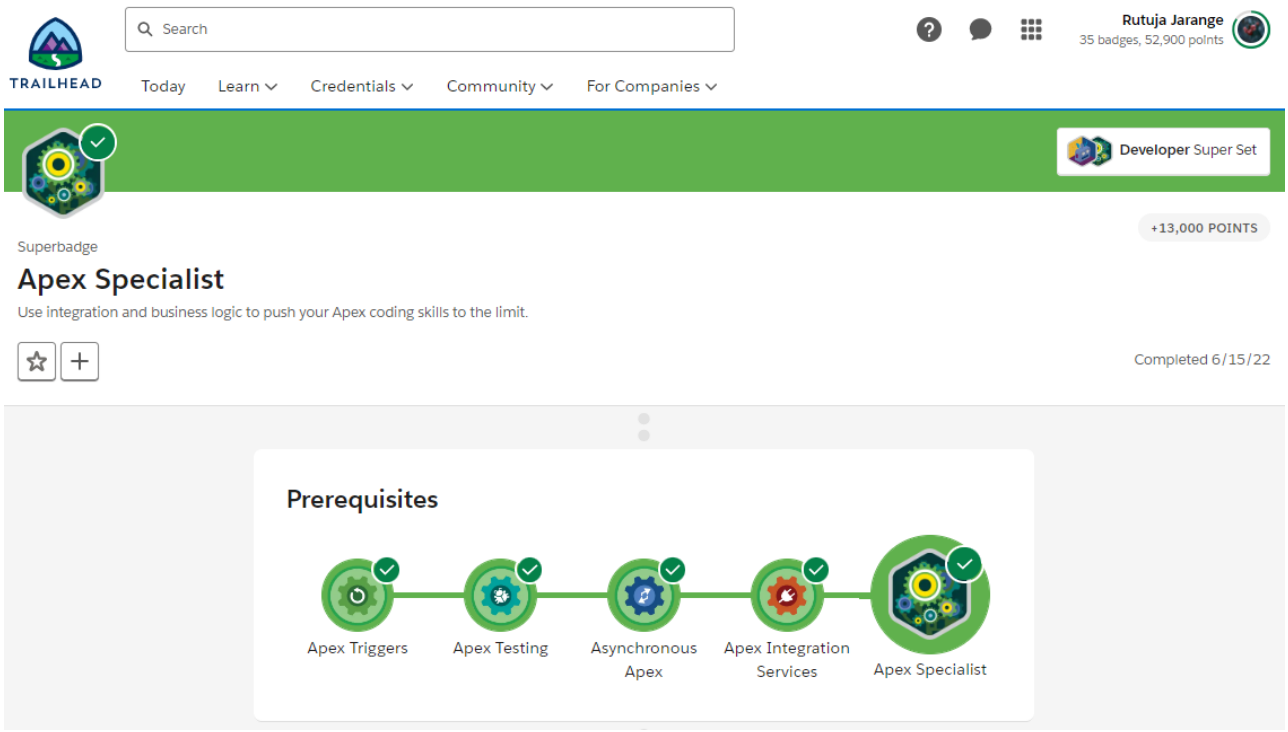
Name - Rutuja Jarange
Project Title : Salesforce Developer Catalyst
About : All the details about Projects, Badges Completion and Codes.

Apex Specialist Super badge



APEX TRIGGERS

1. Get Started with Apex Triggers

Creation of an Apex trigger :
Given code is of an apex trigger named as `"AccountAddressTrigger"`

```
trigger AccountAddressTrigger on Account (before insert, before update) {
    for(Account a : Trigger.New) {
```

```
        if(a.Match_Billing_Address__c ==True){
            a.ShippingPostalCode = a.BillingPostalCode;
        }
    }
}
```

2. Bulk Apex Triggers

Given is a bulkified Apex trigger that adds a follow-up task to an opportunity if its stage is Closed Won. Fire the Apex trigger after inserting or updating an opportunity.

Apex trigger : `ClosedOpportunityTrigger`

```
trigger ClosedOpportunityTrigger on Opportunity (after insert, after update) {
  list <Task> Tl= new list<Task>();
    for ( Opportunity o : Trigger.New){
        if(o.StageName=='Closed Won'){
            Tl.add(new Task(Subject = 'Follow Up Testd = o.ID));
        }
    }
    if(Tl.size()>0){
        insert Tl;
    }
}
```

Apex Testing

1.Get Started with Apex Unit Tests

Create and install a simple Apex class to test if a date is within a proper range, and if not, returns a date that occurs at the end of the month within the range.

Apex class : `VerifyDate`

```
public class VerifyDate {

  //method to handle potential checks against two dates
  public static Date CheckDates(Date date1, Date date2) {
        //if date2 is within the next 30 days of date1, use date2.  Otherwise use the 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 private static Boolean DateWithin30Days(Date date1, Date date2) {
        //check for date2 being in the past
  if( date2 < date1) { return false; }

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

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

}
```

Unit Test : `TestVerifyDate`

```
@isTest
Private class TestVerifyDate {
    @isTest static void Test_CheckDates_case1(){
        Date D = VerifyDate.CheckDates(Date.parse('01/01/2022'), Date.parse('01/05/2022'));
        System.assertEquals( Date.parse('01/05/2022'),D);

    }
    @isTest static void Test_CheckDates_case2(){
        Date D = VerifyDate.CheckDates(Date.parse('01/01/2022'), Date.parse('05/05/2022'));
        System.assertEquals( Date.parse('01/31/2022'),D);
}
    @isTest static void Test_DateWithin30Days_case1(){
        Boolean flag = VerifyDate.DateWithin30Days(date.parse('01/01/2022'), date.parse('12/30/21'));
        System.assertEquals(false,flag);
   }
    @isTest static void Test_DateWithin30Days_case2(){
        Boolean flag = VerifyDate.DateWithin30Days(date.parse('01/01/2022'), date.parse('02/02/21'));
        System.assertEquals(false,flag);
    }
    @isTest static void Test_DateWithin30Days_case3(){
        Boolean flag = VerifyDate.DateWithin30Days(date.parse('01/01/2022'), date.parse('1/15/22'));
        System.assertEquals(true,flag);
    }
    @isTest static Void Test_SetEndOfMonthDate(){
        Date returndate = VerifyDate.SetEndOfMonthDate(date.parse('01/01/2022'));
    }
}
```

2. Test Apex Triggers:

Create and install a simple Apex trigger which blocks inserts and updates to any contact with a last name of 'INVALIDNAME'.

Apex trigger : `RestrictContactByName`

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

 //check contacts prior to insert or update for invalid data
 For (Contact c : Trigger.New) {
  if(c.LastName == 'INVALIDNAME') { //invalidname is invalid
  c.AddError('The Last Name "'+c.LastName+'" is not

  }
 }
}
```

Unit Tests : `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());
         System.assert(result.getErrors().size()> 0);
         System.assertEquals('The Last Name "INVALIDNAME"
```

```
    }
}
```

3. Create Test Data for Apex Tests:

Create an Apex class that returns a list of contacts based on two incoming parameters

Apex class : `RandomContactFactory`

```
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);
            contacts.add(cnt);
        }
        return contacts;
        }
    }
```

Asynchronous Apex -

1. Use Future Methods -

Create an Apex class with a future method that accepts a List of Account IDs and updates a custom field on the Account object with the number of contacts associated to the Account.

Apex class : `AccountProcessor`

```
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;

    }
}
```

Apex test class : `AccountProcessorTest`

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

        insert newAccount;
        Contact newContact1= new Contact(FirstNAme='John',LastName='Doe',AccountId = newAccount.Id);
        insert newContact1;
        Contact newContact2=new Contact(FirstName='J',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

Create an Apex class that implements the Database.Batchable interface to update all Lead records in the org with a specific LeadSource.

Apex class : `LeadProcessor`

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

    global Database.QueryLocator start(Database.BatchableContext bc){
      return Database.getQueryLocator('SELECT ID ,

    }
    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);

    }

}
```

Apex test class : `LeadProcessorTest`

```
@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 = 'Comapany';
            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. Control Processes with Queueable Apex

Apex class : AddPrimaryContact

```
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;
         }
     }

}
```

Apex test class : `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 where accountId in(Select Id from Account where BillingState = 'CA')]);
```

```
    }
}
```

4. Schedule Jobs Using the Apex Scheduler -

Apex class : `DailyLeadProcessor`

```
public without sharing class DailyLeadProcessor implements Schedulable{
    public void execute (SchedulableContext ctx){
        List <Lead> leads =[SELECT Id, LeadSource FROM Lead WHERE LeadSource = null LIMIT 200];
        for (Lead l:leads){
            l.LeadSource = 'Dreamforce';
        }
        update leads;
    }
}
```

Apex test class : `DailyLeadProcessorTest`

```
@isTest
private class DailyLeadProcessorTest{
    //Seconds Minutes Hours Day_of_month Month Day_of_week optional_year
    public static String CRON_EXP = '0 0 0 2 6 ? 2022';

    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

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

Apex Integration Services -

1. Apex REST Callouts

Create an Apex class that calls a REST endpoint to return the name of an animal.

Apex class : `AnimalLocator`

```
public class AnimalLocator{
    public static String getAnimalNameById(Integer x){
        Http http = new Http();
        HttpRequest req = new HttpRequest();

        req.setEndpoint('https://th-apex-http-

        req.setMethod('GET');

        HttpResponse res = http.send(req);
        Map<String, Object> results = (Map<String, Object>)JSON.deserializeUntyped(res.getBody());
        Map<String, Object> animal = (Map<String, Object>) results.get('animal');
        return (String)animal.get('name');
    }
```

```
}
```

Test Class : `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);
    }
}
```

`Mock Test class: AnimalLocatorMock`

```
@isTest
global class AnimalLocatorMock implements HttpCalloutMock {
    global HTTPResponse respond(HTTPRequest request) {
         HttpResponse response = new HttpResponse();
        response.setHeader('Content-Type', 'application/json');
        response.setBody('{"animal":{"id":1,"name":"chicken","eat

        response.setStatusCode(200);
        return response;
    }
}
```

2. Apex SOAP Callouts Units :

Generate an Apex class using WSDL2Apex for a SOAP web service, write unit tests

that achieve 100 percent code coverage for the class using a mock response, and run your Apex tests.

Apex class：`ParkService`

```
//Generated by wsdl2apex

public class ParkService {
    public class byCountryResponse {
        public String[] return_x;
        private String[] return_x_type_info = new String[]{'return','http://parks.services/',null,'0','-

        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','fa

        private String[] 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-

        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,
              request_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');
            return response_x.return_x;
        }
    }
}
```

Apex class : `ParkLocator`

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

```
}
```

Apex Text Class : `ParkLocatorTest`

```
@isTest
private class ParkLocatorTest  {
    @isTest static void testCallout() {
        // This causes a fake response to be generated
        Test.setMock(WebServiceMock.class, new ParkServiceMock());
        // Call the method that invokes a callout
        List<String> result = new List<String>();
        List<String> expectedvalue = new List<String>{'Park1','Park2','Park3'};

        result = ParkLocator.country('India');
        // Verify that a fake result is returned
        System.assertEquals(expectedvalue, result);
    }
}
```

Apex Mock Test class : `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();
```

```

        List<String> myStrings = new List<String> {'Park1','Park2','Park3'};

        response_x.return_x = myStrings;
        // end
        response.put('response_x', response_x);
    }
}
```

3.Apex Web Services -

Create an Apex REST class that is accessible at /Accounts/<Account_ID>/contacts. The service will return the account's ID and name plus the ID and name of all contacts associated with the account.

Apex class : `AccountManager`

```
@RestResource(urlMapping = '/Accounts/*/contacts')
global with sharing class AccountManager{

    @HttpGet
    global static Account getAccount(){
        RestRequest request= restContext.request;
        string accountId =request.requestURI.substringBetween('Accounts/','/contac

        Account result = [SELECT Id ,Name,(Select Id , Name from Contacts) from Account where Id = :accountId Limit 1];
        return result;
    }
}
```

Apex Unit Test Class : `AccountManagerTest`

```
@isTest
```

```
private class AccountManagerTest{
    static testMethod void testMethod1(){
        Account objAccount = new Account(Name = 'test

        insert objAccount;
        Contact objContact = new Contact(LastName = 'test

                                         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 method to test
        Account thisAccount = AccountManager.getAccount();
        // Verify results
        System.assert(thisAccount!= null);
        System.assertEquals('test Account', thisAccount.Name);
    }
}
```

Apex specialist Challenges

step 1 - Quiz

Step 2 - Automate Record Creation

Apex Class : MaintenanceRequestHelper

```
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

                    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;
        }
    }
}
```

Apex Trigger: MaintenanceRequest

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

Step3 - Synchronize Salesforce data with an external system

Apex Calss : `WarehouseCalloutService`

```
public with sharing class WarehouseCalloutService {

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

    // complete this method to make the callout (using @future) to the
    // REST endpoint and update equipment on hand.
    @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);
        // If the request is successful, parse the JSON response.
        if (response.getStatusCode() == 200) {
```

```
        // Deserialize the JSON string into collections of primitive data types.
        List<Object> equipments = (List<Object>) JSON.deserializeUntyped(response.getBody());
            List<Product2> products = new List<Product2>();
            for(Object o :  equipments){
                Map<String, Object> mapProduct = (Map<String, Object>)o;
                Product2 product = new Product2();
                product.Name = (String)mapProduct.get('name');
                product.Cost__c = (Integer)mapProduct.get('cost');
                product.Current_Inventory__c = (Integer)mapProduct.get('quantity');
                product.Maintenance_Cycle__c = (Integer)mapProduct.get('maintenanceperiod');
                product.Replacement_Part__c = (Boolean)mapProduct.get('replacement');
                product.Lifespan_Months__c = (Integer)mapProduct.get('lifespan');
                product.Warehouse_SKU__c = (String)mapProduct.get('sku');
                product.ProductCode = (String)mapProduct.get('_id');
                products.add(product);
            }
            if(products.size() > 0){
                System.debug(products);
                upsert products;
            }
      }
    }
}
```

Step 4 - Schedule Synchronize

Apex code : WarehouseSyncSchedule

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

Step 5 - Test automation logic

Apex Trigger: MaintenanceRequest

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

Apex Class : MaintenanceRequestHelper

```
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

                    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;
        }
    }
}
```

Apex Test Class : MaintenanceRequestHelperTest

```
@isTest
public with sharing class MaintenanceRequestHelperTest {

    // createVehicle
```

```
    private static Vehicle__c createVehicle(){
        Vehicle__c vehicle = new Vehicle__C(name = 'Testing

        return vehicle;
    }

    // createEquipment
    private static Product2 createEquipment(){
        product2 equipment = new product2(name = 'Testing

                                          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
```

```
        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(createEquipmentMaintenanceIte

        }
        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);
    }
}
```

Step 6 - Schedule Synchronize

Apex Calss : `WarehouseCalloutService`

```
public with sharing class WarehouseCalloutService {

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

    // complete this method to make the callout (using @future) to the
    // REST endpoint and update equipment on hand.
    @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);
            // If the request is successful, parse the JSON response.
            if (response.getStatusCode() == 200) {
```

```
            // Deserialize the JSON string into collections of primitive data types.
            List<Object> equipments = (List<Object>)JSON.deserializeUntyped(response.getBody());

            List<Product2> products = new List<Product2>();
            for(Object o :  equipments){
                Map<String, Object> mapProduct = (Map<String, Object>)o;
                Product2 product = new Product2();
                product.Name = (String)mapProduct.get('name');
                product.Cost__c = (Integer)mapProduct.get('cost');
                product.Current_Inventory__c = (Integer)mapProduct.get('quantity');
                product.Maintenance_Cycle__c = (Integer)mapProduct.get('maintenanceperiod');
                product.Replacement_Part__c = (Boolean)mapProduct.get('replacement');
                product.Lifespan_Months__c = (Integer)mapProduct.get('lifespan');
                product.Warehouse_SKU__c = (String)mapProduct.get('sku');
                product.ProductCode = (String)mapProduct.get('_id');
                products.add(product);
            }
            if(products.size() > 0){
                System.debug(products);
                upsert products;
            }
 }
    }
}
```

Apex Mock Class: `WarehouseCalloutServiceMock`

```
@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","rep

        response.setStatusCode(200);

        return response;
    }
}
```

Apex Teast class : WarehouseCalloutServiceTest

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

Step 7 - Test Scheduling Logic

Apex Test Class : `WarehouseSyncScheduleTest`

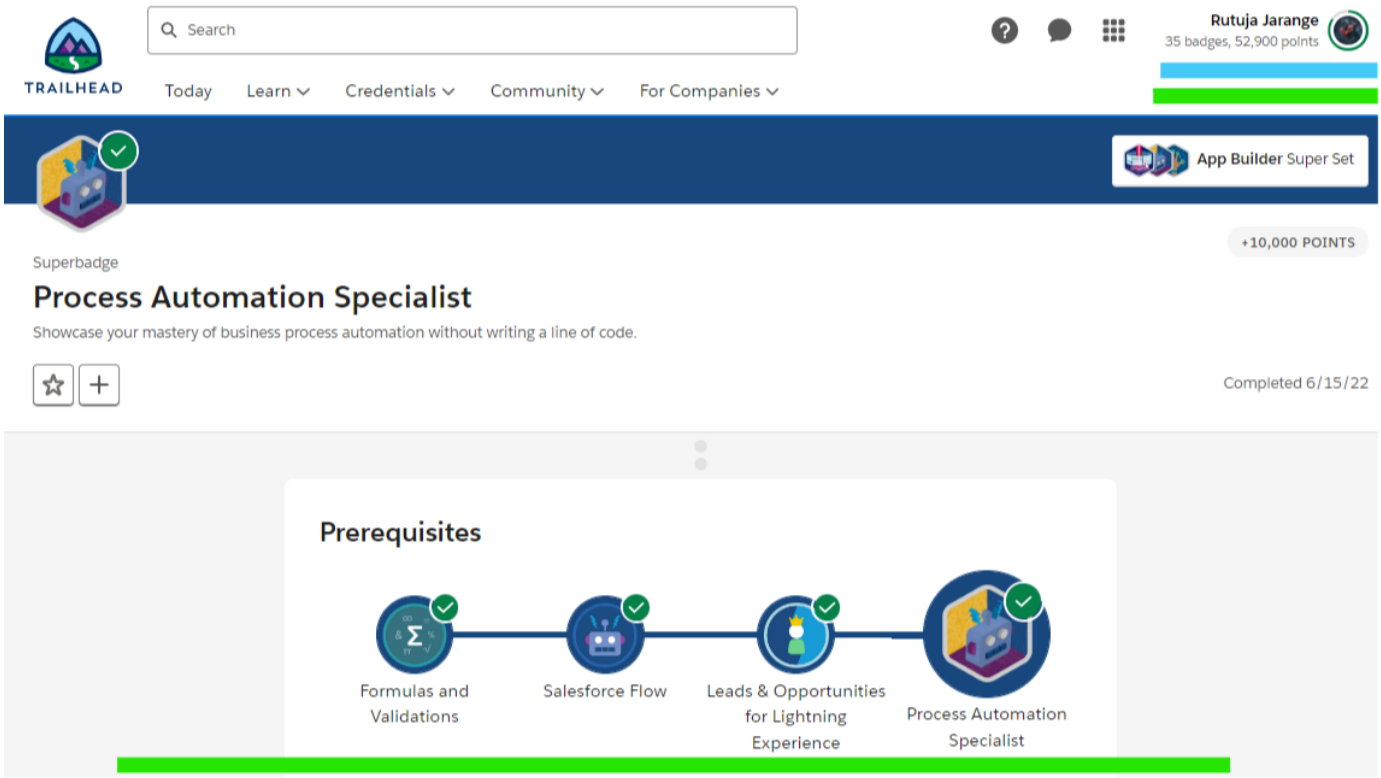
```
@isTest
public with sharing class WarehouseSyncScheduleTest {
    // implement scheduled code here
    //
    @isTest static void test() {
        String scheduleTime = '00 00 00 * * ? *';
        Test.startTest();
        Test.setMock(HttpCalloutMock.class, new WarehouseCalloutServiceMock());
 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

        Test.stopTest();
    }
}
```

Apex code : WarehouseSyncSchedule

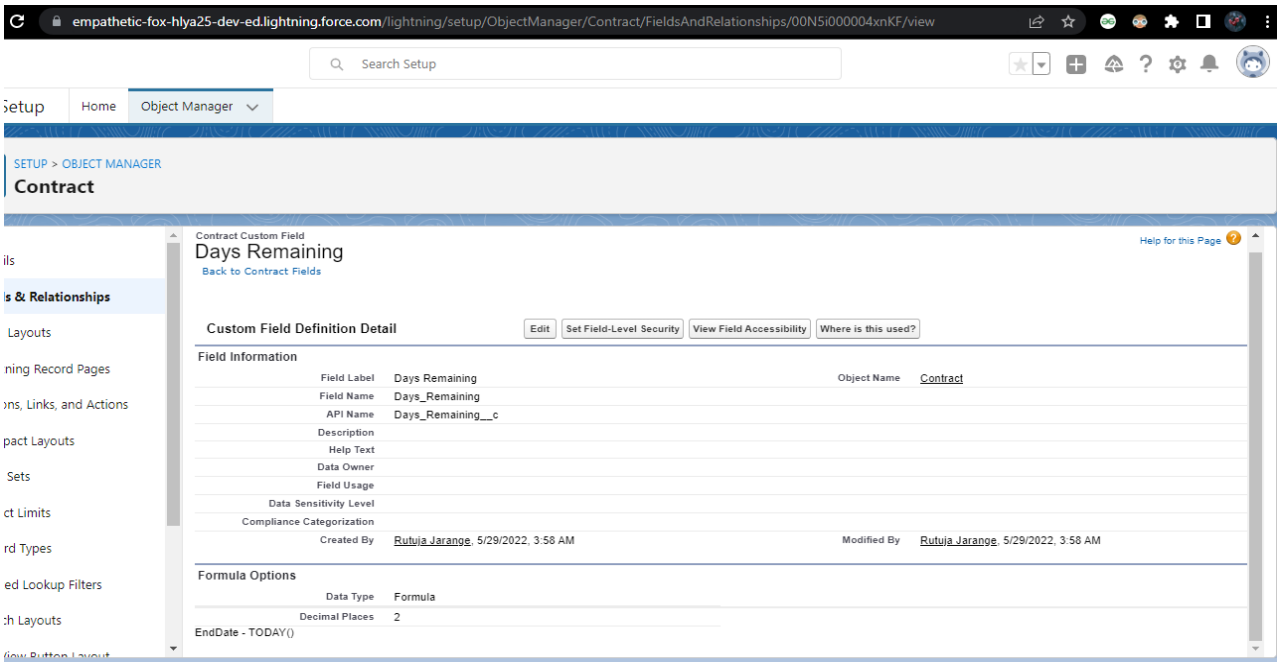
```
global with sharing class WarehouseSyncSchedule implements Schedulable{
    global void execute(SchedulableContext ctx){
```

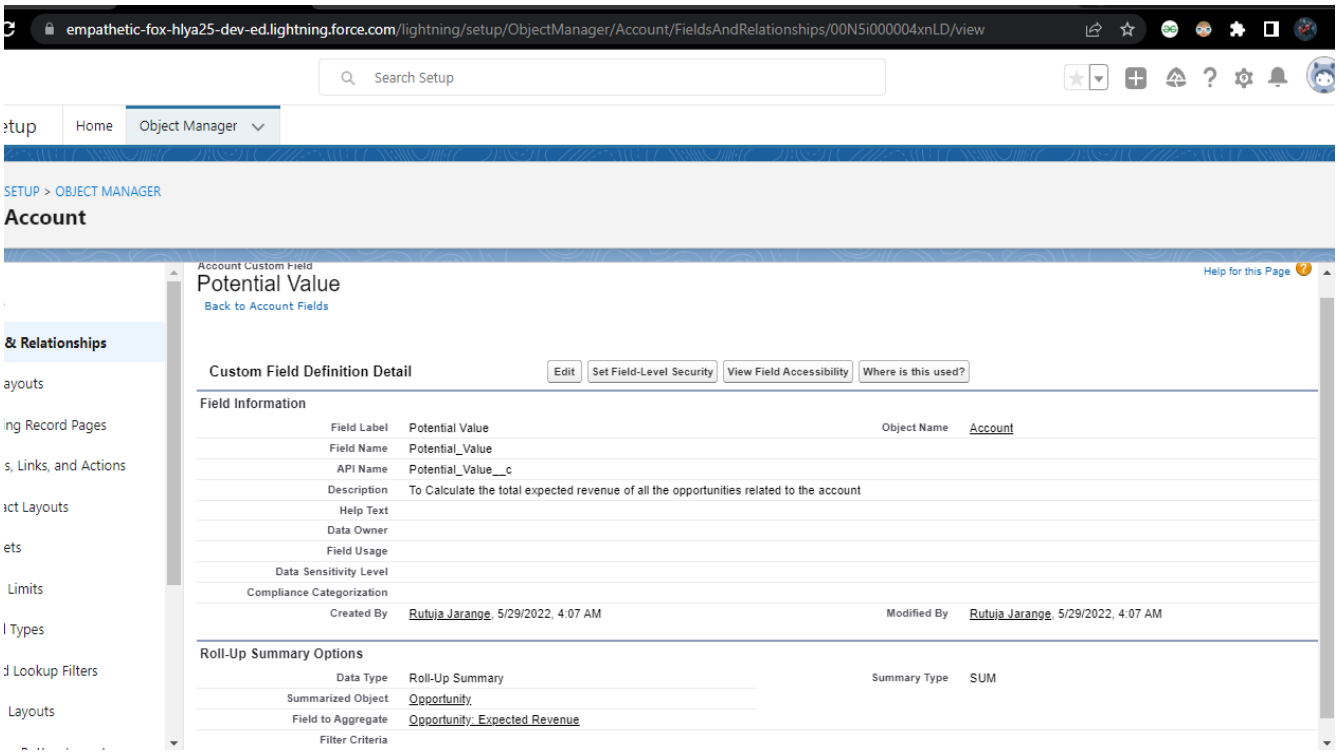
```
        System.enqueueJob(new WarehouseCalloutService());
    }
}
```

Process Automation Specialist :

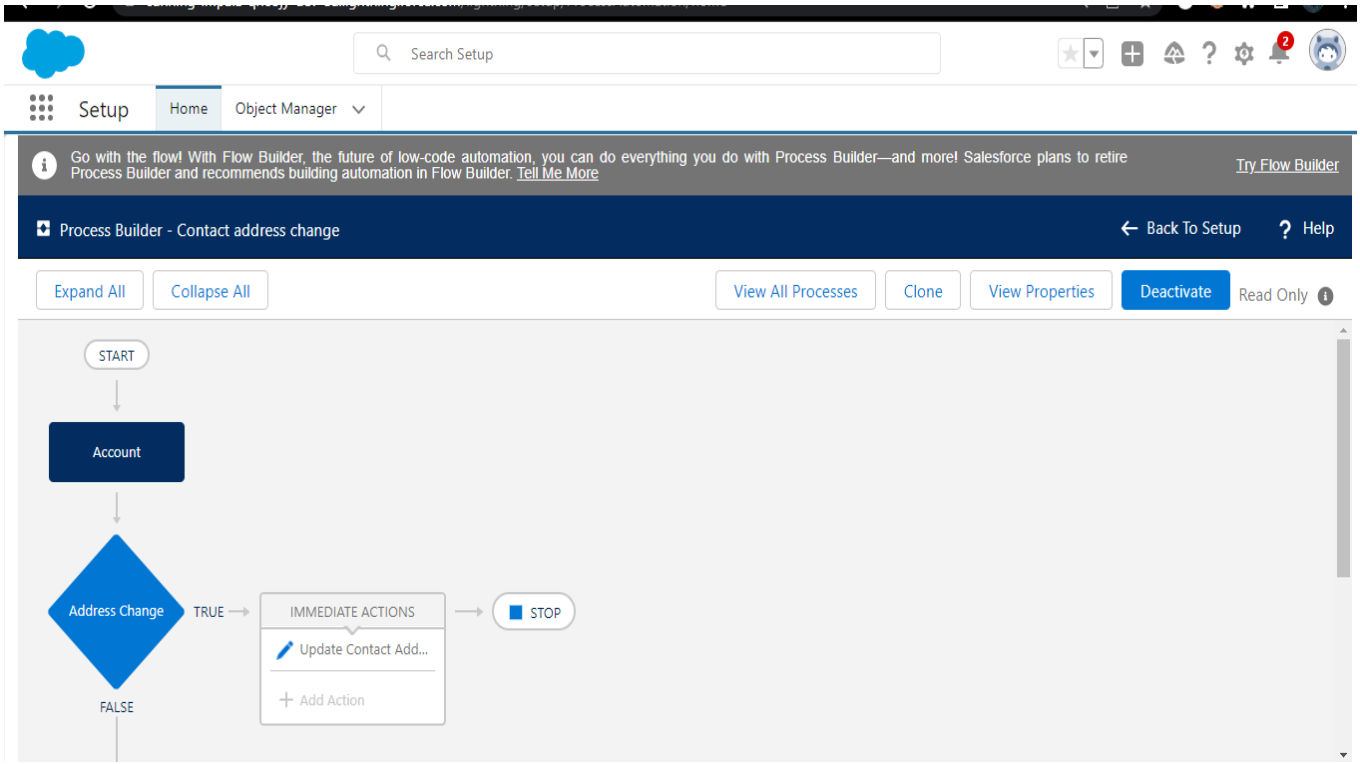


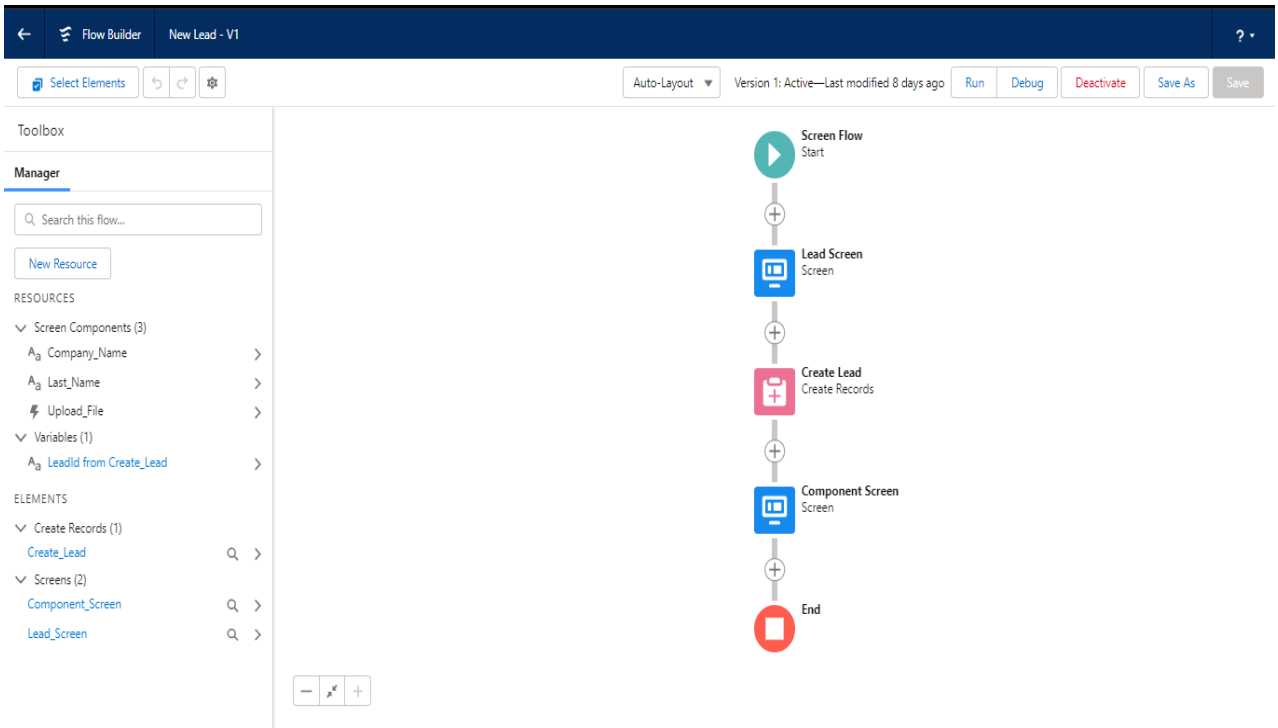
Formula and Validation:



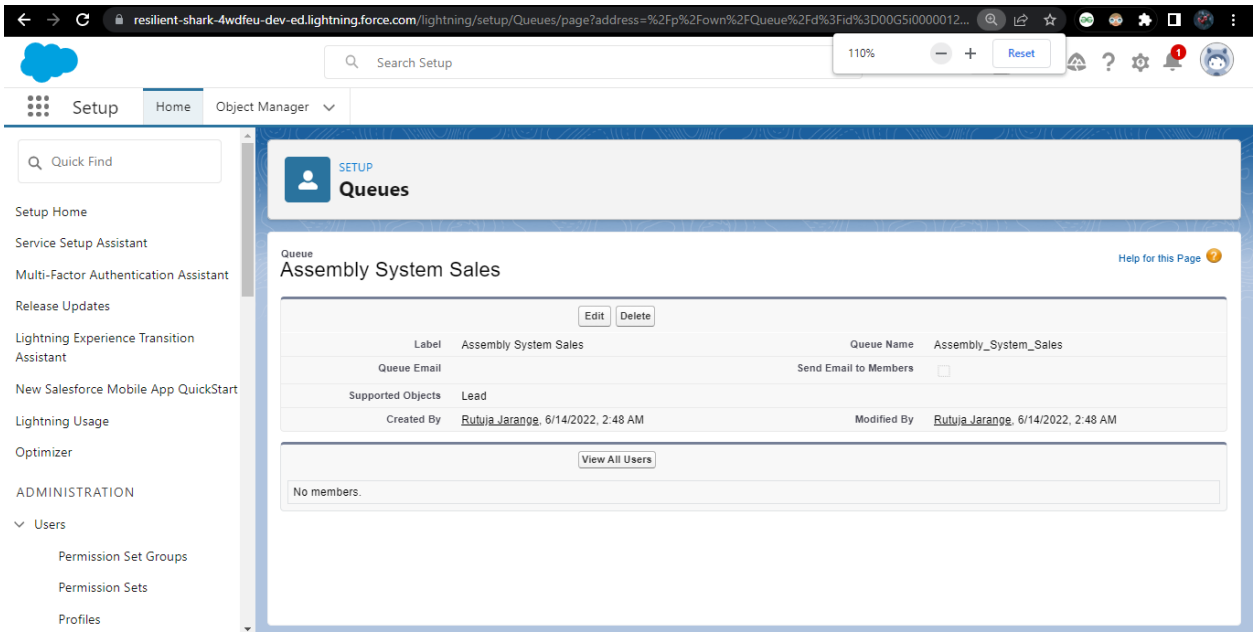


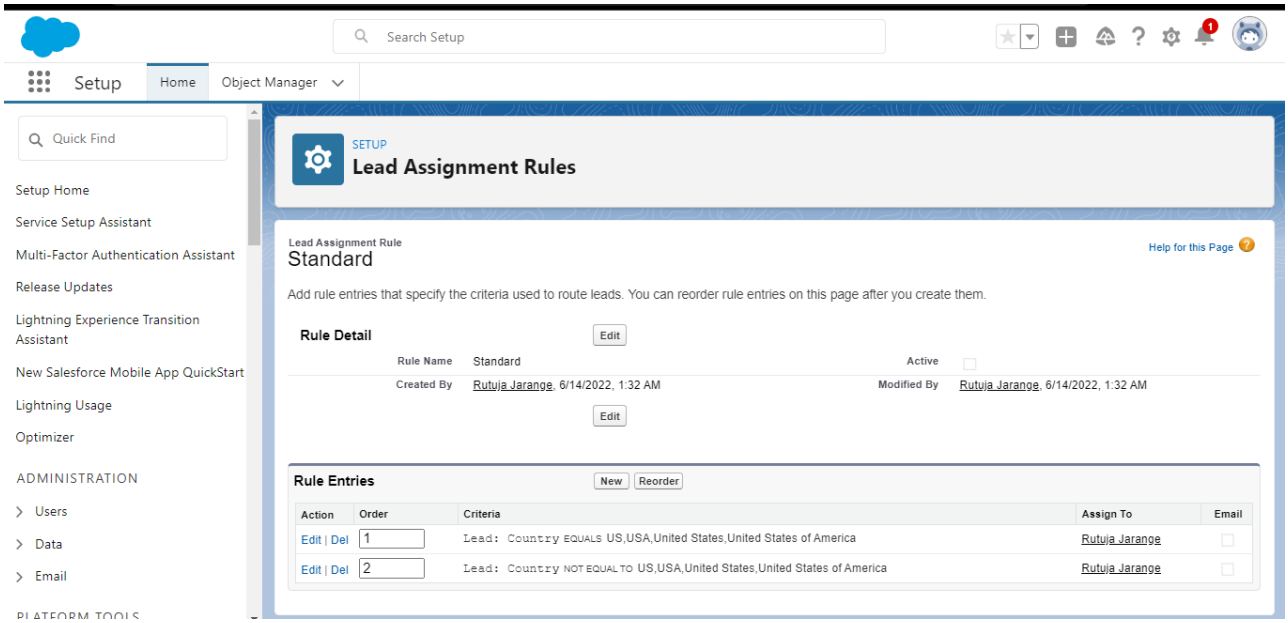
Salesforce Flows:

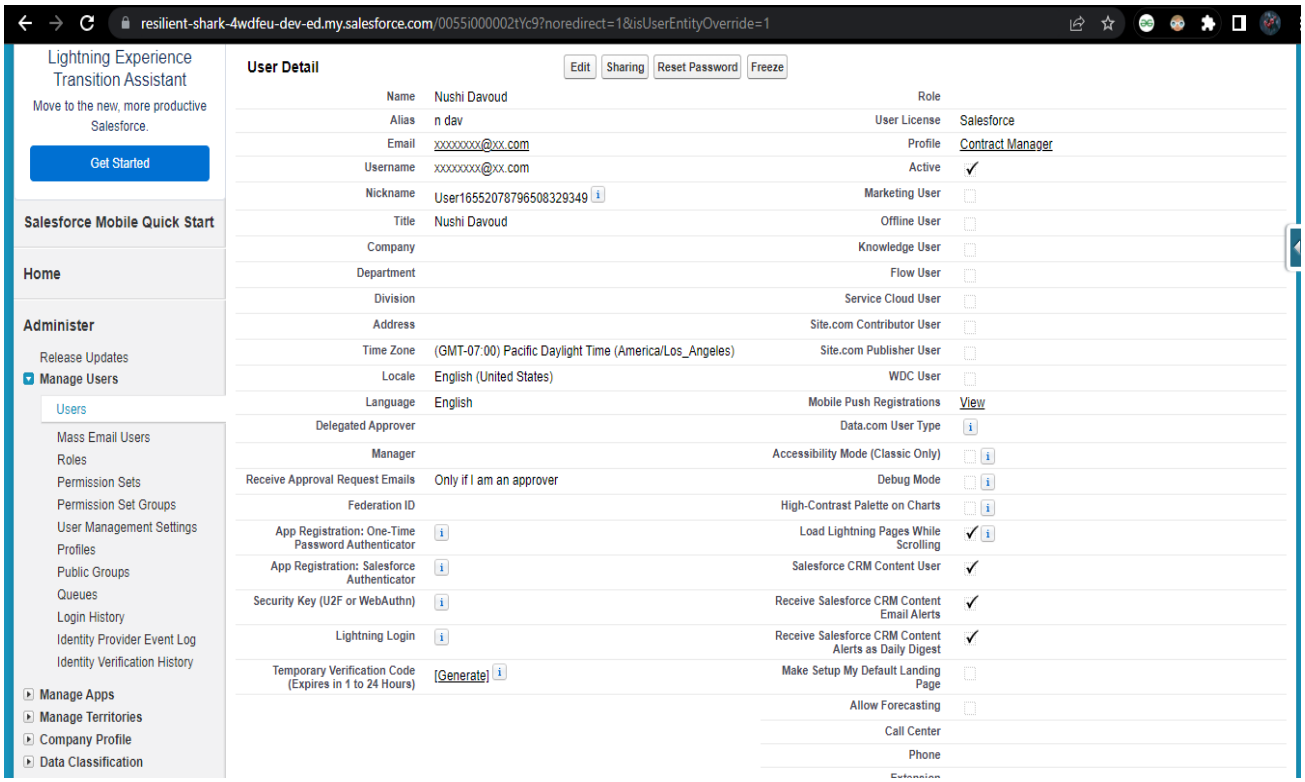


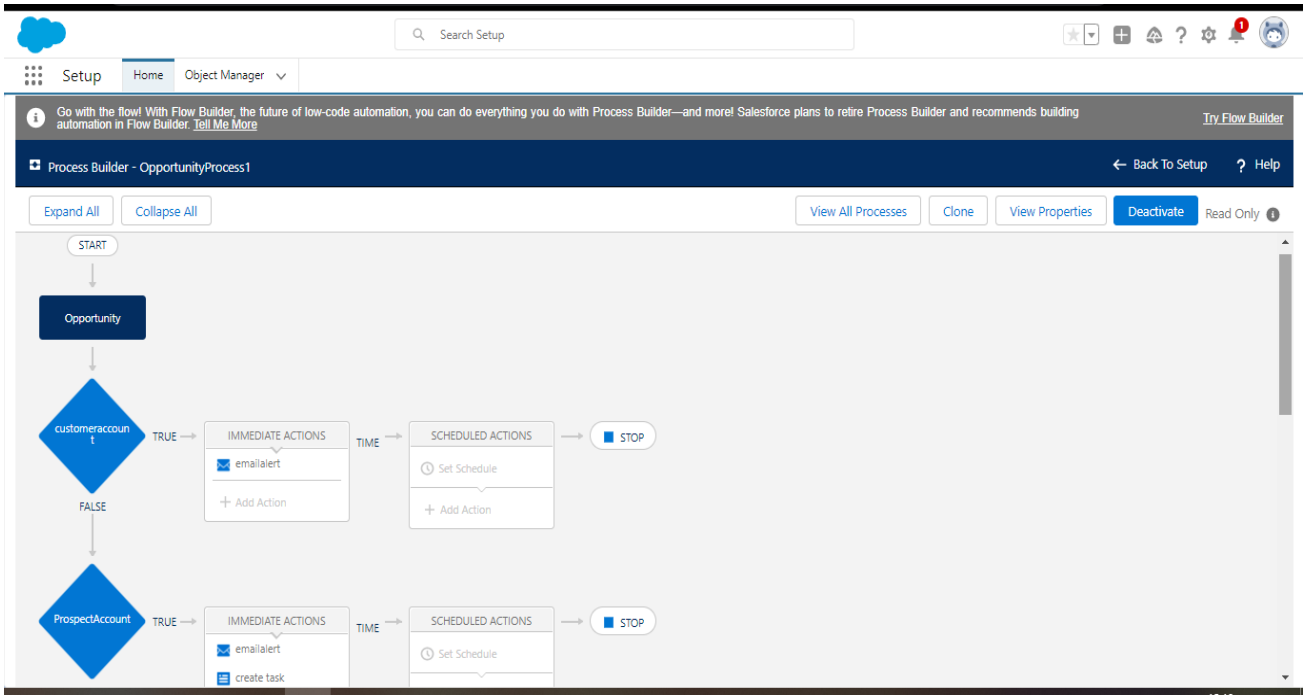

Flow Builder
New Lead - V1
Select Elements
Auto-Layout
Version 1: Active—Last modified 8 days ago
Run
Debug
Deactivate
Save As
Save
Toolbox
Manager
Search this flow...
New Resource
RESOURCES
Screen Components (3)
Company_Name
Last_Name
Upload_File
Variables (1)
LeadId from Create_Lead
ELEMENTS
Create Records (1)
Create_Lead
Screens (2)
Component_Screen
Lead_Screen
Screen Flow
Start
Lead Screen
Screen
Create Lead
Create Records
Component Screen
Screen
End

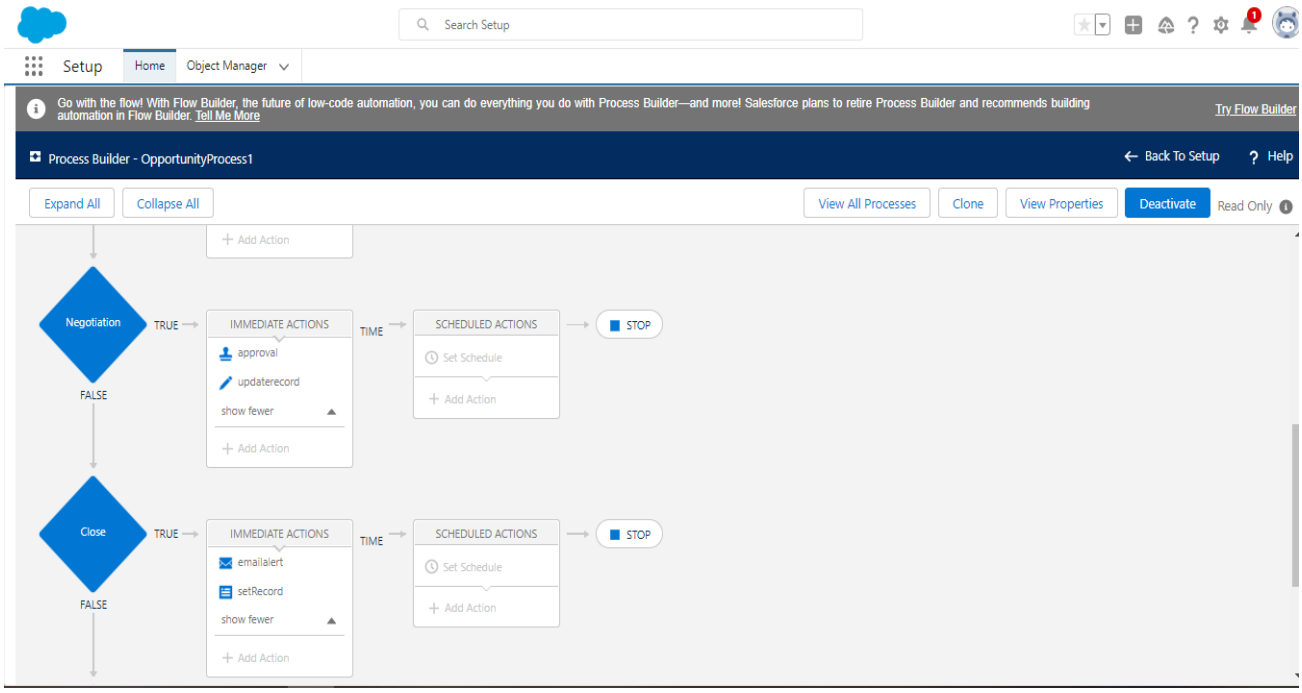
Process Automation Specialist Challenges-

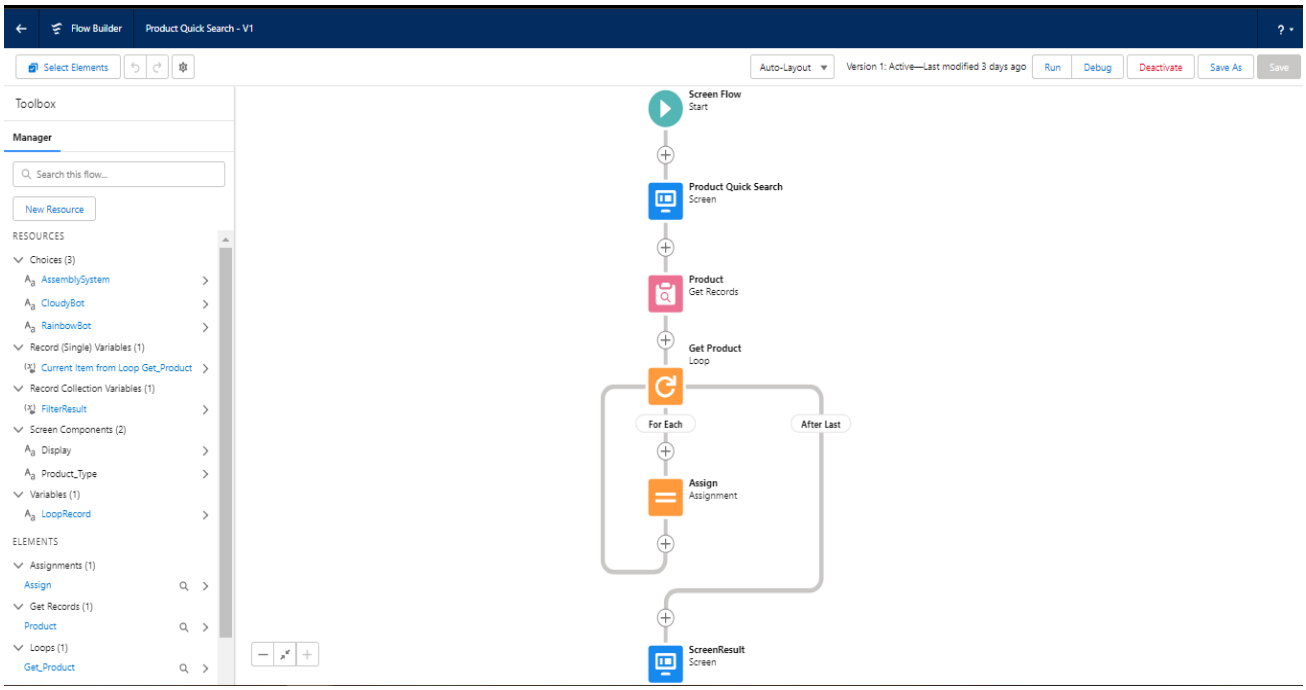





resilient-shark-4wdfeu-dev-ed.my.salesforce.com/0055i000002tYc9?noredirect=1&isUserEntityOverride=1
Lightning Experience Transition Assistant
Move to the new, more productive Salesforce.
Get Started
Salesforce Mobile Quick Start
Home
Administer
Release Updates
Manage Users
Users
Mass Email Users
Roles
Permission Sets
Permission Set Groups
User Management Settings
Profiles
Public Groups
Queues
Login History
Identity Provider Event Log
Identity Verification History
Manage Apps
Manage Territories
Company Profile
Data Classification
User Detail
Edit
Sharing
Reset Password
Freeze
Name Nushi Davoud
Alias n dav
Email xxxxxxxx@xx.com
Username xxxxxxxx@xx.com
Nickname User1655207879650832934 9
Title Nushi Davoud
Company
Department
Division
Address
Time Zone (GMT-07:00) Pacific Daylight Time (America/Los_Angeles)
Locale English (United States)
Language English
Delegated Approver
Manager
Receive Approval Request Emails Only if I am an approver
Federation ID
App Registration: One-Time Password Authenticator
App Registration: Salesforce Authenticator
Security Key (U2F or WebAuthn)
Lightning Login
Temporary Verification Code (Expires in 1 to 24 Hours) [Generate]
Role
User License Salesforce
Profile Contract Manager
Active
Marketing User
Offline User
Knowledge User
Flow User
Service Cloud User
Site.com Contributor User
Site.com Publisher User
WDC User
Mobile Push Registrations View
Data.com User Type
Accessibility Mode (Classic Only)
Debug Mode
High-Contrast Palette on Charts
Load Lightning Pages While Scrolling
Salesforce CRM Content User
Receive Salesforce CRM Content Email Alerts
Receive Salesforce CRM Content Alerts as Daily Digest
Make Setup My Default Landing Page
Allow Forecasting
Call Center
Phone


Search Setup
Setup
Home
Object Manager
Go with the flow! With Flow Builder, the future of low-code automation, you can do everything you do with Process Builder—and more! Salesforce plans to retire Process Builder and recommends building automation in Flow Builder. Tell Me More
Try Flow Builder
Process Builder - OpportunityProcess1
Back To Setup
Help
Expand All
Collapse All
View All Processes
Clone
View Properties
Deactivate
Read Only
START
Opportunity
customeraccount
TRUE
IMMEDIATE ACTIONS
emailalert
Add Action
TIME
SCHEDULED ACTIONS
Set Schedule
Add Action
STOP
FALSE
ProspectAccount
TRUE
IMMEDIATE ACTIONS
emailalert
create task
TIME
SCHEDULED ACTIONS
Set Schedule
STOP


Search Setup
Setup
Home
Object Manager
Go with the flow! With Flow Builder, the future of low-code automation, you can do everything you do with Process Builder—and more! Salesforce plans to retire Process Builder and recommends building automation in Flow Builder. Tell Me More
Try Flow Builder
Process Builder - OpportunityProcess1
Back To Setup
Help
Expand All
Collapse All
View All Processes
Clone
View Properties
Deactivate
Read Only
Add Action
Negotiation
TRUE
FALSE
IMMEDIATE ACTIONS
approval
updaterecord
show fewer
Add Action
TIME
SCHEDULED ACTIONS
Set Schedule
Add Action
STOP
Close
TRUE
FALSE
IMMEDIATE ACTIONS
emailalert
setRecord
show fewer
Add Action
TIME
SCHEDULED ACTIONS
Set Schedule
Add Action
STOP


Flow Builder
Product Quick Search - V1
Select Elements
Auto-Layout
Version 1: Active—Last modified 3 days ago
Run
Debug
Deactivate
Save As
Save
Toolbox
Manager
Search this flow...
New Resource
RESOURCES
Choices (3)
AssemblySystem
CloudyBot
RainbowBot
Record (Single) Variables (1)
Current Item from Loop Get_Product
Record Collection Variables (1)
FilterResult
Screen Components (2)
Display
Product_Type
Variables (1)
LoopRecord
ELEMENTS
Assignments (1)
Assign
Get Records (1)
Product
Loops (1)
Get_Product
Screen Flow
Start
Product Quick Search
Screen
Product
Get Records
Get Product
Loop
For Each
After Last
Assign
Assignment
ScreenResult
Screen