI.substringBetween('Accounts/','/contacts');
      system.debug(accountId);
      Account objAccount = [SELECT Id,Name,(SELECT Id,Name FROM Contacts)FROM Account
WHERE Id = :accountId LIMIT 1];
      return objAccount;
```

```
    }
}


/ Test
class
@isTest
private class AccountManagerTest{
    statictestMethod void
    testMethod1(){
        AccountobjAccount = new Account(Name = 'test Account');
        insert objAccount;
        Contact objContact = new Contact(LastName = 'test Contact',
                                         AccountId = objAccount.Id);
        insert objContact;
        Id recordId= objAccount.Id;
        RestRequest request = new RestRequest();
        request.requestUri =
            'https:/   sandeepidentity-dev-ed.my.salesforce.com/services/apexrest/Accounts/'
            + recordId +'/contacts';
        request.httpMethod = 'GET';
        RestContext.request = request;
        / Call the methodto test
        Account thisAccount = AccountManager.getAccount();
        / Verify results
        System.assert(thisAccount!= null);
        System.assertEquals('test  Account',  thisAccount.Name);
    }
}
```

Lightning Web Components

Deploy Lightning Web Component Files:

bikeCard.html:

```
<template>
    <div>
        <div>Name: {name}</div>
        <div>Description:  {description}</div>
        <lightning-badge  label={material}></lightning-badge>
        <lightning-badge  label={category}></lightning-badge>
        <div>Price: {price}</div>
        <div><img src={pictureUrl}/></div>
    </div>
</template>
```

bikeCard.js:

```
import { LightningElement } from 'lwc';
export default class BikeCardextends LightningElement {
    name = 'Electra X4';
    description = 'A sweet bike built for comfort.';
    category= 'Mountain';
    material =
    'Steel';price =
    '$2,700';
    pictureUrl = 'https:/s3-us-west-1.amazonaws.com/sfdc-demo/ebikes/electrax4.jpg';
 }
```

bikeCard.js-meta.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle  xmlns="http:/   soap.sforce.com/2006/04/metadata">
    <!-- The apiVersionmay need to be increasedfor the current release -->
    <apiVersion>52.0</apiVersion>
    <isExposed>true</isExposed>
    <masterLabel>Product Card</masterLabel>
    <targets>
        <target>lightning_AppPage</target>
        <target>lightning_RecordPage</target>
        <target>lightning_HomePage</target>
    </targets>
</LightningComponentBundle>
```