Apex-Superbadge:

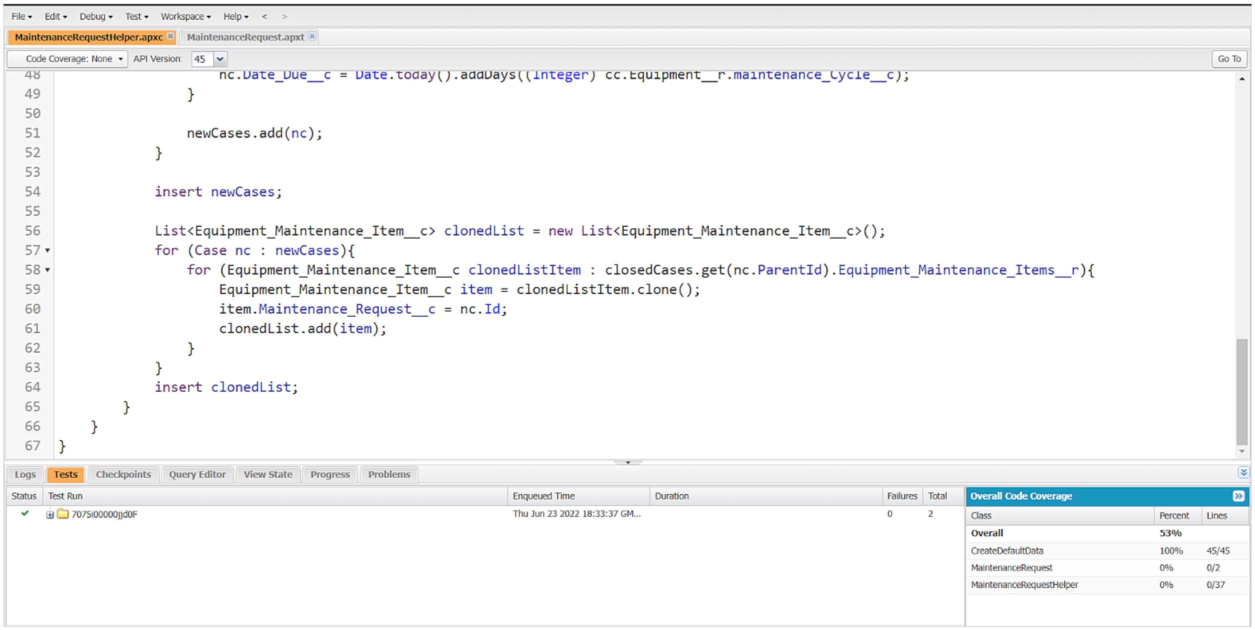
1)Automate record creation

Code:

```
public with sharing class MaintenanceRequestHelper {
  public static void updateworkOrders(List<Case> updWorkOrders, Map<Id,Case>
nonUpdCaseMap) {
    Set<Id> validIds = new Set<Id>();
    For (Case c : updWorkOrders){
      if (nonUpdCaseMap.get(c.Id).Status != 'Closed' && c.Status == 'Closed'){
        if (c.Type == 'Repair' || c.Type == 'Routine Maintenance'){
          validIds.add(c.Id);
        }
      }
    }

    //When an existing maintenance request of type Repair or Routine Maintenance is
closed,
    //create a new maintenance request for a future routine checkup.
    if (!validIds.isEmpty()){
      Map<Id,Case> closedCases = new Map<Id,Case>([SELECT Id, 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>();

      //calculate the maintenance request due dates by using the maintenance cycle
defined on the related equipment records.
      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'));
        }

        List<Case> newCases = new List<Case>();
        for(Case cc : closedCases.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 multiple pieces of equipment are used in the maintenance request,
          //define the due date by applying the shortest maintenance cycle to today's
date.
          If (maintenanceCycles.containskey(cc.Id)){
            nc.Date_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> clonedList = new
List<Equipment_Maintenance_Item__c>();
            for (Case nc : newCases){
                for (Equipment_Maintenance_Item__c clonedListItem :
closedCases.get(nc.ParentId).Equipment_Maintenance_Items__r){
                    Equipment_Maintenance_Item__c item = clonedListItem.clone();
                    item.Maintenance_Request__c = nc.Id;
                    clonedList.add(item);
                }
            }
            insert clonedList;
        }
    }
}

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



Code Coverage: None API Version: 45

```
                    nc.Date_Due__c = Date.today().addDays((Integer) cc.Equipment__r.maintenance_Cycle__c);
                }

                newCases.add(nc);
            }

            insert newCases;

            List<Equipment_Maintenance_Item__c> clonedList = new List<Equipment_Maintenance_Item__c>();
            for (Case nc : newCases){
                for (Equipment_Maintenance_Item__c clonedListItem : closedCases.get(nc.ParentId).Equipment_Maintenance_Items__r){
                    Equipment_Maintenance_Item__c item = clonedListItem.clone();
                    item.Maintenance_Request__c = nc.Id;
                    clonedList.add(item);
                }
            }
            insert clonedList;
        }
    }
}
```

Status	Test Run	Enqueued Time	Duration	Failures	Total
✓	7075i00000jjd0F	Thu Jun 23 2022 18:33:37 GM...		0	2

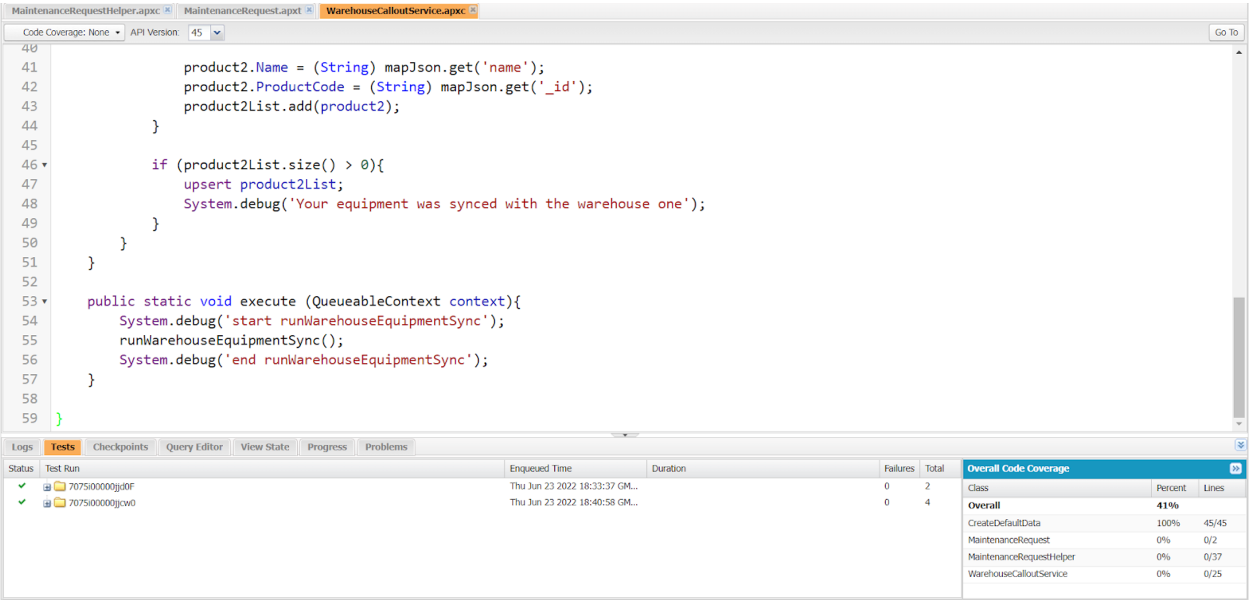
Overall Code Coverage

Class	Percent	Lines
Overall	**53%**	
CreateDefaultData	100%	45/45
MaintenanceRequest	0%	0/2
MaintenanceRequestHelper	0%	0/37

<u>2)Synchronize Salesforce data with an external system</u>

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

    //Write a class that makes a REST callout to an external warehouse system to get a list of equipment that needs to be updated.
    //The callout’s JSON response returns the equipment records that you upsert in Salesforce.

    @future(callout=true)
    public static void runWarehouseEquipmentSync(){
        System.debug('go into runWarehouseEquipmentSync');
        Http http = new Http();
        HttpRequest request = new HttpRequest();

        request.setEndpoint(WAREHOUSE_URL);
        request.setMethod('GET');
        HttpResponse response = http.send(request);
    List<Product2> product2List = new List<Product2>();
    System.debug(response.getStatusCode());
    if (response.getStatusCode() == 200){
      List<Object> jsonResponse =
(List<Object>)JSON.deserializeUntyped(response.getBody());
      System.debug(response.getBody());

      //class maps the following fields:
      //warehouse SKU will be external ID for identifying which equipment records to
update within Salesforce
      for (Object jR : jsonResponse){
        Map<String,Object> mapJson = (Map<String,Object>)jR;
        Product2 product2 = new Product2();
        //replacement part (always true),
        product2.Replacement_Part__c = (Boolean) mapJson.get('replacement');
        //cost
        product2.Cost__c = (Integer) mapJson.get('cost');
        //current inventory
        product2.Current_Inventory__c = (Double) mapJson.get('quantity');
        //lifespan
        product2.Lifespan_Months__c = (Integer) mapJson.get('lifespan');
        //maintenance cycle
        product2.Maintenance_Cycle__c = (Integer)
mapJson.get('maintenanceperiod');
        //warehouse SKU
        product2.Warehouse_SKU__c = (String) mapJson.get('sku');

        product2.Name = (String) mapJson.get('name');
        product2.ProductCode = (String) mapJson.get('_id');
        product2List.add(product2);
      }
```

```
            if (product2List.size() > 0){
                upsert product2List;
                System.debug('Your equipment was synced with the warehouse one');
            }
        }
    }

    public static void execute (QueueableContext context){
        System.debug('start runWarehouseEquipmentSync');
        runWarehouseEquipmentSync();
        System.debug('end runWarehouseEquipmentSync');
    }

}
```



```
                product2.Name = (String) mapJson.get('name');
                product2.ProductCode = (String) mapJson.get('_id');
                product2List.add(product2);
            }

            if (product2List.size() > 0){
                upsert product2List;
                System.debug('Your equipment was synced with the warehouse one');
            }
        }
    }

    public static void execute (QueueableContext context){
        System.debug('start runWarehouseEquipmentSync');
        runWarehouseEquipmentSync();
        System.debug('end runWarehouseEquipmentSync');
    }

}
```

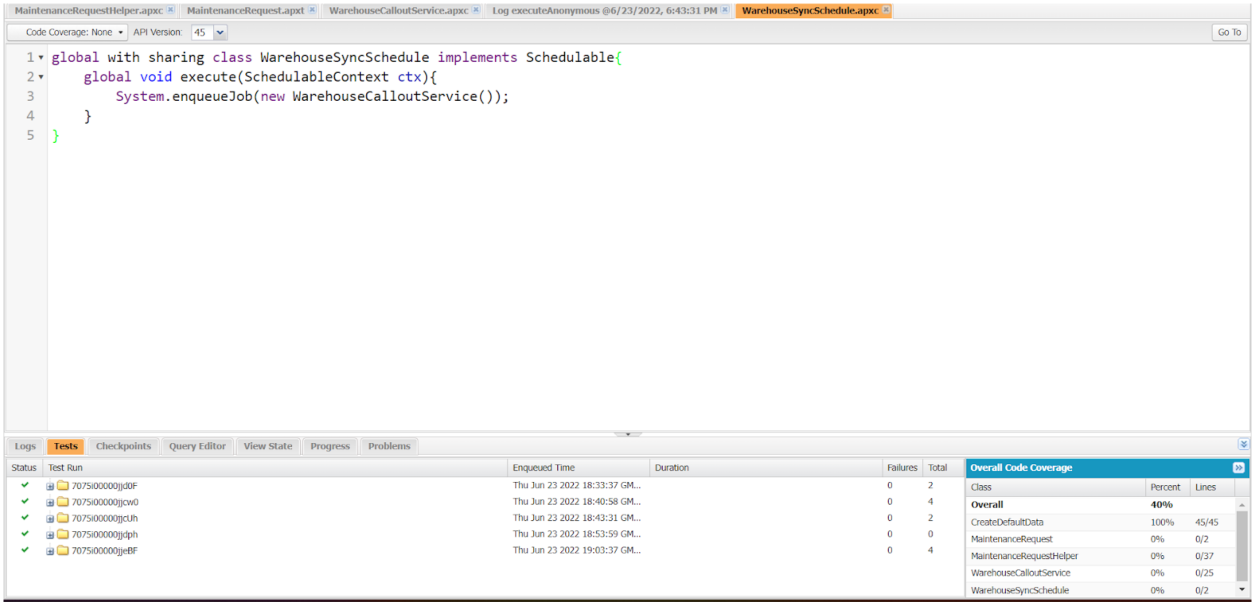
Status	Test Run	Enqueued Time	Duration	Failures	Total
✓	7075i00000jjd0F	Thu Jun 23 2022 18:33:37 GM...		0	2
✓	7075i00000jjcw0	Thu Jun 23 2022 18:40:58 GM...		0	4

Overall Code Coverage

Class	Percent	Lines
Overall	**41%**	
CreateDefaultData	100%	45/45
MaintenanceRequest	0%	0/2
MaintenanceRequestHelper	0%	0/37
WarehouseCalloutService	0%	0/25

3)Schedule synchronization

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


MaintenanceRequestHelper.apxc
MaintenanceRequest.apxt
WarehouseCalloutService.apxc
Log executeAnonymous @6/23/2022, 6:43:31 PM
WarehouseSyncSchedule.apxc
Code Coverage: None
API Version: 45
Go To
Logs
Tests
Checkpoints
Query Editor
View State
Progress
Problems

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

Status	Test Run	Enqueued Time	Duration	Failures	Total
✓	7075i00000jjd0F	Thu Jun 23 2022 18:33:37 GM...		0	2
✓	7075i00000jjcw0	Thu Jun 23 2022 18:40:58 GM...		0	4
✓	7075i00000jjcUh	Thu Jun 23 2022 18:43:31 GM...		0	2
✓	7075i00000jjdph	Thu Jun 23 2022 18:53:59 GM...		0	0
✓	7075i00000jjeBF	Thu Jun 23 2022 19:03:37 GM...		0	4

Overall Code Coverage

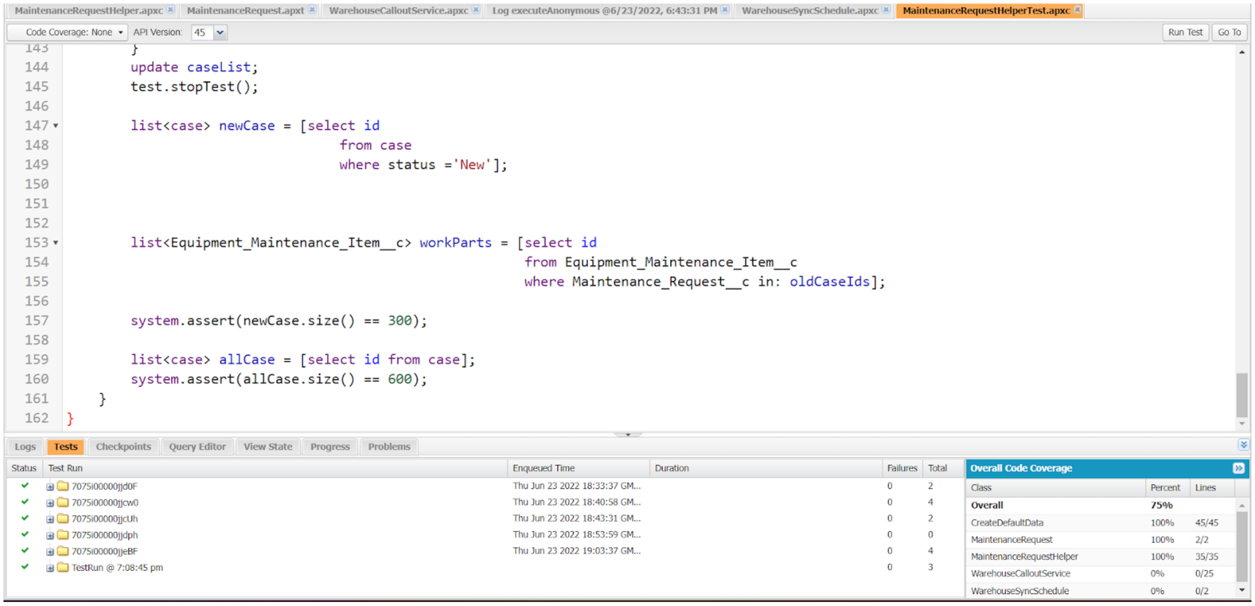
Class	Percent	Lines
Overall	**40%**	
CreateDefaultData	100%	45/45
MaintenanceRequest	0%	0/2
MaintenanceRequestHelper	0%	0/37
WarehouseCalloutService	0%	0/25
WarehouseSyncSchedule	0%	0/2

4)Test automation logic

```
public with sharing class MaintenanceRequestHelper {
   public static void updateworkOrders(List<Case> updWorkOrders, Map<Id,Case> nonUpdCaseMap) {
      Set<Id> validIds = new Set<Id>();
      For (Case c : updWorkOrders){
         if (nonUpdCaseMap.get(c.Id).Status != 'Closed' && c.Status == 'Closed'){
            if (c.Type == 'Repair' || c.Type == 'Routine Maintenance'){
               validIds.add(c.Id);
            }
         }
      }

      //When an existing maintenance request of type Repair or Routine Maintenance is closed,
      //create a new maintenance request for a future routine checkup.
      if (!validIds.isEmpty()){
         Map<Id,Case> closedCases = new Map<Id,Case>([SELECT Id, 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>();

        //calculate the maintenance request due dates by using the maintenance cycle
defined on the related equipment records.
        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'));
        }

        List<Case> newCases = new List<Case>();
        for(Case cc : closedCases.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 multiple pieces of equipment are used in the maintenance request,
                //define the due date by applying the shortest maintenance cycle to today’s
date.
                //If (maintenanceCycles.containskey(cc.Id)){
                    nc.Date_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> clonedList = new
List<Equipment_Maintenance_Item__c>();
            for (Case nc : newCases){
                for (Equipment_Maintenance_Item__c clonedListItem :
closedCases.get(nc.ParentId).Equipment_Maintenance_Items__r){
                    Equipment_Maintenance_Item__c item = clonedListItem.clone();
                    item.Maintenance_Request__c = nc.Id;
                    clonedList.add(item);
                }
            }
            insert clonedList;
        }
    }
}
trigger MaintenanceRequest on Case (before update, after update) {
    if(Trigger.isUpdate && Trigger.isAfter){
        MaintenanceRequestHelper.updateWorkOrders(Trigger.New, Trigger.OldMap);
    }
}
@isTest
public with sharing class MaintenanceRequestHelperTest {

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

    // createEquipment
    private static Product2 createEquipment(){
        product2 equipment = new product2(name = 'Testing equipment',
                                          lifespan_months__c = 10,
                                          maintenance_cycle__c = 10,
                                          replacement_part__c = true);
        return equipment;
    }

    // createMaintenanceRequest
    private static Case createMaintenanceRequest(id vehicleId, id equipmentId){
        case cse = new case(Type='Repair',
                            Status='New',
                            Origin='Web',
                            Subject='Testing subject',
                            Equipment__c=equipmentId,
                            Vehicle__c=vehicleId);
        return cse;
    }
    // createEquipmentMaintenanceItem
    private static Equipment_Maintenance_Item__c createEquipmentMaintenanceItem(id
equipmentId,id requestId){
        Equipment_Maintenance_Item__c equipmentMaintenanceItem = new
Equipment_Maintenance_Item__c(
            Equipment__c = equipmentId,
            Maintenance_Request__c = requestId);
        return equipmentMaintenanceItem;
    }

    @isTest
    private static void testPositive(){
        Vehicle__c vehicle = createVehicle();
        insert vehicle;
        id vehicleId = vehicle.Id;

        Product2 equipment = createEquipment();
        insert equipment;
        id equipmentId = equipment.Id;

        case createdCase = createMaintenanceRequest(vehicleId,equipmentId);
        insert createdCase;

        Equipment_Maintenance_Item__c equipmentMaintenanceItem =
createEquipmentMaintenanceItem(equipmentId,createdCase.id);
        insert equipmentMaintenanceItem;

        test.startTest();
        createdCase.status = 'Closed';
        update createdCase;
        test.stopTest();
    Case newCase = [Select id,
                    subject,
                    type,
                    Equipment__c,
                    Date_Reported__c,
                    Vehicle__c,
                    Date_Due__c
                   from case
                   where status ='New'];

    Equipment_Maintenance_Item__c workPart = [select id
                                              from Equipment_Maintenance_Item__c
                                              where Maintenance_Request__c =:newCase.Id];
    list<case> allCase = [select id from case];
    system.assert(allCase.size() == 2);

    system.assert(newCase != null);
    system.assert(newCase.Subject != null);
    system.assertEquals(newCase.Type, 'Routine Maintenance');
    SYSTEM.assertEquals(newCase.Equipment__c, equipmentId);
    SYSTEM.assertEquals(newCase.Vehicle__c, vehicleId);
    SYSTEM.assertEquals(newCase.Date_Reported__c, system.today());
}

@isTest
private static void testNegative(){
    Vehicle__C vehicle = createVehicle();
    insert vehicle;
    id vehicleId = vehicle.Id;

    product2 equipment = createEquipment();
    insert equipment;
    id equipmentId = equipment.Id;

    case createdCase = createMaintenanceRequest(vehicleId,equipmentId);
    insert createdCase;

    Equipment_Maintenance_Item__c workP =
createEquipmentMaintenanceItem(equipmentId, createdCase.Id);
    insert workP;

    test.startTest();
    createdCase.Status = 'Working';
    update createdCase;
    test.stopTest();

    list<case> allCase = [select id from case];

    Equipment_Maintenance_Item__c equipmentMaintenanceItem = [select id
                          from Equipment_Maintenance_Item__c
                          where Maintenance_Request__c = :createdCase.Id];

    system.assert(equipmentMaintenanceItem != null);
    system.assert(allCase.size() == 1);
  }

  @isTest
  private static void testBulk(){
    list<Vehicle__C> vehicleList = new list<Vehicle__C>();
    list<Product2> equipmentList = new list<Product2>();
    list<Equipment_Maintenance_Item__c> equipmentMaintenanceItemList = new
list<Equipment_Maintenance_Item__c>();
    list<case> caseList = new list<case>();
    list<id> oldCaseIds = new list<id>();
    for(integer i = 0; i < 300; i++){
      vehicleList.add(createVehicle());
      equipmentList.add(createEquipment());
    }
    insert vehicleList;
    insert equipmentList;

    for(integer i = 0; i < 300; i++){
      caseList.add(createMaintenanceRequest(vehicleList.get(i).id,
equipmentList.get(i).id));
    }
    insert caseList;

    for(integer i = 0; i < 300; i++){

equipmentMaintenanceItemList.add(createEquipmentMaintenanceItem(equipmentList.
get(i).id, caseList.get(i).id));
    }
    insert equipmentMaintenanceItemList;

    test.startTest();
    for(case cs : caseList){
      cs.Status = 'Closed';
      oldCaseIds.add(cs.Id);
    }
    update caseList;
    test.stopTest();

    list<case> newCase = [select id
                 from case
                 where status ='New'];
```

```
        list<Equipment_Maintenance_Item__c> workParts = [select id
                                                         from Equipment_Maintenance_Item__c
                                                         where Maintenance_Request__c in: oldCaseIds];

        system.assert(newCase.size() == 300);

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



MaintenanceRequestHelper.apxc
MaintenanceRequest.apxt
WarehouseCalloutService.apxc
Log executeAnonymous @6/23/2022, 6:43:31 PM
WarehouseSyncSchedule.apxc
MaintenanceRequestHelperTest.apxc
Run Test
Go To
Logs
Tests
Checkpoints
Query Editor
View State
Progress
Problems

Code Coverage: None API Version: 45

```
        }
        update caseList;
        test.stopTest();

        list<case> newCase = [select id
                              from case
                              where status ='New'];


        list<Equipment_Maintenance_Item__c> workParts = [select id
                                                         from Equipment_Maintenance_Item__c
                                                         where Maintenance_Request__c in: oldCaseIds];

        system.assert(newCase.size() == 300);

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

Status	Test Run	Enqueued Time	Duration	Failures	Total
✓	7075i00000jjd0F	Thu Jun 23 2022 18:33:37 GM...		0	2
✓	7075i00000jjcw0	Thu Jun 23 2022 18:40:58 GM...		0	4
✓	7075i00000jjcUh	Thu Jun 23 2022 18:43:31 GM...		0	2
✓	7075i00000jjdph	Thu Jun 23 2022 18:53:59 GM...		0	0
✓	7075i00000jjeBF	Thu Jun 23 2022 19:03:37 GM...		0	4
✓	TestRun @ 7:08:45 pm			0	3

Overall Code Coverage

Class	Percent	Lines
Overall	**75%**	
CreateDefaultData	100%	45/45
MaintenanceRequest	100%	2/2
MaintenanceRequestHelper	100%	35/35
WarehouseCalloutService	0%	0/25
WarehouseSyncSchedule	0%	0/2

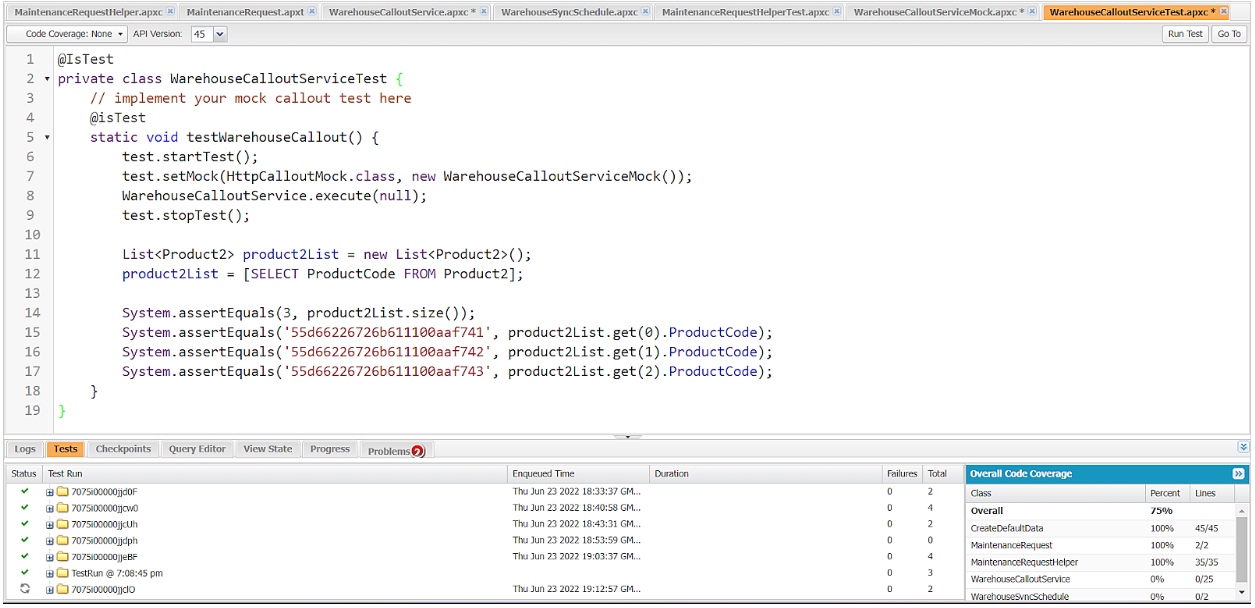
5)Test callout logic

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

    //Write a class that makes a REST callout to an external warehouse system to get a
```

```
list of equipment that needs to be updated.
    //The callout’s JSON response returns the equipment records that you upsert in
Salesforce.

    @future(callout=true)
    public static void runWarehouseEquipmentSync(){
        System.debug('go into runWarehouseEquipmentSync');
        Http http = new Http();
        HttpRequest request = new HttpRequest();

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

        List<Product2> product2List = new List<Product2>();
        System.debug(response.getStatusCode());
        if (response.getStatusCode() == 200){
            List<Object> jsonResponse =
(List<Object>)JSON.deserializeUntyped(response.getBody());
            System.debug(response.getBody());

            //class maps the following fields:
            //warehouse SKU will be external ID for identifying which equipment records to
update within Salesforce
            for (Object jR : jsonResponse){
                Map<String,Object> mapJson = (Map<String,Object>)jR;
                Product2 product2 = new Product2();
                //replacement part (always true),
                product2.Replacement_Part__c = (Boolean) mapJson.get('replacement');
                //cost
                product2.Cost__c = (Integer) mapJson.get('cost');
                //current inventory
                product2.Current_Inventory__c = (Double) mapJson.get('quantity');
                //lifespan
                product2.Lifespan_Months__c = (Integer) mapJson.get('lifespan');
                //maintenance cycle
                product2.Maintenance_Cycle__c = (Integer)
mapJson.get('maintenanceperiod');
                //warehouse SKU
                product2.Warehouse_SKU__c = (String) mapJson.get('sku');

                product2.Name = (String) mapJson.get('name');
                product2.ProductCode = (String) mapJson.get('_id');
                product2List.add(product2);
            }

            if (product2List.size() > 0){
                upsert product2List;
                System.debug('Your equipment was synced with the warehouse one');
            }
        }
    }

    public static void execute (QueueableContext context){
        System.debug('start runWarehouseEquipmentSync');
        runWarehouseEquipmentSync();
        System.debug('end runWarehouseEquipmentSync');
    }

}
@isTest
global class WarehouseCalloutServiceMock implements HttpCalloutMock {
    // implement http mock callout
    global static HttpResponse respond(HttpRequest request) {
```

```
    HttpResponse response = new HttpResponse();
    response.setHeader('Content-Type', 'application/json');

response.setBody('[{"_id":"55d66226726b611100aaf741","replacement":false,"quantity":5
,"name":"Generator 1000
kW","maintenanceperiod":365,"lifespan":120,"cost":5000,"sku":"100003"},{"_id":"55d66226
726b611100aaf742","replacement":true,"quantity":183,"name":"Cooling
Fan","maintenanceperiod":0,"lifespan":0,"cost":300,"sku":"100004"},{"_id":"55d66226726b6
11100aaf743","replacement":true,"quantity":143,"name":"Fuse
20A","maintenanceperiod":0,"lifespan":0,"cost":22,"sku":"100005"}]');
    response.setStatusCode(200);

    return response;
  }
}
@IsTest
private class WarehouseCalloutServiceTest {
  // implement your mock callout test here
      @isTest
  static void testWarehouseCallout() {
    test.startTest();
    test.setMock(HttpCalloutMock.class, new WarehouseCalloutServiceMock());
    WarehouseCalloutService.execute(null);
    test.stopTest();

    List<Product2> product2List = new List<Product2>();
    product2List = [SELECT ProductCode FROM Product2];

    System.assertEquals(3, product2List.size());
    System.assertEquals('55d66226726b611100aaf741',
product2List.get(0).ProductCode);
```

```
        System.assertEquals('55d66226726b611100aaf742',
product2List.get(1).ProductCode);
        System.assertEquals('55d66226726b611100aaf743',
product2List.get(2).ProductCode);
    }
}
```



MaintenanceRequestHelper.apxc
MaintenanceRequest.apxt
WarehouseCalloutService.apxc *
WarehouseSyncSchedule.apxc
MaintenanceRequestHelperTest.apxc
WarehouseCalloutServiceMock.apxc *
WarehouseCalloutServiceTest.apxc *
Code Coverage: None
API Version: 45
Run Test
Go To
Logs
Tests
Checkpoints
Query Editor
View State
Progress
Problems 2

```
@IsTest
private class WarehouseCalloutServiceTest {
    // implement your mock callout test here
    @isTest
    static void testWarehouseCallout() {
        test.startTest();
        test.setMock(HttpCalloutMock.class, new WarehouseCalloutServiceMock());
        WarehouseCalloutService.execute(null);
        test.stopTest();

        List<Product2> product2List = new List<Product2>();
        product2List = [SELECT ProductCode FROM Product2];

        System.assertEquals(3, product2List.size());
        System.assertEquals('55d66226726b611100aaf741', product2List.get(0).ProductCode);
        System.assertEquals('55d66226726b611100aaf742', product2List.get(1).ProductCode);
        System.assertEquals('55d66226726b611100aaf743', product2List.get(2).ProductCode);
    }
}
```

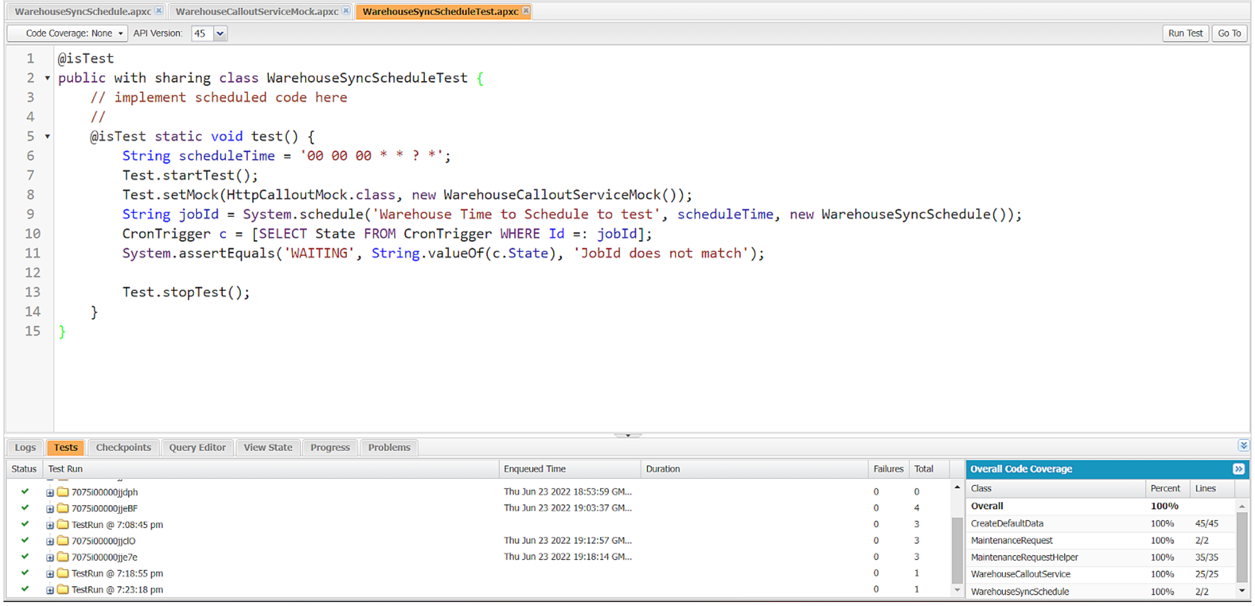
Status	Test Run	Enqueued Time	Duration	Failures	Total
✓	7075i00000jjd0F	Thu Jun 23 2022 18:33:37 GM...		0	2
✓	7075i00000jjcw0	Thu Jun 23 2022 18:40:58 GM...		0	4
✓	7075i00000jjcUh	Thu Jun 23 2022 18:43:31 GM...		0	2
✓	7075i00000jjdph	Thu Jun 23 2022 18:53:59 GM...		0	0
✓	7075i00000jjeBF	Thu Jun 23 2022 19:03:37 GM...		0	4
✓	TestRun @ 7:08:45 pm			0	3
⟳	7075i00000jjclO	Thu Jun 23 2022 19:12:57 GM...		0	2

Overall Code Coverage

Class	Percent	Lines
Overall	**75%**	
CreateDefaultData	100%	45/45
MaintenanceRequest	100%	2/2
MaintenanceRequestHelper	100%	35/35
WarehouseCalloutService	0%	0/25
WarehouseSyncSchedule	0%	0/2

6)Test scheduling logic

```
global with sharing class WarehouseSyncSchedule implements Schedulable {
    // implement scheduled code here
    global void execute (SchedulableContext ctx){
        System.enqueueJob(new WarehouseCalloutService());
    }
}
@isTest
global class WarehouseCalloutServiceMock implements HttpCalloutMock {
    // implement http mock callout
    global static HttpResponse respond(HttpRequest request) {


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

response.setBody('[{"_id":"55d66226726b611100aaf741","replacement":false,"quantity":5
,"name":"Generator 1000
kW","maintenanceperiod":365,"lifespan":120,"cost":5000,"sku":"100003"},{"_id":"55d66226
726b611100aaf742","replacement":true,"quantity":183,"name":"Cooling
Fan","maintenanceperiod":0,"lifespan":0,"cost":300,"sku":"100004"},{"_id":"55d66226726b6
11100aaf743","replacement":true,"quantity":143,"name":"Fuse
20A","maintenanceperiod":0,"lifespan":0,"cost":22,"sku":"100005"}]');
        response.setStatusCode(200);

        return response;
    }
}
@isTest
public with sharing class WarehouseSyncScheduleTest {
    // implement scheduled code here
    //
    @isTest static void test() {
        String scheduleTime = '00 00 00 * * ? *';
        Test.startTest();
        Test.setMock(HttpCalloutMock.class, new WarehouseCalloutServiceMock());
        String jobId = System.schedule('Warehouse Time to Schedule to test',
scheduleTime, new WarehouseSyncSchedule());
        CronTrigger c = [SELECT State FROM CronTrigger WHERE Id =: jobId];
        System.assertEquals('WAITING', String.valueOf(c.State), 'JobId does not match');

        Test.stopTest();
    }
}
```


WarehouseSyncSchedule.apxc
WarehouseCalloutServiceMock.apxc
WarehouseSyncScheduleTest.apxc
Run Test
Go To
Logs
Tests
Checkpoints
Query Editor
View State
Progress
Problems

Code Coverage: None
API Version: 45

```
@isTest
public with sharing class WarehouseSyncScheduleTest {
    // implement scheduled code here
    //
    @isTest static void test() {
        String scheduleTime = '00 00 00 * * ? *';
        Test.startTest();
        Test.setMock(HttpCalloutMock.class, new WarehouseCalloutServiceMock());
        String jobId = System.schedule('Warehouse Time to Schedule to test', scheduleTime, new WarehouseSyncSchedule());
        CronTrigger c = [SELECT State FROM CronTrigger WHERE Id =: jobId];
        System.assertEquals('WAITING', String.valueOf(c.State), 'JobId does not match');

        Test.stopTest();
    }
}
```

Status	Test Run	Enqueued Time	Duration	Failures	Total
✓	7075i00000jjdph	Thu Jun 23 2022 18:53:59 GM...		0	0
✓	7075i00000jjeBF	Thu Jun 23 2022 19:03:37 GM...		0	4
✓	TestRun @ 7:08:45 pm			0	3
✓	7075i00000jjclO	Thu Jun 23 2022 19:12:57 GM...		0	3
✓	7075i00000jje7e	Thu Jun 23 2022 19:18:14 GM...		0	3
✓	TestRun @ 7:18:55 pm			0	1
✓	TestRun @ 7:23:18 pm			0	1

Overall Code Coverage

Class	Percent	Lines
Overall	**100%**	
CreateDefaultData	100%	45/45
MaintenanceRequest	100%	2/2
MaintenanceRequestHelper	100%	35/35
WarehouseCalloutService	100%	25/25
WarehouseSyncSchedule	100%	2/2

Process Automation specialists-superbadge

1)Automate leads

(i)To check whether the given country is either in terms of US or abbreviated forms of US:

OR(

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:MN:MS:MO:MT:NE:NV:NH:NJ:NM:NY:NC:ND:OH:OK:OR:OA:PA:RI:SC:SD:TN:TX:UT:VT:VA:WA:WV:WI:WY", State)),

LEN(State) <> 2,

NOT(OR(Country="US",Country="USA",Country="United States",

ISBLANK(Country)))

)

2)Automate accounts

(i)To calculate deal win percentage:

IF(DATE(YEAR(Last_won_deal_date__c) * 2

,MONTH(Last_won_deal_date__c),DAY(Last_won_deal_date__c)) <=TODAY(),"YES","NO")

(ii)To check whether the given Billing country and Shipping country is either in terms of US or abbreviated forms of US:

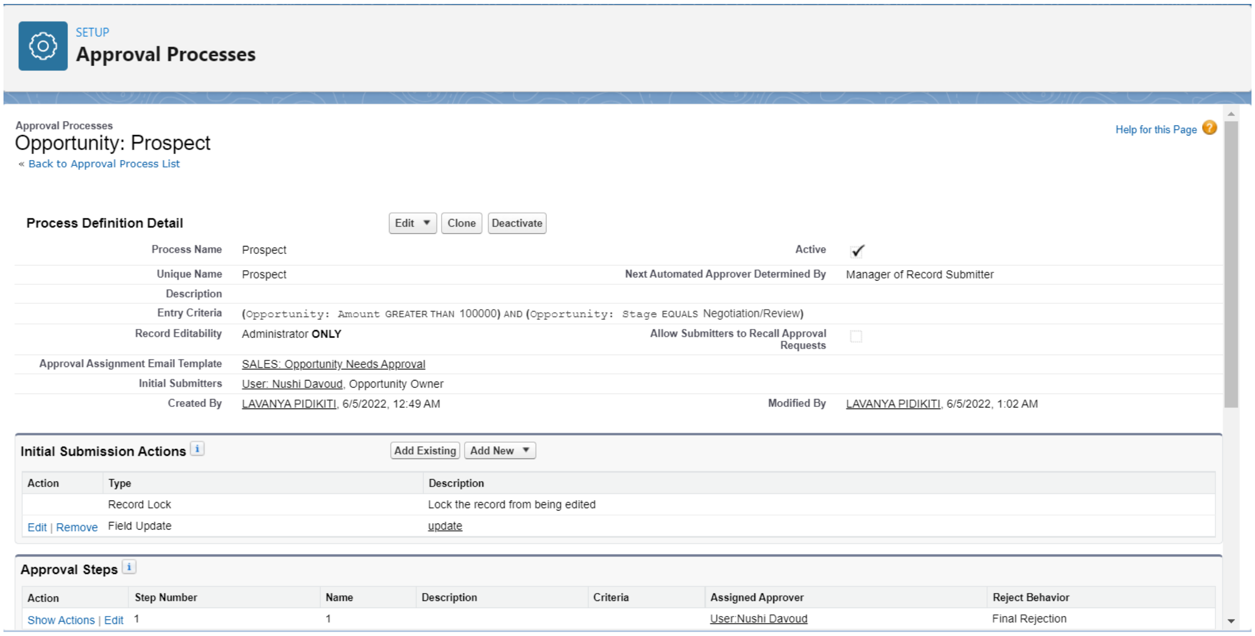
OR(

```
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:MN:MS:MO:MT:NE:NV:NH:NJ:NM:NY:NC:ND:OH:OK:OR:OA:PA:RI:SC:SD:TN:TX:UT:VT:VA
:WA:WV:WI:WY", BillingState)),
LEN(BillingState) <> 2,
NOT(OR(BillingCountry="US",BillingCountry="USA",BillingCountry="United States",
ISBLANK(BillingCountry))),
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:MN:MS:MO:MT:NE:NV:NH:NJ:NM:NY:NC:ND:OH:OK:OR:OA:PA:RI:SC:SD:TN:TX:UT:VT:VA
:WA:WV:WI:WY", ShippingState)),
LEN(ShippingState) <> 2,
NOT(OR(ShippingCountry="US",ShippingCountry="USA",ShippingCountry="United States",
ISBLANK(ShippingCountry)))
)
```

(iii)To check the customer type:

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

6)



Approval Processes

Opportunity: Prospect

« Back to Approval Process List

Process Definition Detail

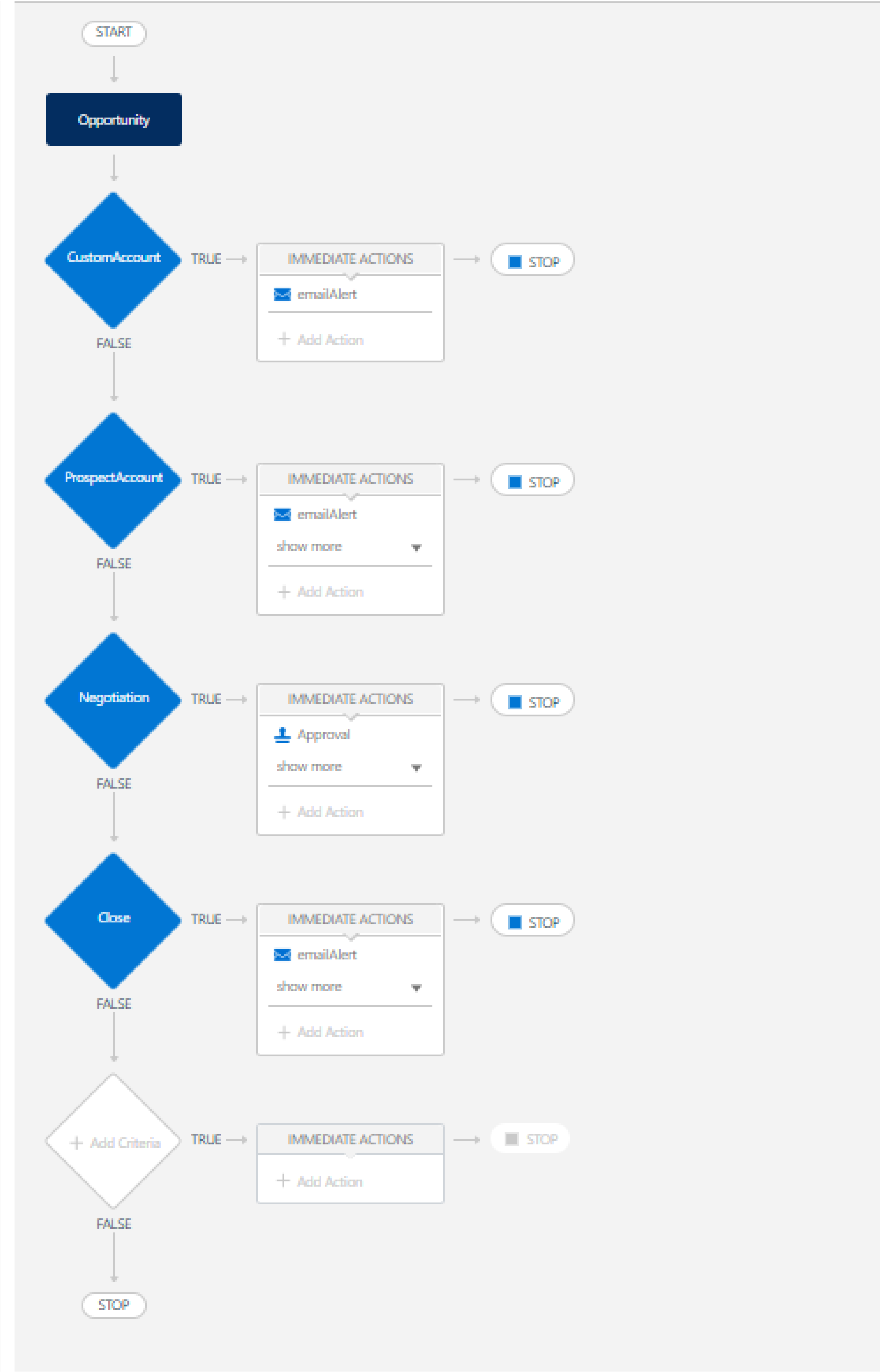
Process Name	Prospect	Active	✓
Unique Name	Prospect	Next Automated Approver Determined By	Manager of Record Submitter
Description			
Entry Criteria	(Opportunity: Amount GREATER THAN 100000) AND (Opportunity: Stage EQUALS Negotiation/Review)		
Record Editability	Administrator **ONLY**	Allow Submitters to Recall Approval Requests	□
Approval Assignment Email Template	SALES: Opportunity Needs Approval		
Initial Submitters	User: Nushi Davoud, Opportunity Owner		
Created By	LAVANYA PIDIKITI, 6/5/2022, 12:49 AM	Modified By	LAVANYA PIDIKITI, 6/5/2022, 1:02 AM

Initial Submission Actions

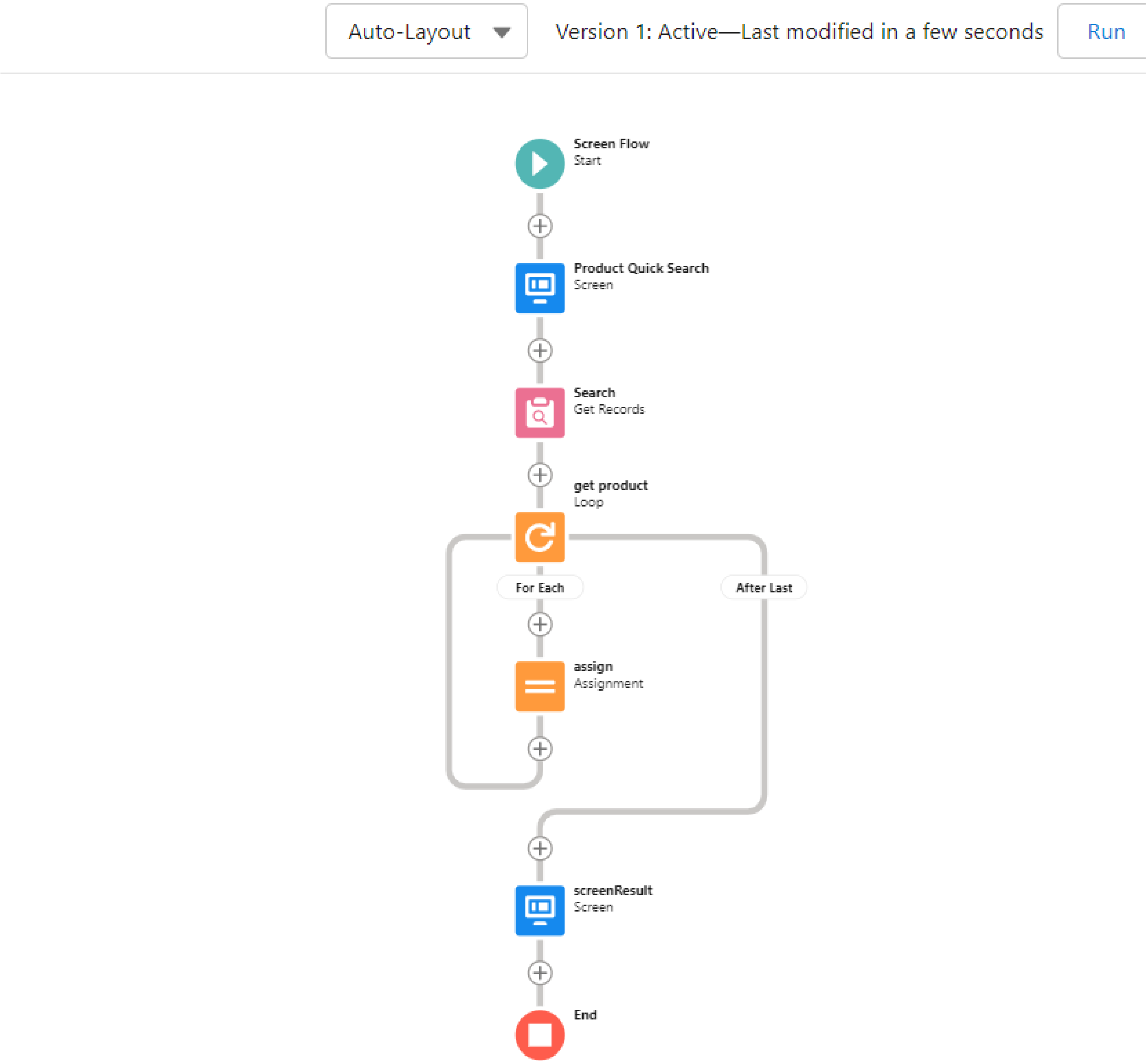
Action	Type	Description
	Record Lock	Lock the record from being edited
Edit | Remove	Field Update	update

Approval Steps

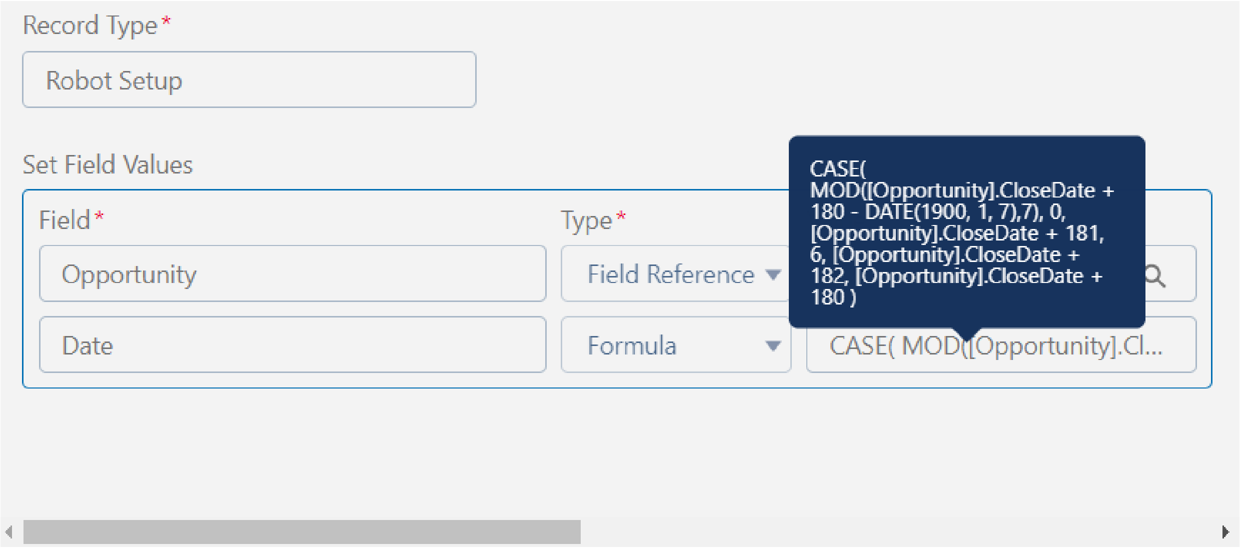
Action	Step Number	Name	Description	Criteria	Assigned Approver	Reject Behavior
Show Actions | Edit	1	1			User:Nushi Davoud	Final Rejection


START
Opportunity
CustomAccount
TRUE
IMMEDIATE ACTIONS
emailAlert
Add Action
STOP
FALSE
ProspectAccount
TRUE
IMMEDIATE ACTIONS
emailAlert
show more
Add Action
STOP
FALSE
Negotiation
TRUE
IMMEDIATE ACTIONS
Approval
show more
Add Action
STOP
FALSE
Close
TRUE
IMMEDIATE ACTIONS
emailAlert
show more
Add Action
STOP
FALSE
Add Criteria
TRUE
IMMEDIATE ACTIONS
Add Action
STOP
FALSE
STOP

7)


Auto-Layout
Version 1: Active—Last modified in a few seconds
Run
Screen Flow
Start
Product Quick Search
Screen
Search
Get Records
get product
Loop
For Each
After Last
assign
Assignment
screenResult
Screen
End

8)



Record Type*

Robot Setup

Set Field Values

Field*	Type*	
Opportunity	Field Reference	
Date	Formula	CASE(MOD([Opportunity].Cl...

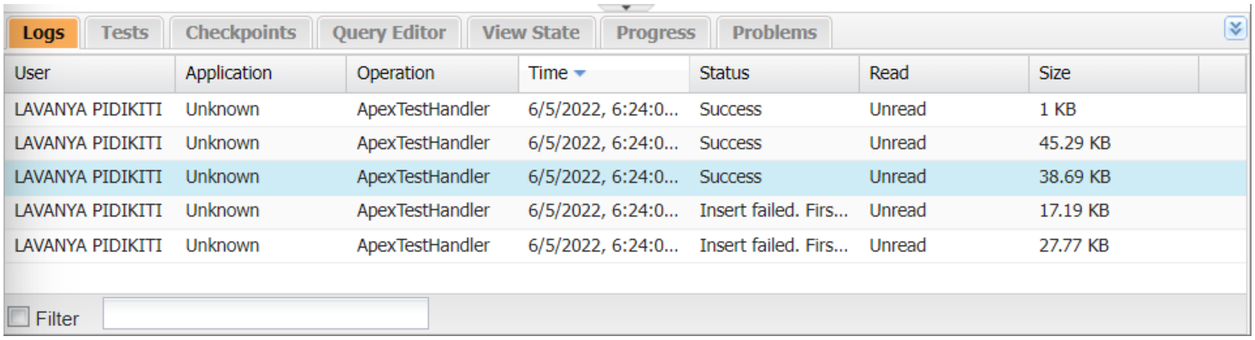
CASE(MOD([Opportunity].CloseDate + 180 - DATE(1900, 1, 7),7), 0, [Opportunity].CloseDate + 181, 6, [Opportunity].CloseDate + 182, [Opportunity].CloseDate + 180)

Self-learning modules

1)Create an Apex trigger

Code:

```
trigger AccountAddressTrigger on Account (before insert, before update) {
   for(Account account:Trigger.New){
      if(account.Match_Billing_Address__c == True){
         account.ShippingPostalCode = account.BillingPostalCode;
      }
   }
}
```



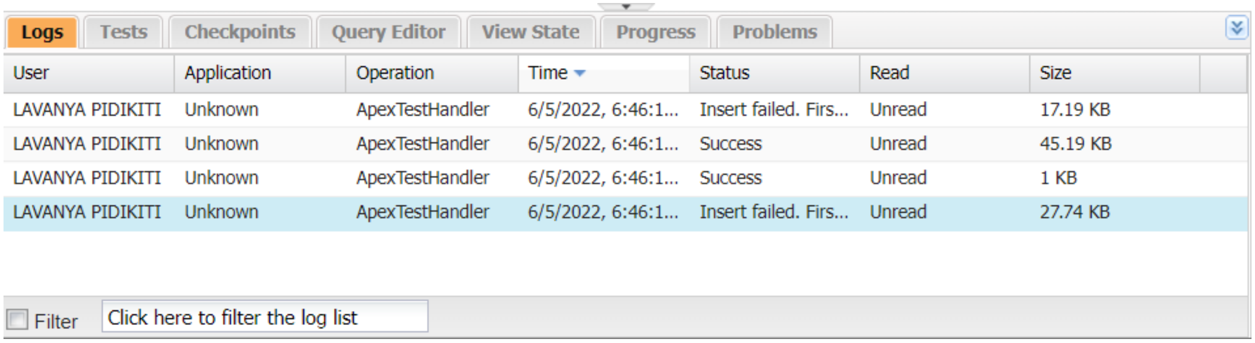
User	Application	Operation	Time	Status	Read	Size
LAVANYA PIDIKITI	Unknown	ApexTestHandler	6/5/2022, 6:24:0...	Success	Unread	1 KB
LAVANYA PIDIKITI	Unknown	ApexTestHandler	6/5/2022, 6:24:0...	Success	Unread	45.29 KB
LAVANYA PIDIKITI	Unknown	ApexTestHandler	6/5/2022, 6:24:0...	Success	Unread	38.69 KB
LAVANYA PIDIKITI	Unknown	ApexTestHandler	6/5/2022, 6:24:0...	Insert failed. Firs...	Unread	17.19 KB
LAVANYA PIDIKITI	Unknown	ApexTestHandler	6/5/2022, 6:24:0...	Insert failed. Firs...	Unread	27.77 KB

2)Bulk Apex Trigger

Code:

```
trigger ClosedOpportunityTrigger on Opportunity (after insert, after update) {
       List<Task> tasklist = new List<Task>();
```

```
    for(Opportunity opp:Trigger.New){
        if(opp.StageName == 'Closed Won'){
            tasklist.add(new Task(Subject = 'Follow Up Test Task', WhatId=opp.Id));
        }
    }
    if(tasklist.size()>0){
        insert tasklist;
    }
}
```



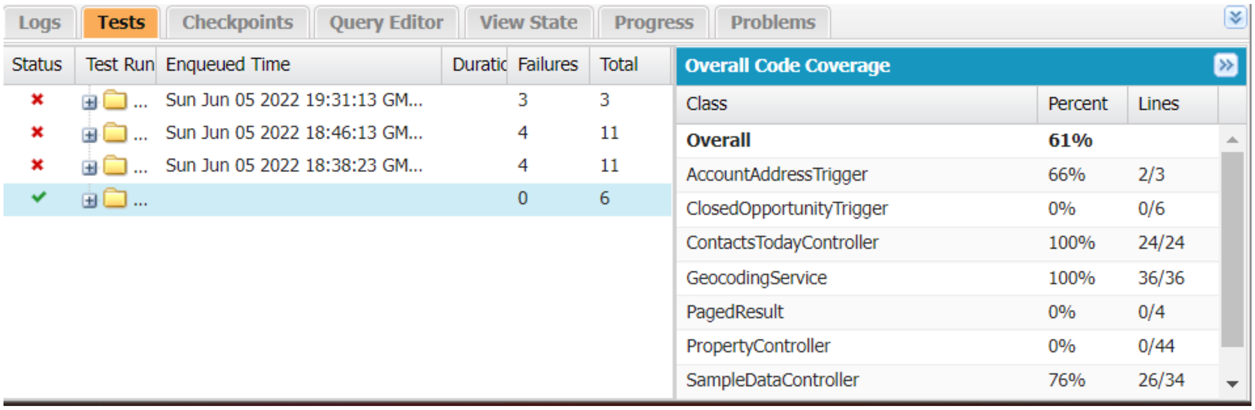
User	Application	Operation	Time	Status	Read	Size
LAVANYA PIDIKITI	Unknown	ApexTestHandler	6/5/2022, 6:46:1...	Insert failed. Firs...	Unread	17.19 KB
LAVANYA PIDIKITI	Unknown	ApexTestHandler	6/5/2022, 6:46:1...	Success	Unread	45.19 KB
LAVANYA PIDIKITI	Unknown	ApexTestHandler	6/5/2022, 6:46:1...	Success	Unread	1 KB
LAVANYA PIDIKITI	Unknown	ApexTestHandler	6/5/2022, 6:46:1...	Insert failed. Firs...	Unread	27.74 KB

3)Create a Unit Test for a Simple Apex Class

Code:

```
@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 static void 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('02/02/2019'));
    System.assertEquals(false,flag);
  }

   @isTest static void Test_DateWithin30Days_case3(){
    Boolean flag =
VerifyDate.DateWithin30Days(date.parse('01/01/2020'),date.parse('12/30/2020'));
    System.assertEquals(false,flag);
  }

  @isTest static void Test_SetEndOfMonthDate(){
    Date returndate = VerifyDate.SetEndOfMonthDate(date.parse('01/01/2020'));
  }
}
```



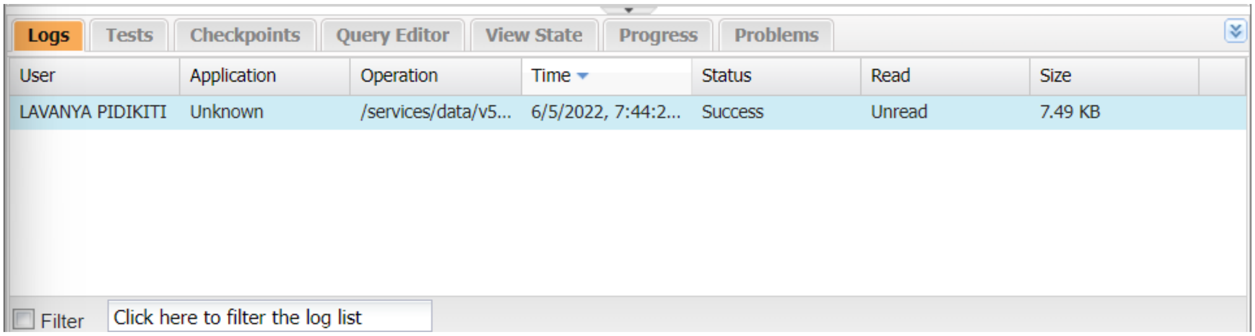
Status	Test Run	Enqueued Time	Duratic	Failures	Total
✗	...	Sun Jun 05 2022 19:31:13 GM...		3	3
✗	...	Sun Jun 05 2022 18:46:13 GM...		4	11
✗	...	Sun Jun 05 2022 18:38:23 GM...		4	11
✓	...			0	6

Overall Code Coverage

Class	Percent	Lines
Overall	**61%**	
AccountAddressTrigger	66%	2/3
ClosedOpportunityTrigger	0%	0/6
ContactsTodayController	100%	24/24
GeocodingService	100%	36/36
PagedResult	0%	0/4
PropertyController	0%	0/44
SampleDataController	76%	26/34

4)Create a Unit Test for a Simple Apex Trigger

```
@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());
      System.assert(result.getErrors().size() > 0);
      System.assertEquals('The Last Name "INVALIDNAME" is not allowed for
DML',result.getErrors()[0].getMessage());

   }
}
```

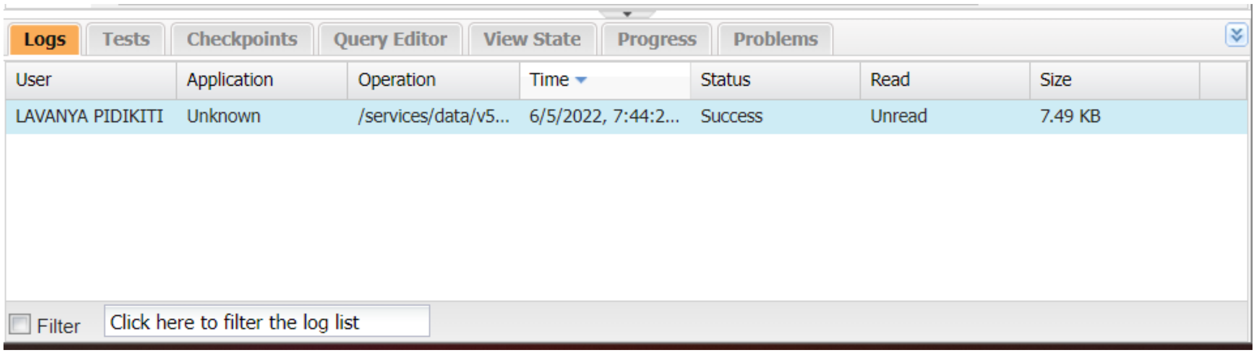


User	Application	Operation	Time	Status	Read	Size
LAVANYA PIDIKITI	Unknown	/services/data/v5...	6/5/2022, 7:44:2...	Success	Unread	7.49 KB

5)Create a Contact Test Factory

Code:

```
public class RandomContactFactory {
   public static List<Contact> generateRandomContacts(Integer numcnt , string lastname){
      List<Contact> contacts = new List<Contact>();
      for(Integer i=0;i<numcnt;i++){
         Contact cnt = new Contact(FirstName = 'Test '+i,LastName=lastname);
      }
      return contacts;
   }
}
```

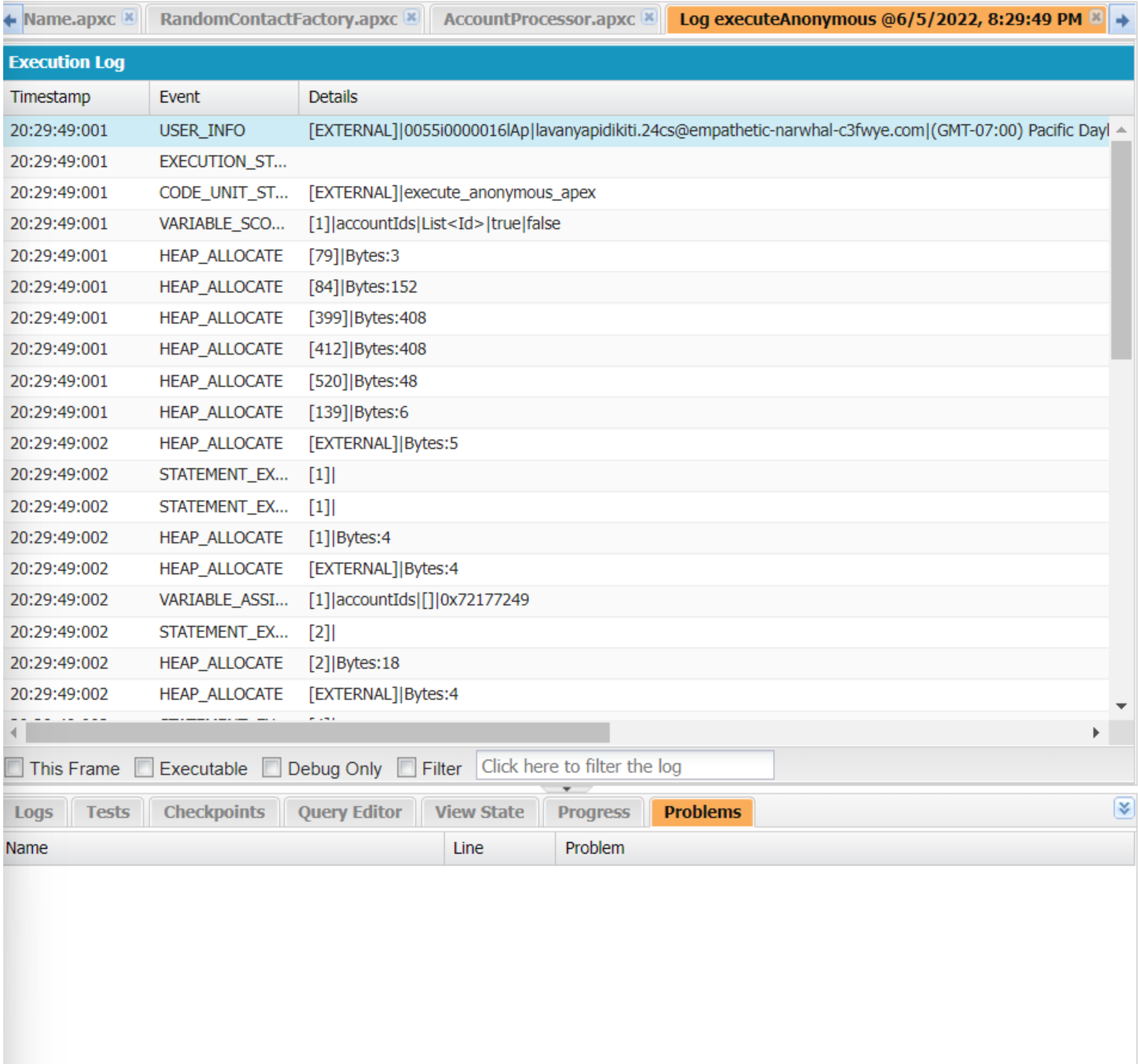


User	Application	Operation	Time	Status	Read	Size
LAVANYA PIDIKITI	Unknown	/services/data/v5...	6/5/2022, 7:44:2...	Success	Unread	7.49 KB

6)Use future method

Code:

```
public class AccountProcessor {
   @future
   public static void countContacts(List<Id> accountIds){
        List<Account> accountsToUpdate = new List<Account>();

        List<Account> accounts = [Select Id, Name , (Select Id 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;

    }

@IsTest
public class AccountProcessorTest {
     @IsTest
    private static void testCountContact(){
        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='Jane',Lastname='Doe',AccountId=
newAccount.id);
        insert newContact2;

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


Name.apxc
RandomContactFactory.apxc
AccountProcessor.apxc
Log executeAnonymous @6/5/2022, 8:29:49 PM
Execution Log
□ This Frame □ Executable □ Debug Only □ Filter
Click here to filter the log
Logs
Tests
Checkpoints
Query Editor
View State
Progress
Problems

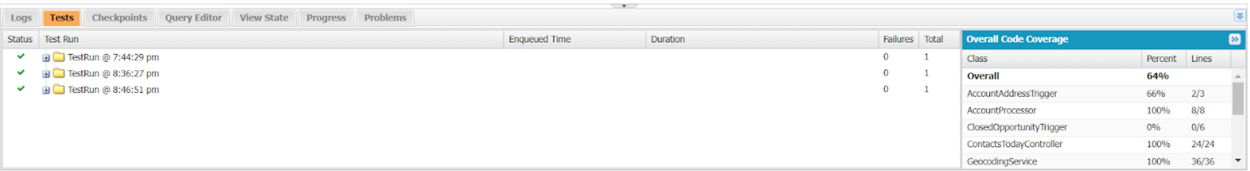
Timestamp	Event	Details
20:29:49:001	USER_INFO	[EXTERNAL]|0055i0000016lAp|lavanyapidikiti.24cs@empathetic-narwhal-c3fwye.com|(GMT-07:00) Pacific Dayl
20:29:49:001	EXECUTION_ST...	
20:29:49:001	CODE_UNIT_ST...	[EXTERNAL]|execute_anonymous_apex
20:29:49:001	VARIABLE_SCO...	[1]|accountIds|List<Id>|true|false
20:29:49:001	HEAP_ALLOCATE	[79]|Bytes:3
20:29:49:001	HEAP_ALLOCATE	[84]|Bytes:152
20:29:49:001	HEAP_ALLOCATE	[399]|Bytes:408
20:29:49:001	HEAP_ALLOCATE	[412]|Bytes:408
20:29:49:001	HEAP_ALLOCATE	[520]|Bytes:48
20:29:49:001	HEAP_ALLOCATE	[139]|Bytes:6
20:29:49:002	HEAP_ALLOCATE	[EXTERNAL]|Bytes:5
20:29:49:002	STATEMENT_EX...	[1]|
20:29:49:002	STATEMENT_EX...	[1]|
20:29:49:002	HEAP_ALLOCATE	[1]|Bytes:4
20:29:49:002	HEAP_ALLOCATE	[EXTERNAL]|Bytes:4
20:29:49:002	VARIABLE_ASSI...	[1]|accountIds|[]|0x72177249
20:29:49:002	STATEMENT_EX...	[2]|
20:29:49:002	HEAP_ALLOCATE	[2]|Bytes:18
20:29:49:002	HEAP_ALLOCATE	[EXTERNAL]|Bytes:4

Name	Line	Problem

7)Use Batch Apex

```
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 = 'Dreamforce';
            L_list_new.add(L);
            count += 1;
        }
        update L_list_new;
    }
    global void finish(Database.BatchableContext bc){
        System.debug('count = '+count);
    }
}

@isTest

public class LeadProcessorTest {

    @isTest
    public static void 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();
    }
```



Status	Test Run	Enqueued Time	Duration	Failures	Total
✓	⊞ TestRun @ 7:44:29 pm			0	1
✓	⊞ TestRun @ 8:36:27 pm			0	1
✓	⊞ TestRun @ 8:46:51 pm			0	1

Overall Code Coverage

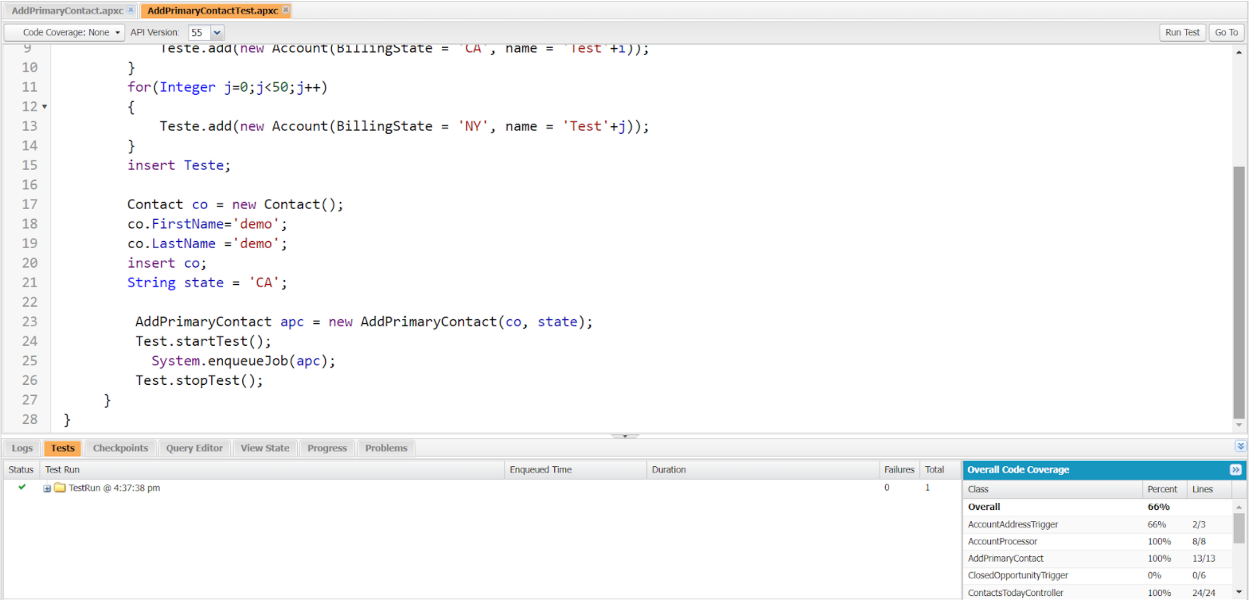
Class	Percent	Lines
Overall	**64%**	
AccountAddressTrigger	66%	2/3
AccountProcessor	100%	8/8
ClosedOpportunityTrigger	0%	0/6
ContactsTodayController	100%	24/24
GeocodingService	100%	36/36

8)Control Processes with Queueable Apex

Code:

```
public class AddPrimaryContact implements Queueable{

    private Contact con;
    private String state;



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

    }

    public void execute(QueueableContext context){
        List<Account> accounts = [Select Id, Name, (Select FirstName, LastName, Id from
contacts)
                from Account where BillingState = :state Limit 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;
    }
  }
}

@isTest
public class AddPrimaryContactTest
{
   @isTest static void 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();
     }
}
```



```
                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();
        }
    }
```

Status	Test Run	Enqueued Time	Duration	Failures	Total
✓	TestRun @ 4:37:38 pm			0	1

Overall Code Coverage

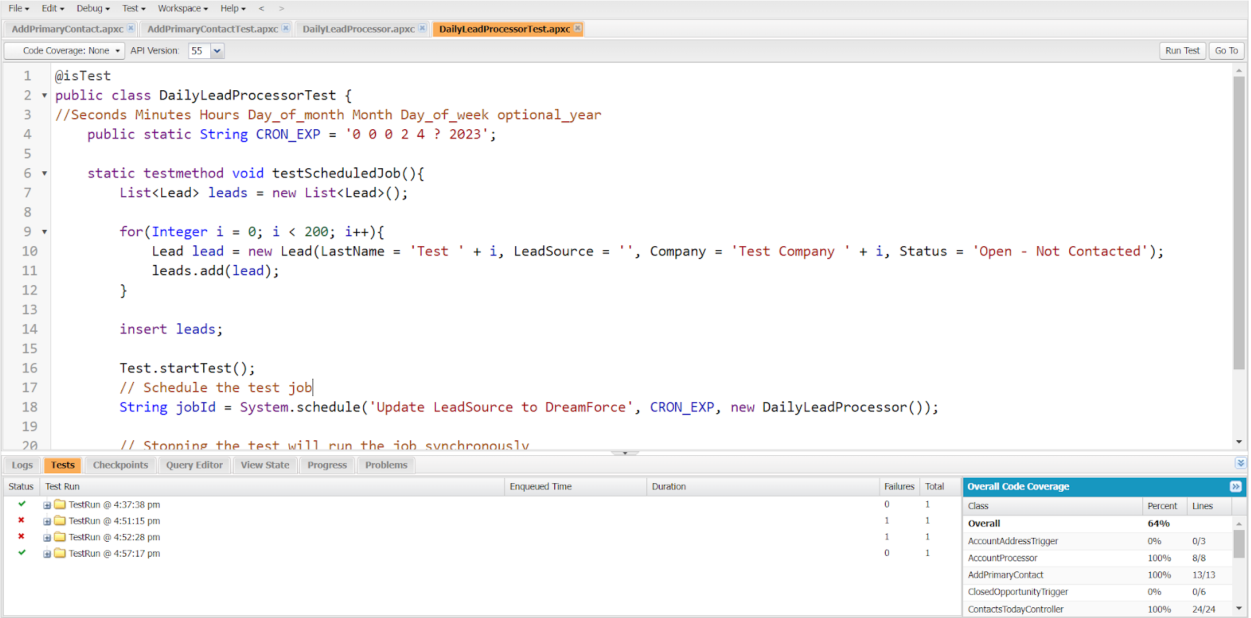
Class	Percent	Lines
Overall	**66%**	
AccountAddressTrigger	66%	2/3
AccountProcessor	100%	8/8
AddPrimaryContact	100%	13/13
ClosedOpportunityTrigger	0%	0/6
ContactsTodayController	100%	24/24

9)Schedule Jobs Using the Apex Scheduler

Code:

```
global class DailyLeadProcessor implements Schedulable {
 global void execute(SchedulableContext ctx) {
        List<Lead> lList = [Select Id, LeadSource from Lead where LeadSource = null];
        if(!lList.isEmpty()) {
   for(Lead l: lList) {
    l.LeadSource = 'Dreamforce';
   }
   update lList;
  }
    }

}



@isTest
public class DailyLeadProcessorTest {
//Seconds Minutes Hours Day_of_month Month Day_of_week optional_year
    public static String CRON_EXP = '0 0 0 2 4 ? 2023';

    static testmethod void testScheduledJob(){
        List<Lead> leads = new List<Lead>();

        for(Integer i = 0; i < 200; i++){
            Lead lead = new Lead(LastName = 'Test ' + i, LeadSource = '', Company = 'Test
Company ' + i, Status = 'Open - Not Contacted');
            leads.add(lead);
        }

        insert leads;

        Test.startTest();
        // Schedule the test job
        String jobId = System.schedule('Update LeadSource to DreamForce', CRON_EXP,
new DailyLeadProcessor());
```

```
        // Stopping the test will run the job synchronously
        Test.stopTest();
    }

}
```



```
@isTest
public class DailyLeadProcessorTest {
//Seconds Minutes Hours Day_of_month Month Day_of_week optional_year
    public static String CRON_EXP = '0 0 0 2 4 ? 2023';

    static testmethod void testScheduledJob(){
        List<Lead> leads = new List<Lead>();

        for(Integer i = 0; i < 200; i++){
            Lead lead = new Lead(LastName = 'Test ' + i, LeadSource = '', Company = 'Test Company ' + i, Status = 'Open - Not Contacted');
            leads.add(lead);
        }

        insert leads;

        Test.startTest();
        // Schedule the test job
        String jobId = System.schedule('Update LeadSource to DreamForce', CRON_EXP, new DailyLeadProcessor());

        // Stopping the test will run the job synchronously
```

Status	Test Run	Enqueued Time	Duration	Failures	Total
✓	TestRun @ 4:37:38 pm			0	1
✗	TestRun @ 4:51:15 pm			1	1
✗	TestRun @ 4:52:28 pm			1	1
✓	TestRun @ 4:57:17 pm			0	1

Overall Code Coverage

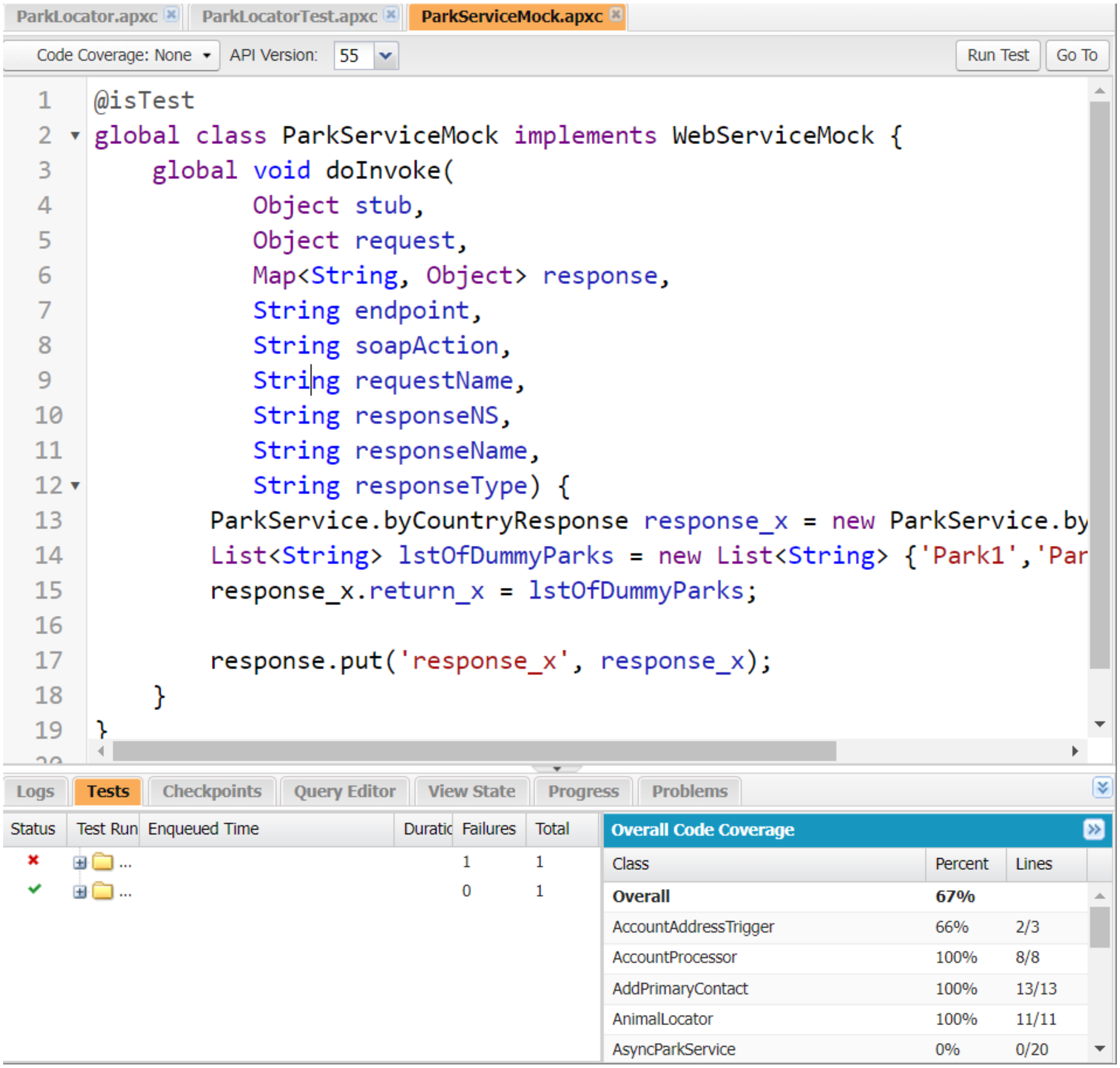
Class	Percent	Lines
Overall	**64%**	
AccountAddressTrigger	0%	0/3
AccountProcessor	100%	8/8
AddPrimaryContact	100%	13/13
ClosedOpportunityTrigger	0%	0/6
ContactsTodayController	100%	24/24

10)Apex SOAP call out

Code:

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

@isTest
private class ParkLocatorTest{
    @isTest
    static void testParkLocator() {
```

```
    Test.setMock(WebServiceMock.class, new ParkServiceMock());
    String[] arrayOfParks = ParkLocator.country('India');

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

@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) {
    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);
  }
}
```



Code Coverage: None

API Version: 55

```
@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) {
        ParkService.byCountryResponse response_x = new ParkService.by
        List<String> lstOfDummyParks = new List<String> {'Park1','Par
        response_x.return_x = lstOfDummyParks;

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

Status	Test Run	Enqueued Time	Duratic	Failures	Total
✗	...			1	1
✓	...			0	1

Overall Code Coverage

Class	Percent	Lines
Overall	**67%**	
AccountAddressTrigger	66%	2/3
AccountProcessor	100%	8/8
AddPrimaryContact	100%	13/13
AnimalLocator	100%	11/11
AsyncParkService	0%	0/20