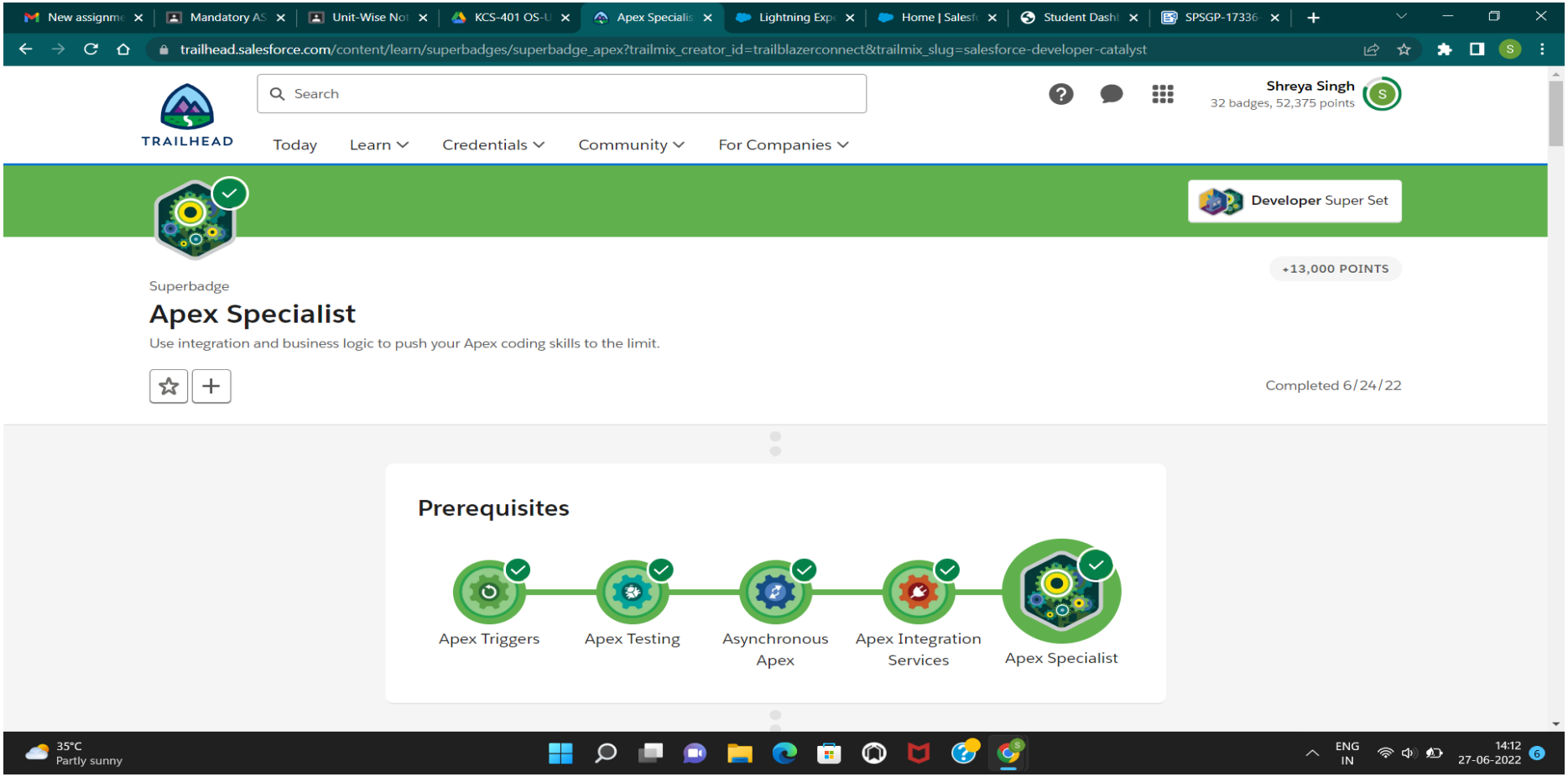
APEX SPECIALIST SUPERBATCH

PREREQUISITES:



SET UP DEVELOPMENT ORG:

1. Create a new Trailhead Playground or Developer Edition Org for this superbadge.
2. Install this unlocked package (package ID: 04t6g000008av9iAAA).
3. Add pick list values `Repair` and `Routine Maintenance` to the **Type** field on the Case object.
4. Update the Case page layout assignment to use the **Case (HowWeRoll) Layout** for your profile.
5. Rename the tab/label for the Case tab to `Maintenance Request.`

`6.`Update the Product page layout assignment to use the **Product (HowWeRoll) Layout** for your profile.

7. Rename the tab/label for the Product object to `Equipment.`

8.Use App Launcher to navigate to the **Create Default Data** tab of the **How We Roll Maintenance** app. Click **Create Data** to generate sample data for the application.

CHALLENGE 1: QUIZ

This challenge consisted of a quiz with MCQs.

CHALLENGE 2: AUTOMATE RECORD CREATION

In this module, we created two apex classes with following codes:

MaintenanceRequest.apxt

```
trigger MaintenanceRequest on Case (before update, after update) {

    if(Trigger.isUpdate && Trigger.isAfter){
        MaintenanceRequestHelper.updateWorkOrders(Trigger.New, Trigger.OldMap);
 }
}
```

MaintenanceRequestHelper.apxc

```
Public with sharing class MaintenanceRequestHelper {
    public static void updateworkOrders(List<Case> updWorkOrders, Map<Id,Case> nonUpdCaseMap) {
        Set<Id> validIds = new Set<Id>();



        For (Case c : updWorkOrders){
            if (nonUpdCaseMap.get(c.Id).Status != 'Closed' && c.Status == 'Closed'){
                if (c.Type == 'Repair' || c.Type == 'Routine Maintenance'){
                    validIds.add(c.Id);



                }
            }
        }


           if (!validIds.isEmpty()){
            List<Case> newCases = new List<Case>();
            Map<Id,Case> closedCasesM = new Map<Id,Case>([SELECT Id, Vehicle__c, Equipment__c, Equipment__r.Maintenance_Cycle__c,(SELECT
Id,Equipment__c,Quantity__c FROM Equipment_Maintenance_Items__r)
                        FROM Case WHERE Id IN :validIds]);
```

```
 Map<Id,Decimal> maintenanceCycles = new Map<ID,Decimal>();
            AggregateResult[] results = [SELECT Maintenance_Request__c, MIN(Equipment__r.Maintenance_Cycle__c)cycle FROM Equipment_Maintenance_Item__c WHERE
Maintenance_Request__c IN :ValidIds GROUP BY Maintenance_Request__c];

 for (AggregateResult ar : results){
            maintenanceCycles.put((Id) ar.get('Maintenance_Request__c'), (Decimal) ar.get('cycle'));
        }

            for(Case cc : closedCasesM.values()){
                Case nc = new Case (
                    ParentId = cc.Id,
                Status = 'New',
                    Subject = 'Routine Maintenance',
                    Type = 'Routine Maintenance',
                    Vehicle__c = cc.Vehicle__c,
                    Equipment__c =cc.Equipment__c,
                    Origin = 'Web',
                    Date_Reported__c = Date.Today()

                );

                If (maintenanceCycles.containskey(cc.Id)){
                    nc.Date_Due__c = Date.today().addDays((Integer) maintenanceCycles.get(cc.Id));
                }

                newCases.add(nc);
            }

           insert newCases;

           List<Equipment_Maintenance_Item__c> clonedWPs = new List<Equipment_Maintenance_Item__c>();
```

```
            for (Case nc : newCases){
                for (Equipment_Maintenance_Item__c wp : closedCasesM.get(nc.ParentId).Equipment_Maintenance_Items__r){
                    Equipment_Maintenance_Item__c wpClone = wp.clone();
                    wpClone.Maintenance_Request__c = nc.Id;
                    ClonedWPs.add(wpClone);

                }
            }
            insert ClonedWPs;
        }
    }
}
```

CHALLENGE 3: SYNCHRONIZE SALESFORCE DATA WITH AN EXTERNAL SYSTEM:

In this module we created an apex class and a trigger using following code:

WarehouseCalloutService.apxc:

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

    @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');
    }

}
```

In anonymous window:

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

CHALLENGE 4: SCHEDULE SYNCHRONIZATION USING APEX CODE :

WarehouseSyncSchedule.apxc:

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

WarehouseCalloutService.apxc:

```
public with sharing class WarehouseCalloutService implements Queueable {
```

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

@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');
    }

}
```

CHALLENGE 5: TEST AUTOMATION LOGIC :

MaintenanceRequestHelperTest.apxc:

```
@istest
public with sharing class MaintenanceRequestHelperTest {
```

```
private static final string STATUS_NEW = 'New';
private static final string WORKING = 'Working';
private static final string CLOSED = 'Closed';
private static final string REPAIR = 'Repair';
private static final string REQUEST_ORIGIN = 'Web';
private static final string REQUEST_TYPE = 'Routine Maintenance';
private static final string REQUEST_SUBJECT = 'Testing subject';

PRIVATE STATIC Vehicle__c createVehicle(){
    Vehicle__c Vehicle = new Vehicle__C(name = 'SuperTruck');
    return Vehicle;
}

PRIVATE STATIC Product2 createEq(){
    product2 equipment = new product2(name = 'SuperEquipment',
                                      lifespan_months__C = 10,
                                      maintenance_cycle__C = 10,
                                      replacement_part__c = true);
    return equipment;
}

PRIVATE STATIC Case createMaintenanceRequest(id vehicleId, id equipmentId){
    case cs = new case(Type=REPAIR,
                       Status=STATUS_NEW,
                       Origin=REQUEST_ORIGIN,
                       Subject=REQUEST_SUBJECT,
                       Equipment__c=equipmentId,
                       Vehicle__c=vehicleId);
    return cs;
}
```

```
PRIVATE STATIC Equipment_Maintenance_Item__c createWorkPart(id equipmentId,id requestId){
    Equipment_Maintenance_Item__c wp = new Equipment_Maintenance_Item__c(Equipment__c = equipmentId,
                                                                         Maintenance_Request__c = requestId);
    return wp;
}


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

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

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

    Equipment_Maintenance_Item__c workP = createWorkPart(equipmentId,somethingToUpdate.id);
    insert workP;

    test.startTest();
    somethingToUpdate.status = CLOSED;
    update somethingToUpdate;
    test.stopTest();

    Case newReq = [Select id, subject, type, Equipment__c, Date_Reported__c, Vehicle__c, Date_Due__c
                   from case
```

```
                where status =:STATUS_NEW];

    Equipment_Maintenance_Item__c workPart = [select id
                                              from Equipment_Maintenance_Item__c
                                              where Maintenance_Request__c =:newReq.Id];

    system.assert(workPart != null);
    system.assert(newReq.Subject != null);
    system.assertEquals(newReq.Type, REQUEST_TYPE);
    SYSTEM.assertEquals(newReq.Equipment__c, equipmentId);
    SYSTEM.assertEquals(newReq.Vehicle__c, vehicleId);
    SYSTEM.assertEquals(newReq.Date_Reported__c, system.today());
}

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

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

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

    Equipment_Maintenance_Item__c workP = createWorkPart(equipmentId, emptyReq.Id);
    insert workP;

    test.startTest();
```

```
    emptyReq.Status = WORKING;
    update emptyReq;
    test.stopTest();

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

    Equipment_Maintenance_Item__c workPart = [select id
                                              from Equipment_Maintenance_Item__c
                                              where Maintenance_Request__c = :emptyReq.Id];

    system.assert(workPart != null);
    system.assert(allRequest.size() == 1);
}

@istest
private static void testMaintenanceRequestBulk(){
    list<Vehicle__C> vehicleList = new list<Vehicle__C>();
    list<Product2> equipmentList = new list<Product2>();
    list<Equipment_Maintenance_Item__c> workPartList = new list<Equipment_Maintenance_Item__c>();
    list<case> requestList = new list<case>();
    list<id> oldRequestIds = new list<id>();

    for(integer i = 0; i < 300; i++){
       vehicleList.add(createVehicle());
        equipmentList.add(createEq());
    }
    insert vehicleList;
    insert equipmentList;

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

```
            requestList.add(createMaintenanceRequest(vehicleList.get(i).id, equipmentList.get(i).id));
        }
        insert requestList;

      for(integer i = 0; i < 300; i++){
            workPartList.add(createWorkPart(equipmentList.get(i).id, requestList.get(i).id));
        }
        insert workPartList;

        test.startTest();
        for(case req : requestList){
            req.Status = CLOSED;
            oldRequestIds.add(req.Id);
        }
        update requestList;
        test.stopTest();

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

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

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

MaintenanceRequestHelper.apxc:

```
public with sharing class MaintenanceRequestHelper {
```

```
    public static void updateworkOrders(List<Case> updWorkOrders, Map<Id,Case> nonUpdCaseMap) {
        Set<Id> validIds = new Set<Id>();


        For (Case c : updWorkOrders){
            if (nonUpdCaseMap.get(c.Id).Status != 'Closed' && c.Status == 'Closed'){
                if (c.Type == 'Repair' || c.Type == 'Routine Maintenance'){
                    validIds.add(c.Id);



                }
            }
        }

        if (!validIds.isEmpty()){
            List<Case> newCases = new List<Case>();
            Map<Id,Case> closedCasesM = new Map<Id,Case>([SELECT Id, Vehicle__c, Equipment__c,
Equipment__r.Maintenance_Cycle__c,(SELECT Id,Equipment__c,Quantity__c FROM Equipment_Maintenance_Items__r)
                                                          FROM Case WHERE Id IN :validIds]);
            Map<Id,Decimal> maintenanceCycles = new Map<ID,Decimal>();
            AggregateResult[] results = [SELECT Maintenance_Request__c, MIN(Equipment__r.Maintenance_Cycle__c)cycle FROM
Equipment_Maintenance_Item__c WHERE Maintenance_Request__c IN :ValidIds GROUP BY Maintenance_Request__c];

        for (AggregateResult ar : results){
            maintenanceCycles.put((Id) ar.get('Maintenance_Request__c'), (Decimal) ar.get('cycle'));
        }

            for(Case cc : closedCasesM.values()){
                Case nc = new Case (
                    ParentId = cc.Id,
                Status = 'New',
```

```
                Subject = 'Routine Maintenance',
                Type = 'Routine Maintenance',
                Vehicle__c = cc.Vehicle__c,
                Equipment__c =cc.Equipment__c,
                Origin = 'Web',
                Date_Reported__c = Date.Today()

            );

            If (maintenanceCycles.containskey(cc.Id)){
                nc.Date_Due__c = Date.today().addDays((Integer) maintenanceCycles.get(cc.Id));
            }

            newCases.add(nc);
        }

        insert newCases;

        List<Equipment_Maintenance_Item__c> clonedWPs = new List<Equipment_Maintenance_Item__c>();
        for (Case nc : newCases){
            for (Equipment_Maintenance_Item__c wp : closedCasesM.get(nc.ParentId).Equipment_Maintenance_Items__r){
                Equipment_Maintenance_Item__c wpClone = wp.clone();
                wpClone.Maintenance_Request__c = nc.Id;
                ClonedWPs.add(wpClone);

            }
        }
        insert ClonedWPs;
    }
  }
}
```

MaintenanceRequest.apxt:

```
trigger MaintenanceRequest on Case (before update, after update) {

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

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

    }

}
```

CHALLENGE 6: TEST CALLOUT LOGIC :

WarehouseCalloutService.apxc:

WarehouseCalloutServiceTest.apxc:

```
@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);
    }
}
```

WarehouseCalloutServiceMock.apxc:

```
@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":"55d66226726b611100aaf742","replacement":true,"qu
antity":183,"name":"Cooling
Fan","maintenanceperiod":0,"lifespan":0,"cost":300,"sku":"100004"},{"_id":"55d66226726b611100aaf743","replacement":true,"quanti
ty":143,"name":"Fuse 20A","maintenanceperiod":0,"lifespan":0,"cost":22,"sku":"100005"}]');
        response.setStatusCode(200);

        return response;
    }
}
```

CHALLENGE 7. TEST SCHEDULING LOGIC :

WarehouseSyncSchedule.apxc:

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

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

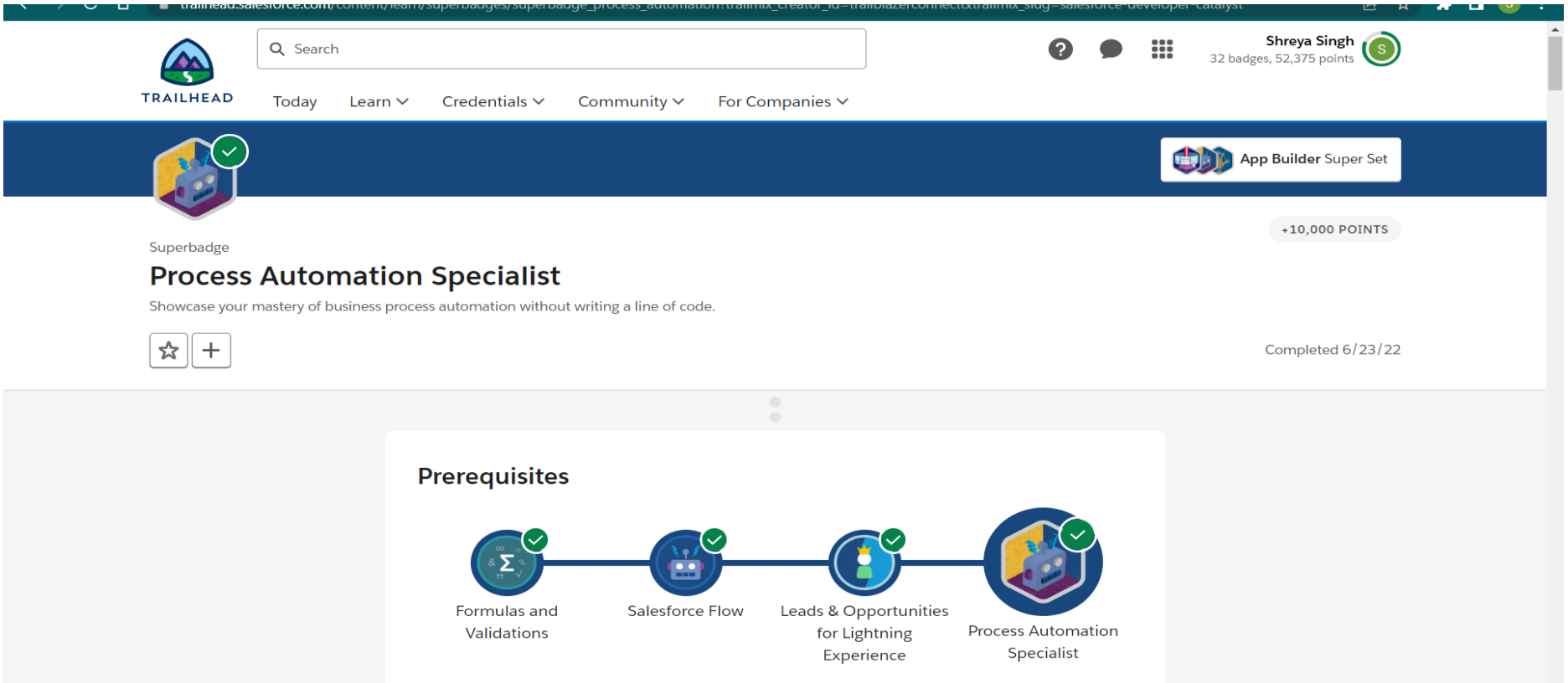
WarehouseSyncScheduleTest.apxc:

```
@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();
    }
}
```

PROCESS AUTOMATION SPECIALIST

PREREQUISITES:



CHALLENGE 1: QUIZ

This challenge consisted of a quiz with MCQs.

CHALLENGE 2: AUTOMATE LEADS

We had to create validation rule on Lead.
Rule name: Anything
Error condition formula:
OR(AND(LEN(State) > 2,
NOT(CONTAINS("AL:AK:AZ:AR:CA:CO:CT:DE:DC:FL:GA:HI:ID:IL:IN:IA:KS:KY:LA:ME:MD:MA:MI:MN:MS:MO:MT:NE:NV:NH:NJ:NM:NY:NC:ND:OH:OK:OR:PA:RI:SC:SD:TN:TX:UT:VT:VA:WA:WV:WI:WY", State))), NOT(OR(Country ="US",Country ="USA",Country ="United States", ISBLANK(Country))))

Then we created two queues:
Rainbow Sales and Assembly Sales

Then we created two Lead assignment rules for the queues.

CHALLENGE 2: AUTOMATE ACCOUNTS

We created 4 Rollup Summary fields:
Field 1:
Label: Number of deals
Summary type: COUNT
Summarized Object: Opportunity

Filter Criteria:None
Field 2:
Label: Number of won deals
Summary type: COUNT
Summarized Object: Opportunity
Filter Criteria: Stage EQUALS Closed Won

Field 3:
Label: Last won deal date
Summary type: MAX
Field to Aggregate: Opportunity: Closed Date
Summarized Object: Opportunity
Filter Criteria:Stage EQUALS Closed Won

Field 4:
Label: Amount of won deals
Summary type: SUM
Field to Aggregate: Opportunity: Amount
Summarized Object: Opportunity
Filter Criteria:Stage EQUALS Closed Won

We created 2 Formula fields:
Field 5:
Label: Deal win percent
Return Type: Percent
Decimal Places: 2
Formula: (Number_of_won_deals__c/Number_of_deals__c)

Field 6:
Label: Call for Service
Return Type: Text
Decimal Places: 2
Formula: IF(DATE(YEAR(Last_won_deal_date__c)+2,MONTH(Last_won_deal_date__c),DAY(Last_won_deal_date__c))<=TODAY(),"Yes","No")

Then Create 2 Validation Rules

RULE1:

Rule Name:US_AddressSomething

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

Error Message: You can not save a new account unless the shipping and billing state fields are valid US state abbreviations, and the country field is either blank or US,USA, or United States.

Error Location: Top of Page

RULE2:

Rule Name: Name_Change
Error Condition Formula: ISCHANGED(Name) && (OR(ISPICKVAL(Type ,'Customer - Direct') ,ISPICKVAL(Type ,'Customer - Channel')))

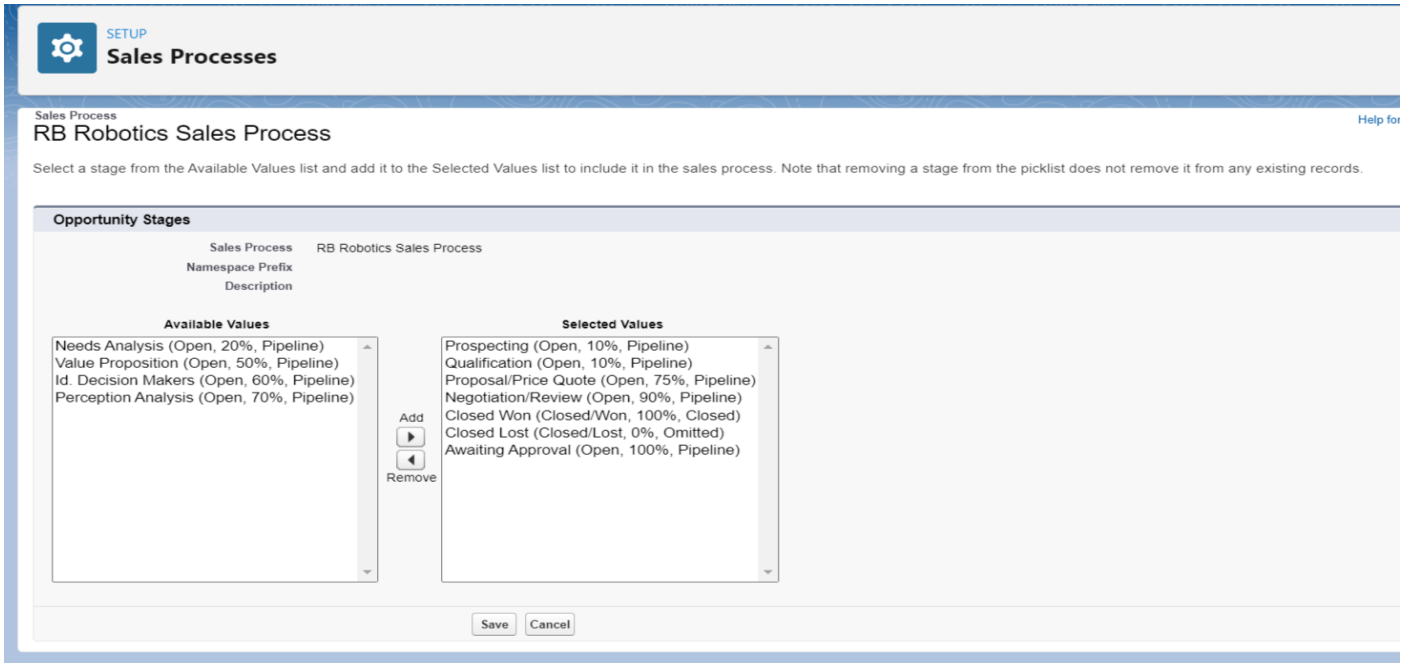
Error Message: You can't change the Account name for "Customer - Direct" or "Customer - Channel"
Error Location: Account Name

CHALLENGE 3:CREATE ROBOT SETUP OBJECT

Create a custom object Robot Setup with master detail relationship to Opprotunity.
Create fields names
Date=>Date type, Day of week=>Number , Notes=>Text type.

CHALLENGE 4: CREATE SALES PROCESS AND VALIDATE OOPORTUNITIES

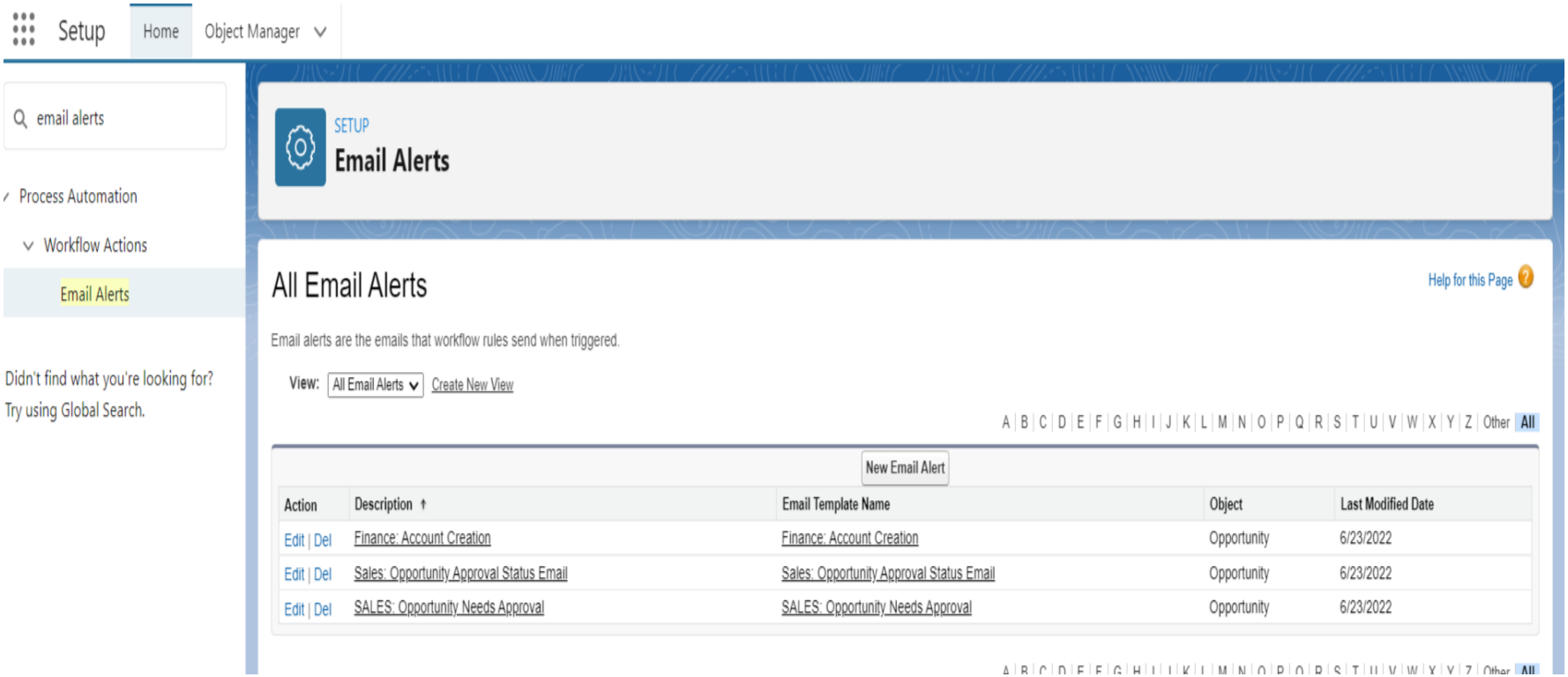
In Opportunities Object=>Fields and relationships=Stage=> add picklist value "Awaiting Approval".
Next, we created a sales process named RB Robotics Sales Process.



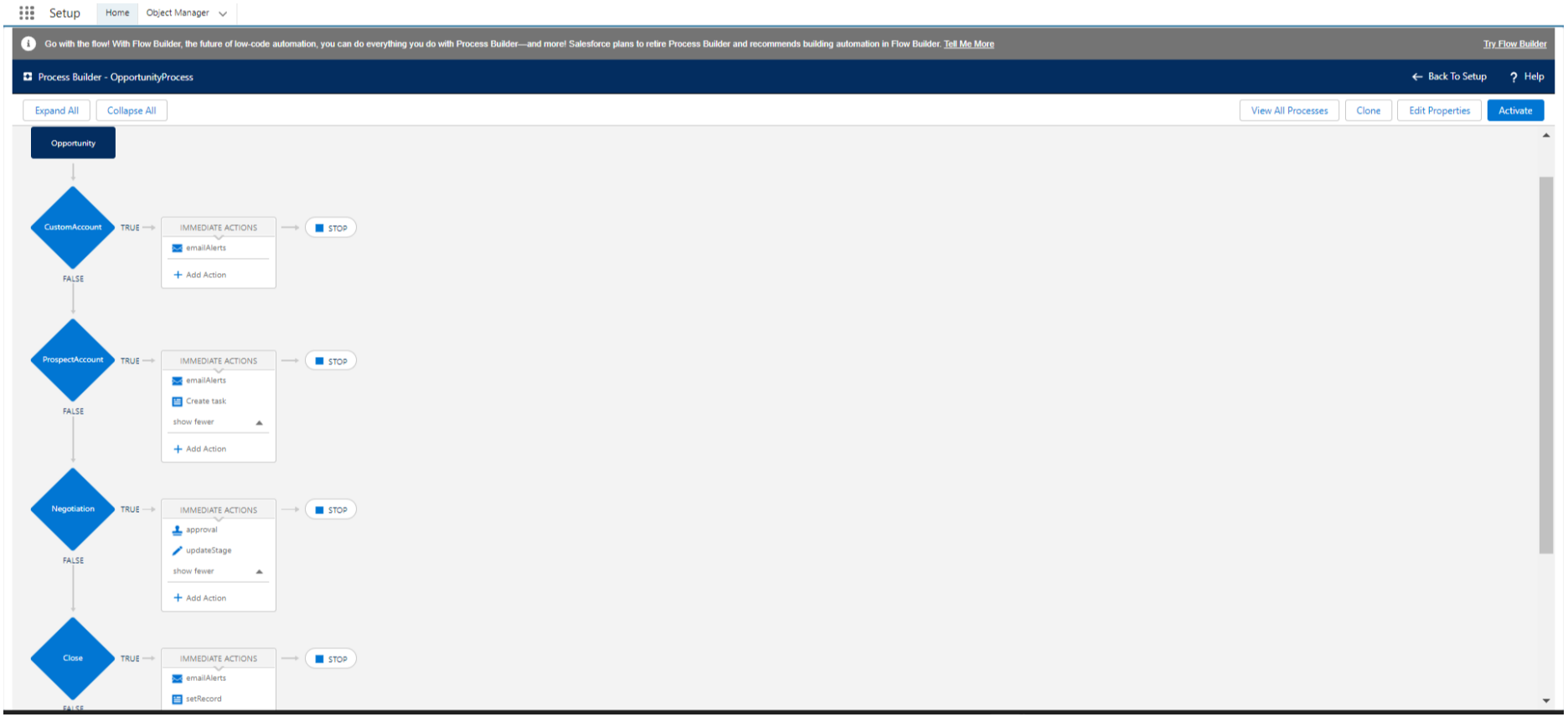
Add Opportunity Validation Rule with Error Formula: IF((Amount > 100000 && Approved__c <> True && ISPICKVAL(StageName,'Closed Won')),True,False)

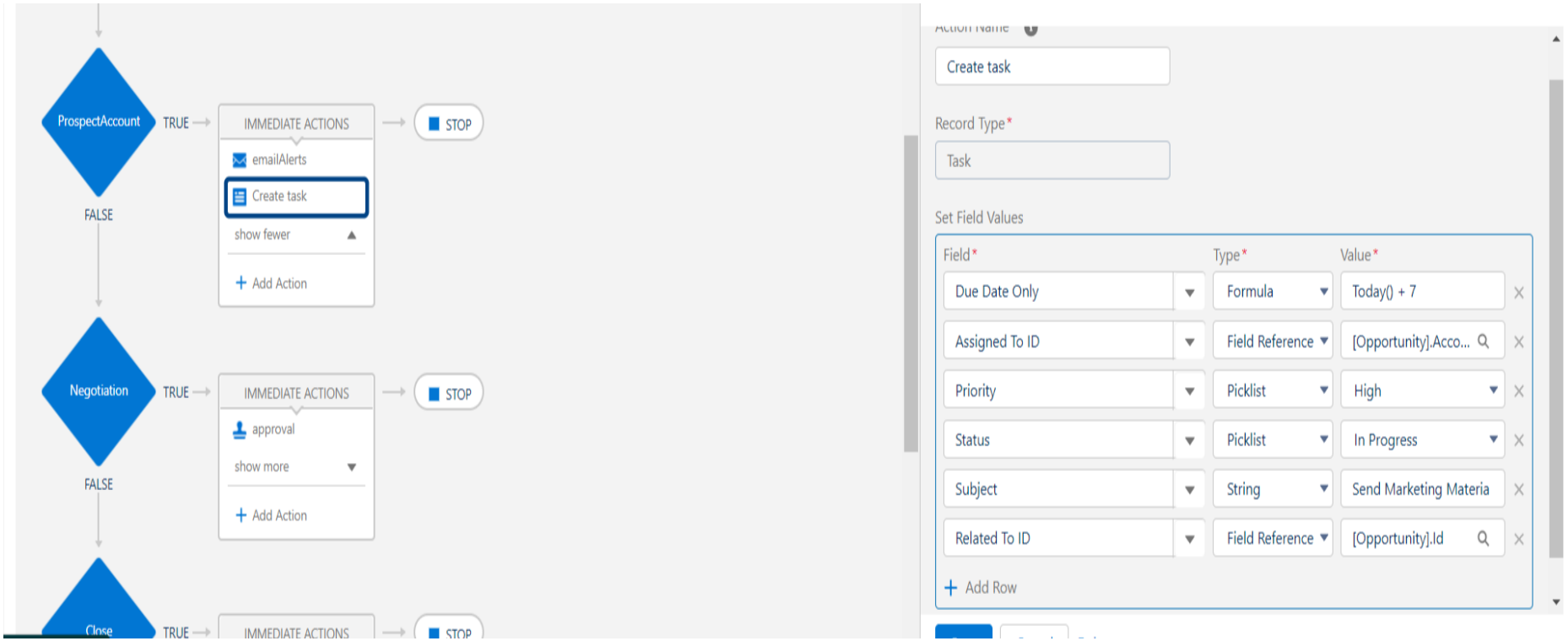
CHALLENGE 5: AUTOMATE OPPORTUNITIES

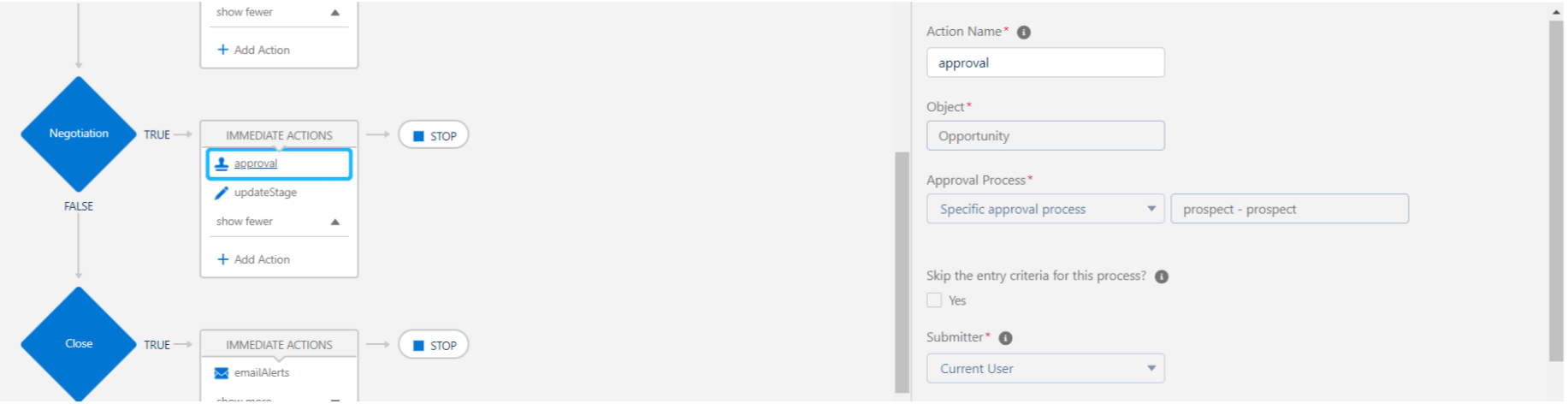
We first created three email templates:

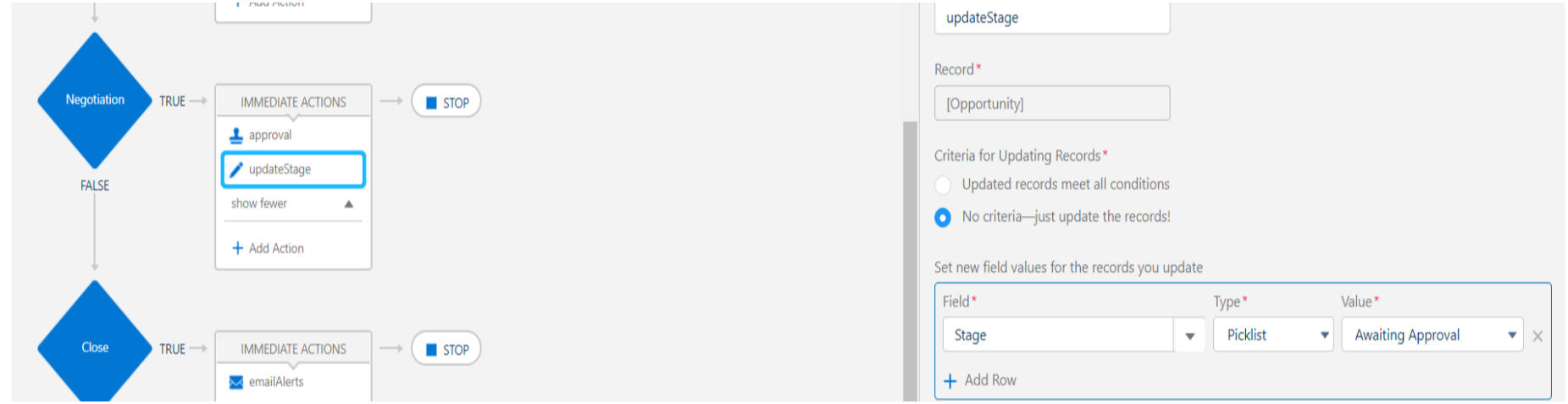


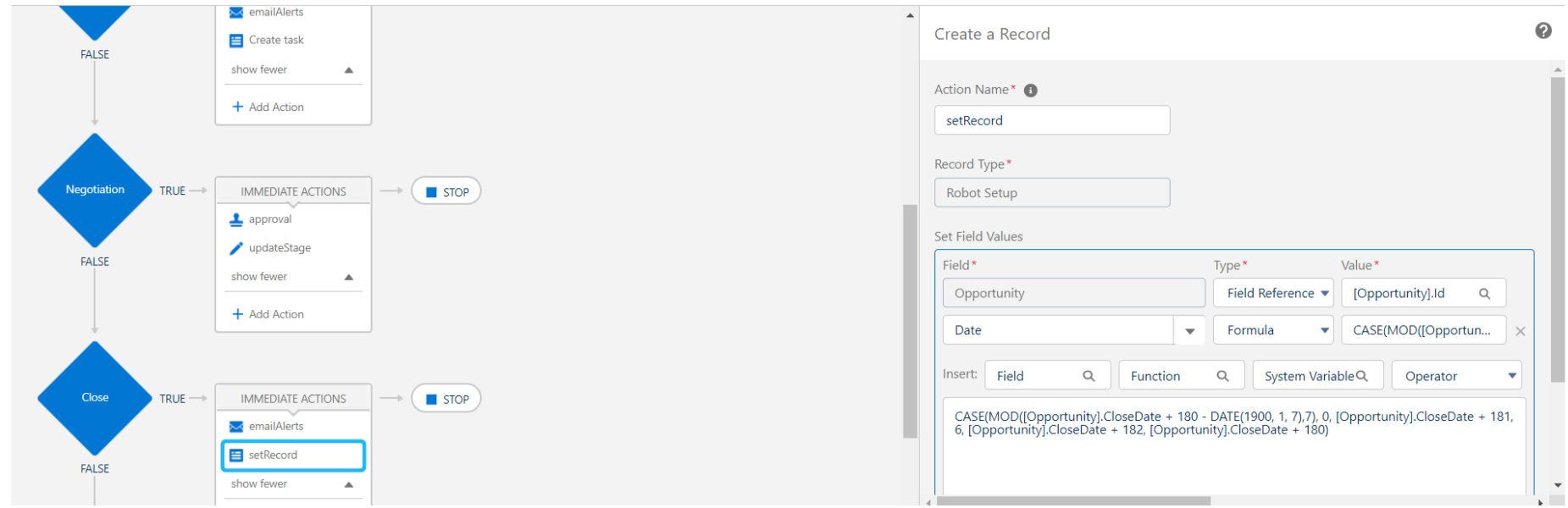
Then we created OpportunityProcess with Process Builder:


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 - OpportunityProcess
Back To Setup
Help
Expand All
Collapse All
View All Processes
Clone
Edit Properties
Activate
Opportunity
CustomAccount
TRUE
IMMEDIATE ACTIONS
emailAlerts
Add Action
STOP
FALSE
ProspectAccount
TRUE
IMMEDIATE ACTIONS
emailAlerts
Create task
show fewer
Add Action
STOP
FALSE
Negotiation
TRUE
IMMEDIATE ACTIONS
approval
updateStage
show fewer
Add Action
STOP
FALSE
Close
TRUE
IMMEDIATE ACTIONS
emailAlerts
setRecord
STOP


ProspectAccount
TRUE
IMMEDIATE ACTIONS
emailAlerts
Create task
show fewer
Add Action
STOP
FALSE
Negotiation
TRUE
IMMEDIATE ACTIONS
approval
show more
Add Action
STOP
FALSE
Close
TRUE
IMMEDIATE ACTIONS
STOP
Create task
Record Type*
Task
Set Field Values
Field*
Type*
Value*
Due Date Only
Formula
Today() + 7
Assigned To ID
Field Reference
[Opportunity].Acco...
Priority
Picklist
High
Status
Picklist
In Progress
Subject
String
Send Marketing Materia
Related To ID
Field Reference
[Opportunity].Id
Add Row

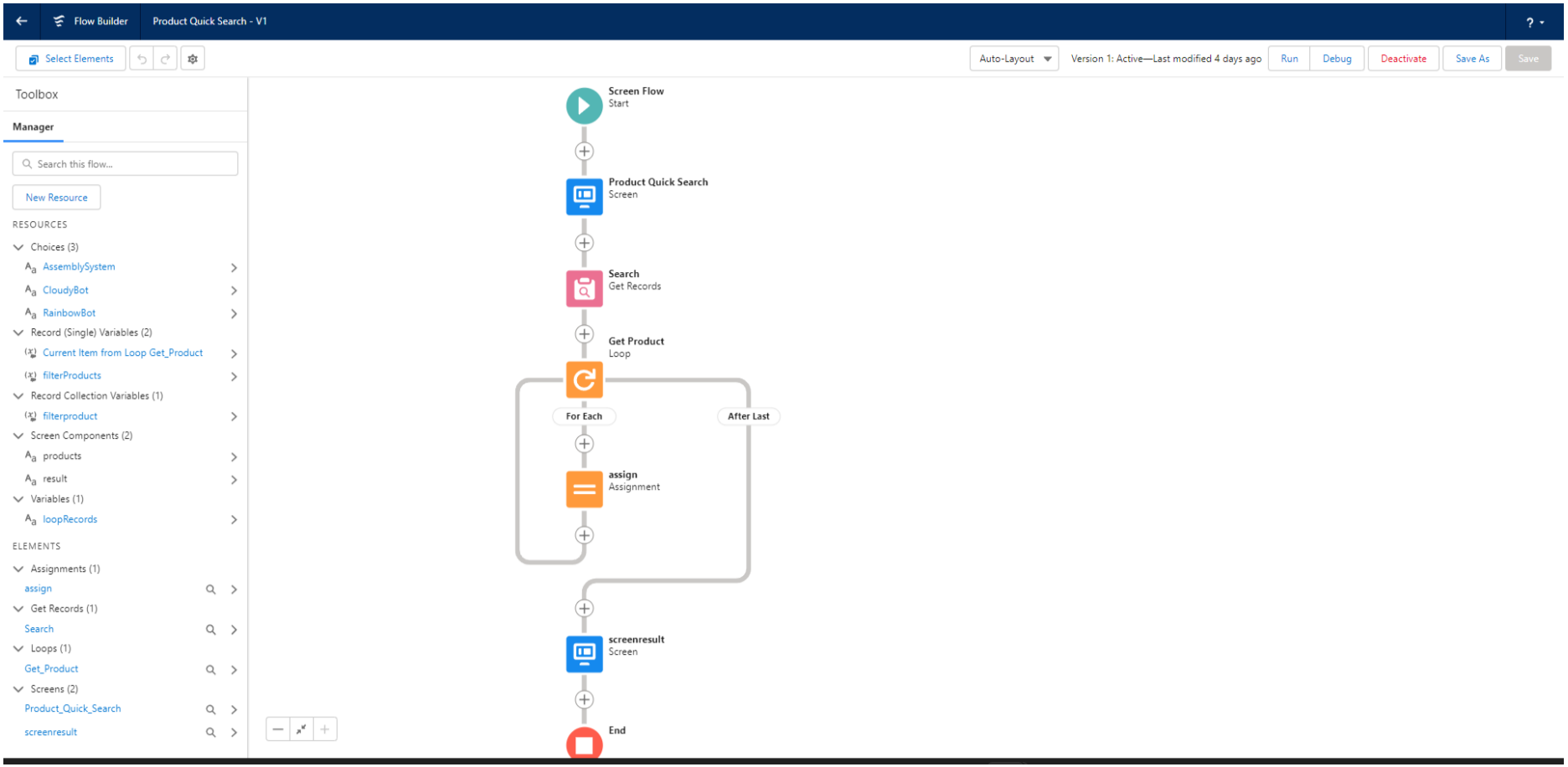

show fewer
Add Action
Negotiation
TRUE
IMMEDIATE ACTIONS
approval
updateStage
show fewer
Add Action
STOP
FALSE
Close
TRUE
IMMEDIATE ACTIONS
emailAlerts
STOP
Action Name*
approval
Object*
Opportunity
Approval Process*
Specific approval process
prospect - prospect
Skip the entry criteria for this process?
Yes
Submitter*
Current User

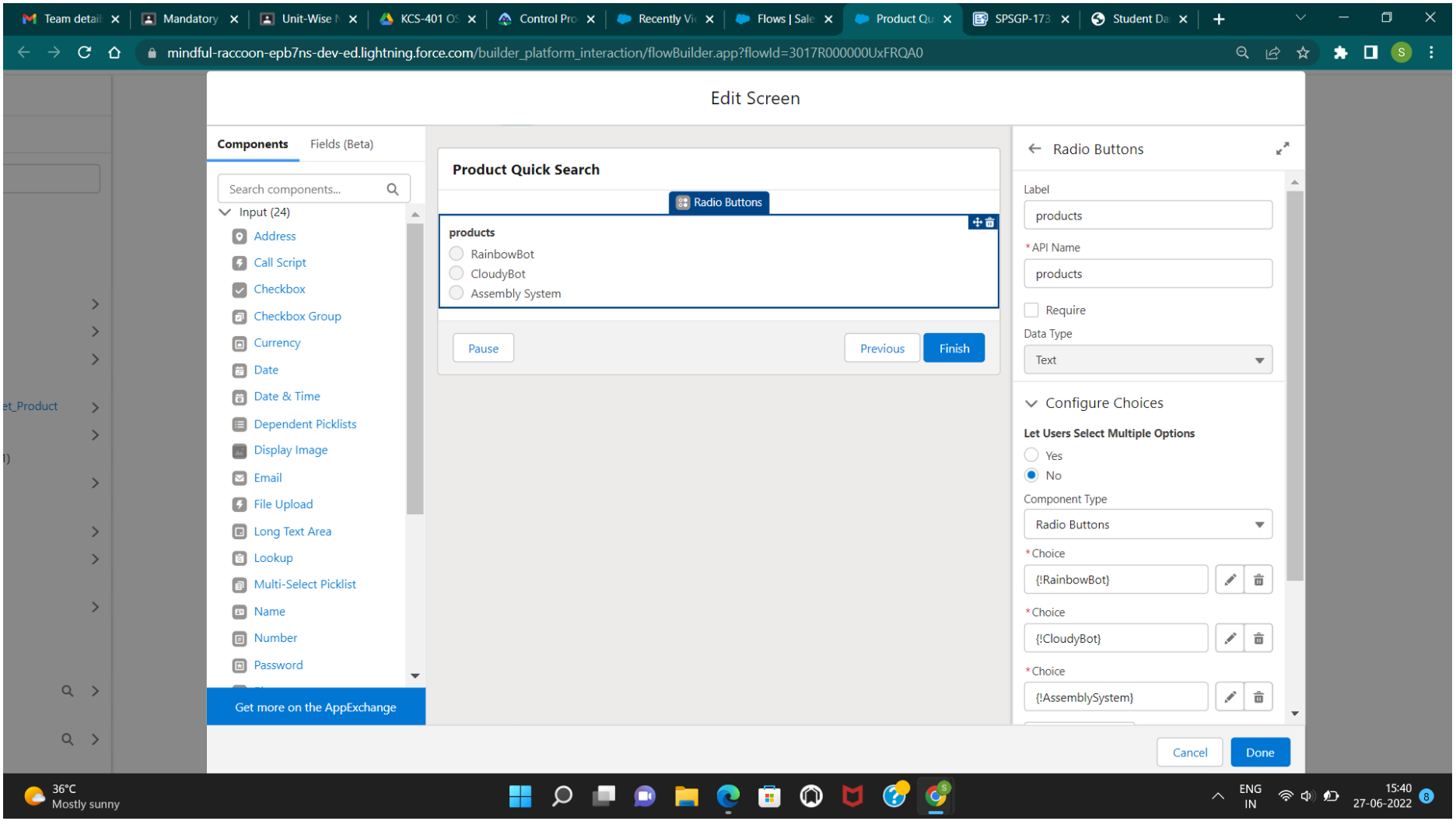

Negotiation
TRUE
IMMEDIATE ACTIONS
approval
updateStage
show fewer
Add Action
STOP
FALSE
Close
TRUE
IMMEDIATE ACTIONS
emailAlerts
STOP
updateStage
Record*
[Opportunity]
Criteria for Updating Records*
Updated records meet all conditions
No criteria—just update the records!
Set new field values for the records you update
Field*
Type*
Value*
Stage
Picklist
Awaiting Approval
Add Row



CHALLENGE 6: CREATE FLOW FOR OOPORTUNITIES:

A flow is created named PRODUCT QUICK SEARCH. And after creating it we use any template in Lightning app builder to save it


Flow Builder
Product Quick Search - V1
Select Elements
Auto-Layout
Version 1: Active—Last modified 4 days ago
Run
Debug
Deactivate
Save As
Save
Toolbox
Manager
Search this flow...
New Resource
RESOURCES
Choices (3)
AssemblySystem
CloudyBot
RainbowBot
Record (Single) Variables (2)
Current Item from Loop Get_Product
filterProducts
Record Collection Variables (1)
filterproduct
Screen Components (2)
products
result
Variables (1)
loopRecords
ELEMENTS
Assignments (1)
assign
Get Records (1)
Search
Loops (1)
Get_Product
Screens (2)
Product_Quick_Search
screenresult
Screen Flow
Start
Product Quick Search
Screen
Search
Get Records
Get Product
Loop
For Each
After Last
assign
Assignment
screenresult
Screen
End


Team detail
Mandatory
Unit-Wise
KCS-401 O
Control Pro
Recently Vi
Flows | Sale
Product Qu
SPSGP-173
Student Da
mindful-raccoon-epb7ns-dev-ed.lightning.force.com/builder_platform_interaction/flowBuilder.app?flowId=3017R000000UxFRQA0
Edit Screen
Components
Fields (Beta)
Search components...
Input (24)
Address
Call Script
Checkbox
Checkbox Group
Currency
Date
Date & Time
Dependent Picklists
Display Image
Email
File Upload
Long Text Area
Lookup
Multi-Select Picklist
Name
Number
Password
Get more on the AppExchange
Product Quick Search
Radio Buttons
products
RainbowBot
CloudyBot
Assembly System
Pause
Previous
Finish
Radio Buttons
Label
products
* API Name
products
Require
Data Type
Text
Configure Choices
Let Users Select Multiple Options
Yes
No
Component Type
Radio Buttons
* Choice
{!RainbowBot}
* Choice
{!CloudyBot}
* Choice
{!AssemblySystem}
Cancel
Done
36°C
Mostly sunny
ENG
IN
15:40
27-06-2022

Team detail | Mandatory | Unit-Wise | KCS-401 | Control Pro | Recently Vi | Flows | Sale | Product Qu | SPSGP-173 | Student Da

mindful-raccoon-epb7ns-dev-ed.lightning.force.com/builder_platform_interaction/flowBuilder.app?flowId=3017R000000UxFRQA0

Flow Builder | Product Quick Search - V1

Select Elements | 1: Active—Last modified 4 days ago | Run | Debug | Deactivate | Save As | Save

Toolbox

Manager

Search this flow...

New Resource

RESOURCES

- Choices (3)
  - AssemblySystem
  - CloudyBot
  - RainbowBot
- Record (Single) Variables (2)
  - Current Item from Loop Get_Product
  - filterProducts
- Record Collection Variables (1)
  - filterproduct
- Screen Components (2)
  - products
  - result
- Variables (1)
  - loopRecords

ELEMENTS

- Assignments (1)
  - assign
- Get Records (1)
  - Search
- Loops (1)
  - Get_Product
- Screens (2)
  - Product_Quick_Search
  - screenresult

Edit Get Records

Find Salesforce records and store their field values in flow variables.

Search (Search)

Get Records of This Object

* Object

Equipment

Filter Equipment Records

Condition Requirements

All Conditions Are Met (AND)

Field	Operator	Value
Name	Contains	products

+ Add Condition

Sort Equipment Records

Sort Order

Not Sorted

If you store only the first record, filter by a unique field, such as ID.

How Many Records to Store

- Only the first record
- All records

How to Store Record Data

- Automatically store all fields
- Choose fields and let Salesforce do the rest
- Choose fields and assign variables (advanced)

Cancel | Done

End

36°C Mostly sunny

ENG IN

15:40 27-06-2022

CHALLENGE 7: AUTOMATE SETUP:

In robot setup object => day of week field of formula type update formula:

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

And then in process builder=> OpportunityProcess update date field formula:

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

Superbadge

+13,000 POINTS

Apex Specialist

Use integration and business logic to push your Apex coding skills to the limit.

Completed 6/24/22

Module

+1,200 POINTS

Leads & Opportunities for Lightning Experience

Learn to power your sales process with leads and opportunities in Salesforce.

Completed 6/4/22

Superbadge

+10,000 POINTS

Process Automation Specialist

Showcase your mastery of business process automation without writing a line of code.

Completed 6/23/22

Completed 6/24/22

HENCE, BOTH OF MY SUPERBADGES(APEX SPECIALIST , PROCESS AUTOMATION SPECIALIST) ALONG WITH COMPLETE SALEFORCE DEVELOPER CATALYST MODULE ARE DONE .

HERE IS MY TRAILHEAD PLAYGROUND LINK:

https://trailblazer.me/id/ssingh3255

THANK YOU!