Project Title	:	Salesforce Developer Catalyst Self-Learning & Super Badges
Duration	:	30 Days
Team	:	ED
Mentor(s) Name	:	Sai Manikh

PROJECT DESCRIPTION

A project Report
On
Salesforce Developer Catalyst Self-Learning & Super Badges

INTRODUCTION

My name is Enjeti DivyaSree.I am glad to announce that I have successfully completed Developer super set as part of Salesforce supported virtual internship program.
Thanks to SmartInternz and Salesforce for providing this opportunity.
There are three roles in salesforce super set
1) Salesforce administrator
2) Salesforce developer
3) Salesforce consultant
I choose "SALESFORCE DEVELOPER" among these three roles and finished my super set badges successfully.

ACKNOWLEDGEMENT

I am thankful to my guide Sai Manikh for his valuable guidance and encouragement.His helping attitude and suggestios have helped in the successful completion of the project report.
I am also thankful to the team SmartInternz and Smart Bridge for providing this free internship program to the students.Thanks to the salesforce team for providing virtual classes to the students and by making understand about the program.

SALESFORCE:

Salesforce, Inc. is an American cloud-based software company headquartered in San Francisco, California. It provides customer relationship management (CRM) software and applications focused on sales, customer service, marketing automation, analytics, and application development.

SERVICES:

Salesforce's products include several customer relationship management (CRM) technologies, including: Sales Cloud,Service Cloud, Marketing Cloud, and Commerce Cloud and Platform.Additional technologies include Slack, MuleSoft, Tableau Analytics, and Trailhead.

SALESFORCE DEVELOPER:

Salesforce Developers manage and customize the Salesforce instance of a company by using three important technologies:

- **Lightning Component Framework**
- **Apex**
- **Visualforce**

Salesforce Developers use these tools to come up with custom apps and processes. Even though this is a technical job, it goes beyond programming. They have to collaborate with support, sales, and marketing to make sure the needs are met. Salesforce Developers are also responsible for QA, debugging, testing, and user documentation.

HOW TO BECOME A SALESFORCE DEVELOPER:

To become a successful Salesforce Developer, you need to specialize in the following areas:

- Lightning framework

- Data management, security, and modeling
- Apex object-oriented programming language
- Salesforce app customization
- Salesforce Object Query Language (SOQL)
- Salesforce Developer Console
- Visualforce basics
- Search solutions basics

SALESFORCE DEVELOPER SKILLS:

Salesforce Developer job descriptions these days are looking out for skills that are found in the same arenas where programmers and administrators have proved their expertise in. In the first place, certain Salesforce Developer soft skills are required for applicants to enter the Salesforce career. If you wish to join the race, you need to prove yourself proficient in the following soft skills:

- **Problem-solving:** A developer has to oversee all the stages of software development, and hence, they should be well-equipped with problem-solving skills.
- **Communication:** You have to be able to make your customers and your team understand what you want to communicate.
- **Analytical mindset:** You need to consistently analyze and understand the client's brief and design the software according to their needs.
- **Solutions-focused:** Your main objective has to be to find the solution to all the client's and your team's problems.

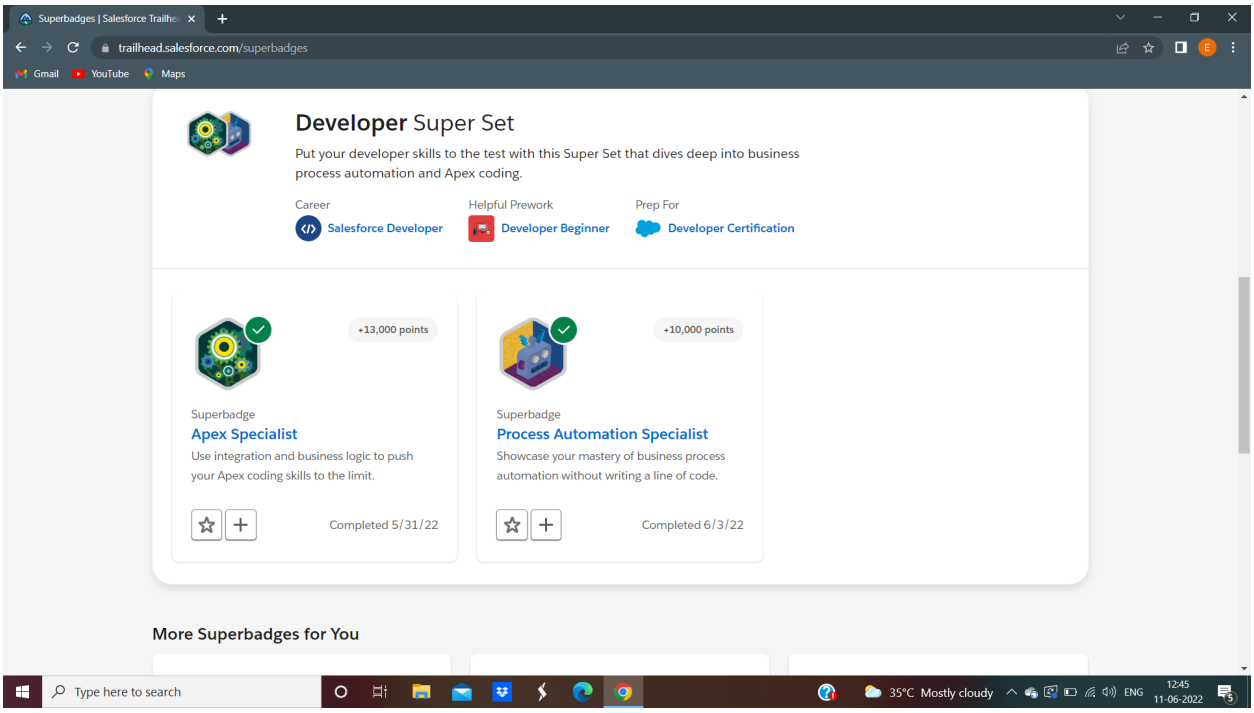
- **Project management:** You have to manage projects and make sure it is delivered in time. This also includes quality checks and client feedback.

ROLES OF A SALESFORCE DEVELOPER:

- Working on both mobile and website applications
- Developing products on the Force.com platform using Visualforce and Apex
- Integrating multiple systems with Salesforce
- Customizing the Salesforce environment
- Taking part in development, deployment, testing, training, and documentation

RESPONSIBILITIES OF DEVELOPER IN SALESFORCE:

- Meeting Project Managers to determine the **CRM** needs
- Testing and implementing applications
- Creating customer workflows
- Maintaining user roles and security



I completed my Developer Super Set.This includes

1) Apex Specialist

2)Process Automation Specialist

Apex Specialist Superbadge:

In this superbadge, initial step is to create a new playground. Now the steps which are mentioned in 'set up development org' has to be done. Then according to the given process, write the code for each step mentioned below:

Step 1 : Answering the multiple choice questions.

CHALLENGE 1: AUTOMATE RECORD CREATION

MaintenanceRequestHelper

```
1  public with sharing class MaintenanceRequestHelper {
2      public static void updateworkOrders(List<Case> updWorkOrders,
   Map<Id,Case> nonUpdCaseMap) {
3          Set<Id> validIds = new Set<Id>();
4
5
6          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);



                }
            }
        }

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

                newCases.add(nc);
            }

            insert newCases;

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

                }
            }
            insert ClonedWPs;
        }
    }
}
```

Maintainance Request.apxt

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

```
2  // ToDo: Call MaintenanceRequestHelper.updateWorkOrders
3  if(Trigger.isAfter)
4  MaintenanceRequestHelper.updateWorkOrders(Trigger.New);
5  }
```

CHALLENGE 2:SYNCHRONIZE SALESFORCE DATA WITH AN EXTERNAL SYSTEM

SOLUTION:

- Setup -> Search in quick find box -> click Remote Site Settings -> Name = Warehouse URL , Remote Site URL = https://th-superbadge-apex.herokuapp.com , make sure active is selected.
- Go to the developer console use below code

WarehouseCalloutService.apxc :-

```
1  public with sharing class WarehouseCalloutService {
2  private static final String WAREHOUSE_URL = 'https://th-

3  @future(callout=true)
4  public static void runWarehouseEquipmentSync() {
5  //ToDo: complete this method to make the callout (using @future)
   to the
6  //      REST endpoint and update equipment on hand.
7  HttpResponse response = getResponse();
8  if(response.getStatusCode() == 200)
9  {
10 List<Product2> results = getProductList(response); //get list of
   products from Http callout response
11 if(results.size() >0)
12 upsert results Warehouse_SKU__c; //Upsert the products in your
   org based on the external ID SKU
13 }
14 }
15 //Get the product list from the external link
16 public static List<Product2> getProductList(HttpResponse
   response)
17 {
18 List<Object> externalProducts = (List<Object>)
   JSON.deserializeUntyped(response.getBody()); //desrialize the
```

```
    json response
19 List<Product2> newProducts = new List<Product2>();
20 for(Object p : externalProducts)
21 {
22 Map<String, Object> productMap = (Map<String, Object>) p;
23 Product2 pr = new Product2();
24 //Map the fields in the response to the appropriate fields in the
   Equipment object
25 pr.Replacement_Part__c = (Boolean)productMap.get('replacement');
26 pr.Cost__c = (Integer)productMap.get('cost');
27 pr.Current_Inventory__c = (Integer)productMap.get('quantity');
28 pr.Lifespan_Months__c = (Integer)productMap.get('lifespan') ;
29 pr.Maintenance_Cycle__c =
   (Integer)productMap.get('maintenanceperiod');
30 pr.Warehouse_SKU__c = (String)productMap.get('sku');
31 pr.ProductCode = (String)productMap.get('_id');
32 pr.Name = (String)productMap.get('name');
33 newProducts.add(pr);
34 }
35 return newProducts;
36 }
37 // Send Http GET request and receive Http response
38 public static HttpResponse getResponse() {
39 Http http = new Http();
40 HttpRequest request = new HttpRequest();
41 request.setEndpoint(WAREHOUSE_URL);
42 request.setMethod('GET');
43 HttpResponse response = http.send(request);
44 return response;
45 }
46 }
```

After saving the code open execute anonymous window (CTRl+E) and run this method ,

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

CHALLENGE 3:SCHEDULE SYNCHRONIZATION

Edit the following code in developer console

```
global  class WarehouseSyncSchedule implements Schedulable{
// implement scheduled code here
global  void execute (SchedulableContext sc){
WarehouseCalloutService.runWarehouseEquipmentSync();
//optional this can be done by debug mode
String sch = '00 00 01 * * ?';//on 1 pm
System.schedule('WarehouseSyncScheduleTest', sch, new
WarehouseSyncSchedule());
}
}
```

Execute in anonymous window

```
WarehouseSyncSchedule scheduleInventoryCheck();
```

CHALLENGE 4: TEST AUTOMATION LOGIC

- Go to the developer console use below code :

MaintenanceRequestHelperTest.apx

```
@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

    private static final string REQUEST_SUBJECT = 'Testing


    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;
```

```
50          id equipmentId = equipment.Id;
51
52          case somethingToUpdate =
    createMaintenanceRequest(vehicleId,equipmentId);
53          insert somethingToUpdate;
54
55          Equipment_Maintenance_Item__c workP =
    createWorkPart(equipmentId,somethingToUpdate.id);
56          insert workP;
57
58          test.startTest();
59          somethingToUpdate.status = CLOSED;
60          update somethingToUpdate;
61          test.stopTest();
62
63          Case newReq = [Select id, subject, type, Equipment__c,
    Date_Reported__c, Vehicle__c, Date_Due__c
64                         from case
65                         where status =:STATUS_NEW];
66
67          Equipment_Maintenance_Item__c workPart = [select id
68                                                    from
    Equipment_Maintenance_Item__c
69                                                    where
    Maintenance_Request__c =:newReq.Id];
70
71          system.assert(workPart != null);
72          system.assert(newReq.Subject != null);
73          system.assertEquals(newReq.Type, REQUEST_TYPE);
74          SYSTEM.assertEquals(newReq.Equipment__c, equipmentId);
75          SYSTEM.assertEquals(newReq.Vehicle__c, vehicleId);
76          SYSTEM.assertEquals(newReq.Date_Reported__c,
    system.today());
77      }
78
79      @istest
80      private static void testMaintenanceRequestNegative(){
81          Vehicle__C vehicle = createVehicle();
82          insert vehicle;
83          id vehicleId = vehicle.Id;
```

```
84
85          product2 equipment = createEq();
86          insert equipment;
87          id equipmentId = equipment.Id;
88
89          case emptyReq =
    createMaintenanceRequest(vehicleId,equipmentId);
90          insert emptyReq;
91
92          Equipment_Maintenance_Item__c workP =
    createWorkPart(equipmentId, emptyReq.Id);
93          insert workP;
94
95          test.startTest();
96          emptyReq.Status = WORKING;
97          update emptyReq;
98          test.stopTest();
99
100          list<case> allRequest = [select id
101                                   from case];
102
103          Equipment_Maintenance_Item__c workPart = [select id
104                                                    from
    Equipment_Maintenance_Item__c
105                                                    where
    Maintenance_Request__c = :emptyReq.Id];
106
107          system.assert(workPart != null);
108          system.assert(allRequest.size() == 1);
109      }
110
111      @istest
112      private static void testMaintenanceRequestBulk(){
113          list<Vehicle__C> vehicleList = new list<Vehicle__C>();
114          list<Product2> equipmentList = new list<Product2>();
115          list<Equipment_Maintenance_Item__c> workPartList = new
    list<Equipment_Maintenance_Item__c>();
116          list<case> requestList = new list<case>();
117          list<id> oldRequestIds = new list<id>();
118
```

```
119         for(integer i = 0; i < 300; i++){
120            vehicleList.add(createVehicle());
121             equipmentList.add(createEq());
122         }
123         insert vehicleList;
124         insert equipmentList;
125 
126         for(integer i = 0; i < 300; i++){
127 
   requestList.add(createMaintenanceRequest(vehicleList.get(i).id,
   equipmentList.get(i).id));
128         }
129         insert requestList;
130 
131         for(integer i = 0; i < 300; i++){
132 
   workPartList.add(createWorkPart(equipmentList.get(i).id,
   requestList.get(i).id));
133         }
134         insert workPartList;
135 
136         test.startTest();
137         for(case req : requestList){
138             req.Status = CLOSED;
139             oldRequestIds.add(req.Id);
140         }
141         update requestList;
142         test.stopTest();
143 
144         list<case> allRequests = [select id
145                                   from case
146                                   where status =: STATUS_NEW];
147 
148         list<Equipment_Maintenance_Item__c> workParts = [select
   id
149                                                          from
   Equipment_Maintenance_Item__c
150                                                          where
   Maintenance_Request__c in: oldRequestIds];
151 
```

```
        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

                    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];
```

```
23
24          for (AggregateResult ar : results){
25              maintenanceCycles.put((Id)
    ar.get('Maintenance_Request__c'), (Decimal) ar.get('cycle'));
26          }
27
28              for(Case cc : closedCasesM.values()){
29                  Case nc = new Case (
30                      ParentId = cc.Id,
31                  Status = 'New',
32                      Subject = 'Routine Maintenance',
33                      Type = 'Routine Maintenance',
34                      Vehicle__c = cc.Vehicle__c,
35                      Equipment__c =cc.Equipment__c,
36                      Origin = 'Web',
37                      Date_Reported__c = Date.Today()
38
39                  );
40
41                  If (maintenanceCycles.containskey(cc.Id)){
42                      nc.Date_Due__c =
    Date.today().addDays((Integer) maintenanceCycles.get(cc.Id));
43                  }
44
45                  newCases.add(nc);
46              }
47
48              insert newCases;
49
50              List<Equipment_Maintenance_Item__c> clonedWPs = new
    List<Equipment_Maintenance_Item__c>();
51              for (Case nc : newCases){
52                  for (Equipment_Maintenance_Item__c wp :
    closedCasesM.get(nc.ParentId).Equipment_Maintenance_Items__r){
53                      Equipment_Maintenance_Item__c wpClone =
    wp.clone();
54                      wpClone.Maintenance_Request__c = nc.Id;
55                      ClonedWPs.add(wpClone);
56
57                  }
```

```
58              }
59              insert ClonedWPs;
60          }
61      }
62 }
```

MaintenanceRequest.apxt:

```
1  trigger MaintenanceRequest on Case (before update, after update) {
2      if(Trigger.isUpdate && Trigger.isAfter){
3          MaintenanceRequestHelper.updateWorkOrders(Trigger.New,
   Trigger.OldMap);
4      }
5  }
```

run all and check the challenge

CHALLENGE 5: TEST CALLOUT LOGIC

Modify the below apex classes ,save and run all

```
1 @IsTest
2 private class WarehouseCalloutServiceTest {
3 // implement your mock callout test here
4 @isTest
5 static void testWareHouseCallout(){
6 Test.setMock(HttpCalloutMock.class, new
   WarehouseCalloutServiceMock());
7 WarehouseCalloutService.runWarehouseEquipmentSync();
8 }
9 }
```

```
1  @isTest
2  public class WarehouseCalloutServiceMock implements
   HTTPCalloutMock {
3  // implement http mock callout
4  public HTTPResponse respond (HttpRequest request){
5  HttpResponse response = new HTTPResponse();
6  response.setHeader('Content-type','application/json');
```

```
7  response.setBody('[{"_id":"55d66226726b611100aaf741","replacement
8  response.setStatusCode(200);
9  return response;
10 }
11 }
```

Now run all and check the challenge.

CHALLENGE 6: TEST SCHEDULING LOGIC

Modify the code as below in the developer console

WarehouseSyncSchedule.apxc :-

```
1global class WarehouseSyncSchedule implements Schedulable {
2    global void execute(SchedulableContext ctx) {
3
4        WarehouseCalloutService.runWarehouseEquipmentSync();
5    }
6}
```

WarehouseSyncScheduleTest.apxc :-

```
1@isTest
2public class WarehouseSyncScheduleTest {
3
4    @isTest static void WarehousescheduleTest(){
5        String scheduleTime = '00 00 01 * * ?';
6        Test.startTest();
7        Test.setMock(HttpCalloutMock.class, new
   WarehouseCalloutServiceMock());
```

```
8         String jobID=System.schedule('Warehouse Time To Schedule to

9         Test.stopTest();
10          //Contains schedule information for a scheduled job.
   CronTrigger is similar to a cron job on UNIX systems.
11          // This object is available in API version 17.0 and
   later.
12          CronTrigger a=[SELECT Id FROM CronTrigger where
   NextFireTime > today];
13          System.assertEquals(jobID, a.Id,'Schedule ');
14
15
16      }
17 }
```

Run all and check the challenge.

Process Automation Specialist Super Badge

In this super badge we can do the following things

1. Automate lead ownership using assignment rules
2. Enforce data integrity with formula fields and validation rules
3. Create a custom object in a master-detail relationship to a standard object
4. Define an opportunity sales process using stages, record types, and validation rules
5. Automate business processes to send emails, create related records, and submit opportunities for approval
6. Create a flow to display dynamic information on a Lightning record page
7. Create a process to evaluate and update records

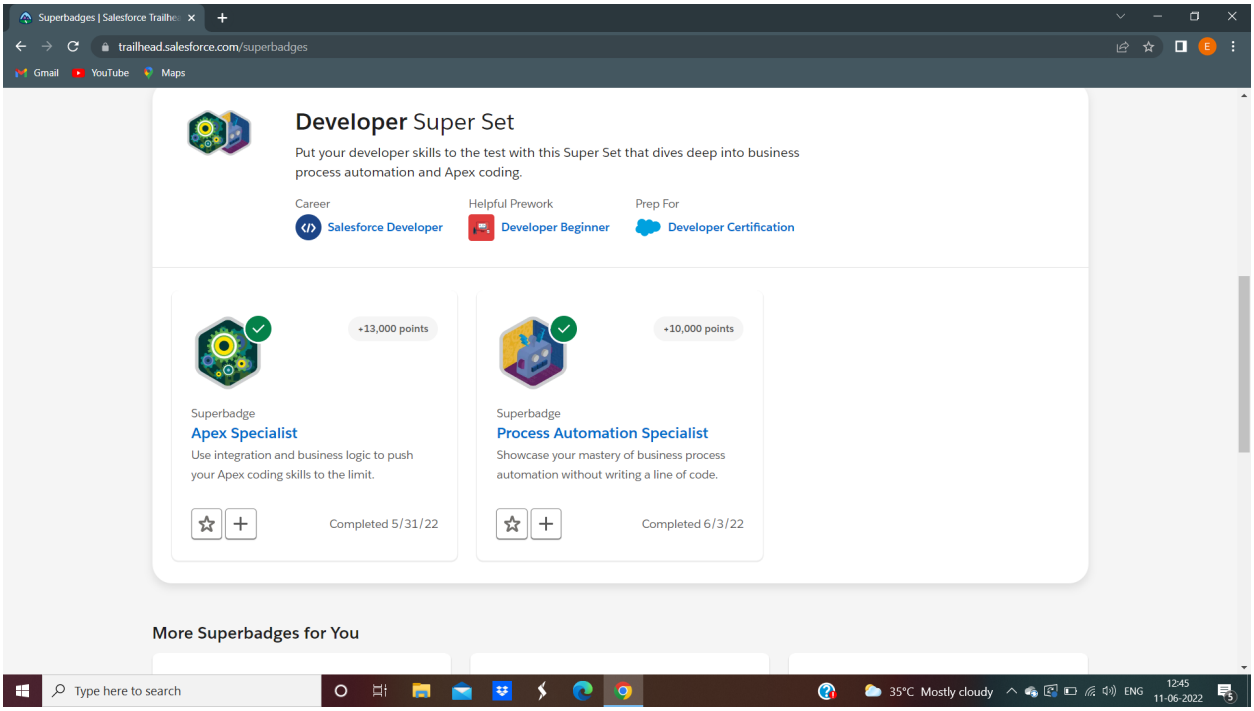
Steps in Process Automation Specialist Super Badge is

1. Challenge 1 : Automate Leads
2. Challenge 2 : Automate Accounts
3. Challenge 3 : Create Robot setup Object
4. Challenge 4 : Create Sales Process and Validate Opportunities
5. Challenge 5 : Automate opportunities

6. Challenge 6 : Create Flow for Opportunities
7. Challenge 7 : Automate Setups

After completing all these challenges,the process automation specialist super badge will be completed successfully.

CONCLUSION



Thanks to SmartInternz,Salesforce and the Smart Bridge for providing this Opportunity.