I Have completed developer catalyst in trialhead and I have finished 2 superbadges they are
1.APEX SPECIALIST
2.Process Automation Specialist
so in this pdf iam going to mention all the codes related apex specialist super badge and apex modules

APEX SPECIALIST

1-Automate record creation

MaintenanceRequestHelper

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


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 newCases = new List();
 Map closedCasesM = new Map([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 maintenanceCycles = new Map(); 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 clonedWPs = new List();
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;
       }
     }
}
```

MaitenanceRequest

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

2. Synchronize Salesforce data with an external system

WarehouseCalloutService

```
public with sharing class WarehouseCalloutService implements Queueable {
 private static final String WAREHOUSE_URL = 'https://th-
superbadgeapex.herokuapp.com/equipment';
@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);
 List warehouseEq = new List();

 if (response.getStatusCode() == 200){ List jsonResponse =
(List)JSON.deserializeUntyped(response.getBody());
System.debug(response.getBody());
for (Object eq : jsonResponse){ Map mapJson = (Map)eq;
Product2 myEq = new Product2();
 myEq.Replacement_Part__c = (Boolean) mapJson.get('replacement');
myEq.Name = (String) mapJson.get('name');
myEq.Maintenance_Cycle__c = (Integer) mapJson.get('maintenanceperiod');
myEq.Lifespan_Months__c = (Integer) mapJson.get('lifespan');
 myEq.Cost__c = (Integer) mapJson.get('cost');
 myEq.Warehouse_SKU__c = (String) mapJson.get('sku');
myEq.Current_Inventory__c = (Double) mapJson.get('quantity');
myEq.ProductCode = (String) mapJson.get('_id');
warehouseEq.add(myEq);
}
 if (warehouseEq.size() > 0){
 upsert warehouseEq;
System.debug('Your equipment was synced with the warehouse one');
     }
   }
 }
public static void execute (QueueableContext context){
runWarehouseEquipmentSync();     }
 }
```

3.Schedule synchronization

WarehouseSyncShedule

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

4.Test automation logic

MaintenanceRequestHelperTest

```
 @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 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 vehicleList = new list(); list equipmentList = new list();
 list workPartList = new list();
 list requestList = new list();
list oldRequestIds = new list();
```

```
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 allRequests = [select id from case where status =: STATUS_NEW];
list workParts = [select id from Equipment_Maintenance_Item__c where
Maintenance_Request__c in: oldRequestIds];
system.assert(allRequests.size() == 300);
 }
}
```

MaintenanceRequestHelper

```
 public with sharing class MaintenanceRequestHelper {
public static void updateworkOrders(List updWorkOrders, Map nonUpdCaseMap) {
Set validIds = new Set();
 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 newCases = new List();
Map closedCasesM = new Map([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 maintenanceCycles = new Map();
 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 clonedWPs = new List();
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 trigger

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

5.Test callout logic

WarehouseCalloutService

```
 public with sharing class WarehouseCalloutService {
private static final String WAREHOUSE_URL = 'https://th-
superbadgeapex.herokuapp.com/equipment';
public static void runWarehouseEquipmentSync(){
 Http http = new Http();
 HttpRequest request = new HttpRequest();
request.setEndpoint(WAREHOUSE_URL);
request.setMethod('GET');
 HttpResponse response = http.send(request);
List warehouseEq = new List();
 if (response.getStatusCode() == 200){ List jsonResponse =
(List)JSON.deserializeUntyped(response.getBody());
System.debug(response.getBody());
for (Object eq : jsonResponse){ Map mapJson = (Map)eq;
Product2 myEq = new Product2();
myEq.Replacement_Part__c = (Boolean) mapJson.get('replacement');
myEq.Name = (String) mapJson.get('name');
myEq.Maintenance_Cycle__c = (Integer) mapJson.get('maintenanceperiod');
myEq.Lifespan_Months__c = (Integer) mapJson.get('lifespan'); myEq.Cost__c =
(Decimal) mapJson.get('lifespan');
 myEq.Warehouse_SKU__c = (String) mapJson.get('sku');
myEq.Current_Inventory__c = (Double) mapJson.get('quantity');
warehouseEq.add(myEq);

}
if (warehouseEq.size() > 0){
```

```
upsert warehouseEq;
System.debug('Your equipment was synced with the warehouse one');
System.debug(warehouseEq);
        }
      }
   }
 }
```

WarehouseCalloutServiceTest

```
 @isTest
 private class WarehouseCalloutServiceTest {
@isTest
 static void testWareHouseCallout(){
 Test.startTest();
 Test.setMock(HTTPCalloutMock.class, new WarehouseCalloutServiceMock());
WarehouseCalloutService.runWarehouseEquipmentSync();
Test.stopTest();
System.assertEquals(1, [SELECT count() FROM Product2]);
    }
 }
```

WarehouseCalloutServiceMock

```
 @isTest
 global class WarehouseCalloutServiceMock implements HttpCalloutMock {
 global static HttpResponse respond(HttpRequest request){
System.assertEquals('https://th-superbadge-apex.herokuapp.com/equipment',
request.getEndpoint());
 System.assertEquals('GET', request.getMethod());
 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"}]');
response.setStatusCode(200);
return response;
```

```
    }
}
```

6.Test scheduling logic

WarehouseSyncSchedule

```
 global class WarehouseSyncSchedule implements Schedulable {
global void execute(SchedulableContext ctx) {
WarehouseCalloutService.runWarehouseEquipmentSync();

   }
}
```

WarehouseSyncScheduleTest

```
@isTest

 public class WarehouseSyncScheduleTest {

@isTest

static void WarehousescheduleTest(){
String scheduleTime = '00 00 01 * * ?'; Test.startTest();
Test.setMock(HttpCalloutMock.class, new WarehouseCalloutServiceMock());
 String jobID=System.schedule('Warehouse Time To Schedule to Test', scheduleTime,
new WarehouseSyncSchedule());
 Test.stopTest();
CronTrigger a=[SELECT Id FROM CronTrigger where NextFireTime > today];
System.assertEquals(jobID, a.Id,'Schedule ');

  }
}
```

HERE IT STARTS WITH APEX MODULES

Apex Triggers

1.
Get Started with Apex Triggers

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

2.
Bulk Apex Triggers

```
 trigger ClosedOpportunityTrigger on Opportunity (after insert,after update) {
  List tasklist = new List();
for(Opportunity opp : Trigger.New){
   if(opp.StageName == 'Closed Won'){
      taskList.add(new Task(subject ='Follow Up Test Task',WhatId = opp.Id));
     }
 }
  if(taskList.size()>0){
       insert taskList;
   }
}
```

Apex Testing

1. Get Started with Apex Unit Tests

```
 public class VerifyDate { public static Date CheckDates(Date date1, Date date2) {
if(DateWithin30Days(date1,date2)) {
return date2;
}
 else
 {
return SetEndOfMonthDate(date1);
  }
 }
private static Boolean DateWithin30Days(Date date1, Date date2) {
 if( date2 < date1) { return false;
}
 Date date30Days = date1.addDays(30);
//create a date 30 days away from date1 if( date2 >= date30Days ) {
return false;
}
else
 {
 return true;
    }
}
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;
  }
}
```

2.

Test Apex Triggers

```
 trigger RestrictContactByName on Contact (before insert, before update) {
For (Contact c : Trigger.New) {
if(c.LastName == 'INVALIDNAME')
 {
 c.AddError('The Last Name "'+c.LastName+'" is not allowed for DML');
     }
   }
 }
```

3.

Create Test Data for Apex Tests

```
 public class RandomContactFactory {
  public static List generateRandomContacts(Integer num,String lastName){
List contactList=new List();
   for(Integer i=1;i<=num;i++){
            Contact ct=new Contact(FirstName='Test'+i,LastName=lastName);
contactList.add(ct);
        }
    return contactList;
    }
 }
```

Asynchronous Apex

1.

Use Future Methods

```
 public class AccountProcessor {
    @future     public static void countContacts(List accountIds){ List accList = [Select Id,
Number_Of_Contacts__c, (Select Id from Contacts) from Account where Id in
:accountIds];
for(Account acc : accList){
       acc.Number_Of_Contacts__c = acc.Contacts.size();
        }
      update accList;
   }
 }
```

test class

```
 @isTest
public class AccountProcessorTest {
public static testmethod void testAccountProcessor(){
       Account a = new Account();
```

```
        a.Name = 'Test Account';
     insert a;
       Contact con = new Contact();
       con.FirstName = 'Binary';
    con.LastName = 'Programming';
      con.AccountId = a.Id;
        insert con;
     List accListId = new List();
       accListId.add(a.Id);
     Test.startTest();
    AccountProcessor.countContacts(accListId);
     Test.stopTest();
        Account acc = [Select Number_Of_Contacts__c from Account where Id =: a.Id];
System.assertEquals(Integer.valueof(acc.Number_Of_Contacts__c),1);
  }
 }
```

2.

Use Batch Apex

```
 global class LeadProcessor implements Database.Batchable, Database.Stateful {
global Integer recordsProcessed = 0;
   global Database.QueryLocator start(Database.BatchableContext bc) {
     return Database.getQueryLocator([SELECT ID, LeadSource FROM Lead]);
    }
  global void execute(Database.BatchableContext bc, List scope) {
      for (Lead lead : scope) {
        lead.LeadSource = 'Dreamforce';
            recordsProcessed = recordsProcessed + 1;
System.debug(lead.LeadSource);
      }
        update scope;
   }
    global void finish(Database.BatchableContext bc){
System.debug(recordsProcessed + ' records processed. Shazam!');

    }
}
```

test class

```
 @isTest
 private class LeadProcessorTest {
     @TestSetup     static void setup(){
       List leads = new List();
         for (Integer i = 0; i < 200; i++) {
          leads.add(new Lead(LastName='Lead ' + i, Company='Company Number ' + i,
Status='Open - Not Contacted'));
       }
     insert leads;
    }
     static testMethod void test() {
         Test.startTest();
       LeadProcessor lp = new LeadProcessor();
       Id batchId = Database.executeBatch(lp);
        Test.stopTest();
            System.assertEquals(200, [select count() from lead where LeadSource =
'Dreamforce']);
     }
 }
```

3.
 Control Processes with Queueable Apex

```
 public class AddPrimaryContact implements Queueable {
    public contact c;
  public String state;
   public AddPrimaryContact(Contact c, String state) {
       this.c = c;
         this.state = state;
    }
 public void execute(QueueableContext qc) {
     system.debug('this.c = '+this.c+' this.state = '+this.state);
```

```
      List acc_lst = new List([select id, name, BillingState from account where
account.BillingState = :this.state limit 200]);
      List c_lst = new List();
        for(account a: acc_lst) {
        contact c = new contact();
           c = this.c.clone(false, false, false, false);
           c.AccountId = a.Id;
             c_lst.add(c);
        }
  insert c_lst;
   }
```

test class

```
 @IsTest
public class AddPrimaryContactTest {
 @IsTest
 public static void testing() {
         List acc_lst = new List();
    for (Integer i=0; i<50;i++) {
            account a = new account(name=string.valueOf(i),billingstate='NY');
system.debug('account a = '+a);
          acc_lst.add(a);
    }
        for (Integer i=0; i<50;i++) {
           account a = new account(name=string.valueOf(50+i),billingstate='CA');
system.debug('account a = '+a);
  acc_lst.add(a);
   }
        insert acc_lst;
      Test.startTest();
         contact c = new contact(lastname='alex');
     AddPrimaryContact apc = new AddPrimaryContact(c,'CA');
  system.debug('apc = '+apc);
        System.enqueueJob(apc);
       Test.stopTest();
       List c_lst = new List([select id from contact]);
       Integer size = c_lst.size();
        system.assertEquals(50, size);
        }
}
```

4.

Schedule Jobs Using the Apex Scheduler

```
 global class DailyLeadProcessor implements Schedulable{
 global void execute(SchedulableContext ctx){
     List leads = [SELECT Id, LeadSource FROM Lead WHERE LeadSource = ''];
if(leads.size() > 0){
           List newLeads = new List();
        for(Lead lead : leads){
           lead.LeadSource = 'DreamForce';
            newLeads.add(lead);
        }
   update newLeads;
       }
    }
 }
```

test class

```
 @isTest

 private class DailyLeadProcessorTest{
     public static String CRON_EXP = '0 0 0 2 6 ? 2022';
     static testmethod void testScheduledJob(){
List leads = new List();
      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();
```

```
        String jobId = System.schedule('Update LeadSource to DreamForce', CRON_EXP,
new DailyLeadProcessor());
      Test.stopTest();
     }
 }
```

Apex Integration Services

1.

Apex REST Callouts

```
 public class AnimalLocator{
  public static String getAnimalNameById(Integer x){
      Http http = new Http();
        HttpRequest req = new HttpRequest();
     req.setEndpoint('https://th-apex-http-callout.herokuapp.com/animals/' + x);
req.setMethod('GET');
        Map animal= new Map();
     HttpResponse res = http.send(req);
          if (res.getStatusCode() == 200) {
         Map results = (Map)JSON.deserializeUntyped(res.getBody());
    animal = (Map) results.get('animal');
     }
 return (String)animal.get('name');
   }
 }
```

test class

```
@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 class

```
@isTest
global class AnimalLocatorMock implements HttpCalloutMock {
       global HTTPResponse respond(HTTPRequest request) {
           HttpResponse response = new HttpResponse();
response.setHeader('Content-Type', 'application/json');
response.setBody('{"animals": ["majestic badger", "fluffy bunny", "scary bear", "chicken",
"mighty moose"]}');
       response.setStatusCode(200);
     return response;
   }
}
```

2.
Apex SOAP Callouts
parklocater class

```
public class ParkLocator {
 public static string[] country(string theCountry) {
     ParkService.ParksImplPort  parkSvc = new  ParkService.ParksImplPort();
// remove space        return parkSvc.byCountry(theCountry);
    }
}
park service class public class parkService {
     public class byCountryResponse {
      public String[] return_x;
        private String[] return_x_type_info = new
String[]{'return','http://parks.services/',null,'0','-1','false'};
     private String[] apex_schema_type_info = new
String[]{'http://parks.services/','false','false'};
         private String[] field_order_type_info = new String[]{'return_x'};
   }
 public class byCountry {
  public String arg0;
      private String[] arg0_type_info = new
String[]{'arg0','http://parks.services/',null,'0','1','false'};
         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-
service.herokuapp.com/service/parks';
      public Map inputHttpHeaders_x;
      public Map 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 response_map_x = new Map();
         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;
     }
     }
 }
```

test class

```
@isTest
private class ParkLocatorTest {
 @isTest
static void testCallout() {
                Test.setMock(WebServiceMock.class, new ParkServiceMock ());
     String country = 'United States';
   List result = ParkLocator.country(country);
 List parks = new List{'Yellowstone', 'Mackinac National Park', 'Yosemite'};
System.assertEquals(parks, result);
 }
}
```

mock class

```
@isTest
global class ParkServiceMock implements WebServiceMock {
  global void doInvoke(          Object stub,          Object request,          Map response,
String endpoint,          String soapAction,          String requestName,          String
responseNS,          String responseName,          String responseType) {
ParkService.byCountryResponse response_x = new ParkService.byCountryResponse();
response_x.return_x = new List{'Yellowstone', 'Mackinac National Park', 'Yosemite'};
response.put('response_x', response_x);
  }
}
```

3.

Apex Web Services

```
@RestResource(urlMapping='/Accounts/*/contacts') global class AccountManager {
@HttpGet    global static Account getAccount() {
    RestRequest req = RestContext.request;
  String accId = req.requestURI.substringBetween('Accounts/', '/contacts');
    Account acc = [SELECT Id, Name, (SELECT Id, Name FROM Contacts)
FROM Account WHERE Id = :accId];
     return acc;
   }
 }
```

test class

```
@isTest private class AccountManagerTest {
   private static testMethod void getAccountTest1() {
   Id recordId = createTestRecord();
      RestRequest request = new RestRequest();
        request.requestUri = 'https://na1.salesforce.com/services/apexrest/Accounts/'+
recordId +'/contacts' ;
       request.httpMethod = 'GET';
       RestContext.request = request;
       Account thisAccount = AccountManager.getAccount();
System.assert(thisAccount != null);
      System.assertEquals('Test record', thisAccount.Name);
  }
   static Id createTestRecord() {
     Account TestAcc = new Account(
       Name='Test record');
         insert TestAcc;
    Contact TestCon= new Contact(
     LastName='Test',
         AccountId = TestAcc.id);
       return TestAcc.Id;
     }
 }
```