SPSGP-15895-Salesforce Developer Catalyst
Self-Learning & Super Badges
1
APEX SPECIALIST SUPER BADGE CODES
APEX TRIGGERS
AccountAddressTrigger.axpt:

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

ClosedOpportunityTrigger.axpt:

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

APEX TESTING
VerifyData.apxc:

```
public class VerifyDate {
public static Date CheckDates(Date date1, Date date2) {
if(DateWithin30Days(date1,date2)) {
return date2;
} else {
return SetEndOfMonthDate(date1);
}
}
@TestVisible private static Boolean DateWithin30Days(Date date1,
```

```
40 Date date2) {
41 37 //check for date2 being in the past
42 38 if( date2 < date1) { return false; }
43 39 SPSGP-10445-Salesforce Developer Catalyst
44 40 Self-Learning & Super Badges
45 41 2
46 42 APEX SPECIALIST SUPER BADGE CODES
47 43 //check that date2 is within (>=) 30 days of date1
48 44 Date date30Days = date1.addDays(30); //create a date 30 days
   away
49 from date1
50 45 if( date2 >= date30Days ) { return false; }
51 46 else { return true; }
52 47 }
53 48 //method to return the end of the month of a given date
54 49 @TestVisible private static Date SetEndOfMonthDate(Date date1)
   {
55 50 Integer totalDays = Date.daysInMonth(date1.year(),
56 date1.month());
57 51 Date lastDay = Date.newInstance(date1.year(), date1.month(),
58 totalDays);
59 52 return lastDay;
60 53 }
61 54 }
62 55 TestVerifyData.apxc:
63 56 @isTest
64 57 private class TestVerifyDate {
65 58 @isTest static void Test_CheckDates_case1(){
66 59 Date D =
67 VerifyDate.CheckDates(date.parse('01/01/2022'),date.parse('01/05/
68 60 System.assertEquals(date.parse('01/05/2022'), D);
69 61 }
70 62 @isTest static void Test_CheckDates_case2(){
71 63 Date D = VerifyDate.CheckDates(date.parse('01/01/2022'),
72 date.parse('05/05/2022'));
73 64 System.assertEquals(date.parse('01/31/2022'), D);
74 65 }
75 66 @isTest static void Test_Within30Days_case1(){
76 67 Boolean flag =
77 VerifyDate.DateWithin30Days(date.parse('01/01/2022'),
```

```
78 68 date.parse('12/30/2021'));
79 69 System.assertEquals(false, flag);
80 70 }
81 71 @isTest static void Test_Within30Days_case2(){
82 72 Boolean flag =
83 VerifyDate.DateWithin30Days(date.parse('01/01/2022'),
84 73 date.parse('02/02/2021'));
85 74 System.assertEquals(false, flag);
86 75 }
87 76 @isTest static void Test_Within30Days_case3(){
88 77 SPSGP-10445-Salesforce Developer Catalyst
89 78 Self-Learning & Super Badges
90 79 3
91 80 APEX SPECIALIST SUPER BADGE CODES
92 81 Boolean flag =
93 VerifyDate.DateWithin30Days(date.parse('01/01/2022'),
94 82 date.parse('01/15/2022'));
95 83 System.assertEquals(true, flag);
96 84 }
97 85 @isTest static void Test_SetEndOfMonthDate(){
98 86 Date returndate =
99 VerifyDate.SetEndOfMonthDate(date.parse('01/01/2022'));
100 87 }
101 88 }
102 89 RestrictContactByName.apxt:
103 90 trigger RestrictContactByName on Contact (before insert,
    before
104 update) {
105 91 //check contacts prior to insert or update for invalid data
106 92 For (Contact c : Trigger.New) {
107 93 if(c.LastName == 'INVALIDNAME') { //invalidname is invalid
108 94 c.AddError('The Last Name "'+c.LastName+'" is not allowed for
109 95 }
110 96 }
111 97 }
112 98 TestRestrictContactByName.apxc:
113 99 @isTest
114 100 private class TestRestrictContactByName {
115 101 @isTest static void Test_insertupdateContact(){
116 102 Contact cnt = new Contact();
```

```
117 103 cnt.LastName = 'INVALIDNAME';
118 104 Test.startTest();
119 105 Database.SaveResult result = Database.insert(cnt,false);
120 106 Test.stopTest();
121 107 System.assert(!result.isSuccess());
122 108 System.assert(result.getErrors().size() > 0);
123 109 System.assertEquals('The Last Name "INVALIDNAME" is not
    allowed
124 110 result.getErrors()[0].getMessage());
125 111 }
126 112 }
127 113 SPSGP-10445-Salesforce Developer Catalyst
128 114 Self-Learning & Super Badges
129 115 4
130 116 APEX SPECIALIST SUPER BADGE CODES
131 117 RandomContactFactory.apxc:
132 118 public class RandomContactFactory {
133 119 public static List<Contact> generateRandomContacts(Integer
134 num_cnts, string lastname) {
135 120 List<Contact> contacts = new List<Contact>();
136 121 for(Integer i = 0; i < num_cnts; i++) {
137 122 Contact cnt = new Contact(FirstName = 'Test' +i,LastName =
138 lastname);
139 123 contacts.add(cnt);
140 124 }
141 125 return contacts;
142 126 }
143 127 }
144 128 ASYNCHRONOUS APEX
145 129 AccountProcessor.apxc:
146 130 public class AccountProcessor {
147 131 @future
148 132 public static void countContacts(List<Id> accountIds){
149 133 List<Account> accountsToUpdate = new List<Account>();
150 134 List<Account> accounts = [Select Id, Name, (Select Id from
151 Contacts)from Account Where Id in
152 135 :accountIds];
153 136 For(Account acc: accounts) {
154 137 List<Contact> contactList = acc.contacts;
155 138 acc.Number_Of_Contacts__c = contactList.size();
```

```
156 139 accountsToUpdate.add(acc);
157 140 }
158 141 update accountsToUpdate;
159 142 }
160 143 }
161 144 AccountProcessorTest.apxc:
162 145 @isTest
163 146 public class AccountProcessorTest {
164 147 @isTest
165 148 private static void testCountContacts() {
166 149 Account newAccount = new Account(Name = 'Test Account');
167 150 insert newAccount;
168 151 Contact newContact1 = new Contact(FirstName =
    'John',LastName =
169 'Doe',AccountId =
170 152 SPSGP-10445-Salesforce Developer Catalyst
171 153 Self-Learning & Super Badges
172 154 5
173 155 APEX SPECIALIST SUPER BADGE CODES
174 156 newAccount.Id);
175 157 insert newContact1;
176 158 Contact newContact2 = new Contact(FirstName =
    'John',LastName =
177 'Doe',AccountId =
178 159 newAccount.Id);
179 160 insert newContact2;
180 161 List<Id> accountIds = new List<Id>();
181 162 accountIds.add(newAccount.Id);
182 163 Test.startTest();
183 164 AccountProcessor.countContacts(accountIds);
184 165 Test.stopTest();
185 166 }
186 167 }
187 168 LeadProcessor.apxc:
188 169 global class LeadProcessor implements
189 Database.Batchable<sObject>{
190 170 global Integer count = 0;
191 171 global Database.QueryLocator start(Database.BatchableContext
    bc)
192 {
```

```
193 172 return Database.getQueryLocator('SELECT ID,LeadSource FROM
194 173 }
195 174 global void execute(Database.BatchableContext bc, List<Lead>
196 L_list){
197 175 List<lead> L_list_new = new List<lead>();
198 176 for(lead L: L_list){
199 177 L.leadSource = 'Dreamforce';
200 178 L_list_new.add(L);
201 179 count += 1;
202 180 }
203 181 update L_list_new;
204 182 }
205 183 global void finish(Database.BatchableContext bc){
206 184 system.debug('count = ' + count);
207 185 }
208 186 }
209 187 LeadProcessorTest.apxc:
210 188 @isTest
211 189 public class LeadProcessorTest {
212 190 @isTest
213 191 public static void testit() {
214 192 SPSGP-10445-Salesforce Developer Catalyst
215 193 Self-Learning & Super Badges
216 194 6
217 195 APEX SPECIALIST SUPER BADGE CODES
218 196 List<lead> L_list = new List<lead>();
219 197 for(Integer i = 0; i < 200; i++) {
220 198 Lead L = new Lead();
221 199 L.LastName = 'name' + i;
222 200 L.Company = 'Company';
223 201 L.Status = 'Random Status';
224 202 L_list.add(L);
225 203 }
226 204 insert L_list;
227 205 Test.startTest();
228 206 LeadProcessor lp = new LeadProcessor();
229 207 Id batchId = Database.executeBatch(lp);
230 208 Test.stopTest();
231 209 }
232 210 }
```

```
211 AddPrimaryContact.apxc:
212 public class AddPrimaryContact implements Queueable{
213 private Contact con;
214 private String state;
215 public AddPrimaryContact(Contact con, String state) {
216 this.con = con;
217 this.state = state;
218 }
219 public void execute(QueueableContext context) {
220 List<Account> accounts = [Select Id,Name,(Select
FirstName,LastName, Id from contacts)
221 from Account where BillingState = :state Limit 200];
222 List<Contact> primaryContacts = new List<Contact>();
223 for(Account acc : accounts) {
224 Contact c = con.clone();
225 c.AccountId = acc.Id;
226 primaryContacts.add(c);
227 }
228 if(primaryContacts.size() > 0) {
229 insert primaryContacts;
230 }
231 }
232 }
233 SPSGP-10445-Salesforce Developer Catalyst
234 Self-Learning & Super Badges
235 7
236 APEX SPECIALIST SUPER BADGE CODES
237 AddPrimaryContactTest.apxc:
238 @isTest
239 public class AddPrimaryContactTest {
240 static testmethod void testQueueable() {
241 List<Account> testAccounts = new List<Account>();
242 for(Integer i = 0; i < 50; i++) {
243 testAccounts.add(new Account (Name = 'Account' +
  i,BillingState
= 'CA'));
244 }
245 for(Integer j = 0; j < 50; j++) {
246 testAccounts.add(new Account(Name = 'Account'+ j,
  BillingState =
```

```
271 'NY'));
272 247 }
273 248 insert testAccounts;
274 249 Contact testContact = new Contact(FirstName = 'John',
    LastName =
275 'Doe');
276 250 insert testContact;
277 251 AddPrimaryContact addit = new
278 AddPrimaryContact(testContact,'CA');
279 252 Test.startTest();
280 253 system.enqueueJob(addit);
281 254 Test.stopTest();
282 255 System.assertEquals(50, [Select count() from Contact where
283 accountId in (Select Id from
284 256 Account where BillingState = 'CA')]);
285 257 }
286 258 }
287 259 DailyLeadProcessor.apxc:
288 260 global class DailyLeadProcessor implements Schedulable{
289 261 global void execute(SchedulableContext ctx) {
290 262 List<Lead> leadstoupdate = new List<Lead>();
291 263 List<Lead> leads = [Select id From Lead Where LeadSource =
    NULL
292 Limit 200];
293 264 for(Lead l: leads) {
294 265 l.LeadSource = 'Dreamforce';
295 266 leadstoupdate.add(l);
296 267 }
297 268 update leadstoupdate;
298 269 }
299 270 }
300 271 SPSGP-10445-Salesforce Developer Catalyst
301 272 Self-Learning & Super Badges
302 273 8
303 274 APEX SPECIALIST SUPER BADGE CODES
304 275 DailyLeadProcessorTest.apxc:
305 276 @isTest
306 277 private class DailyLeadProcessorTest {
307 278 public static String CRON_EXP = '0 0 0 15 3 ? 2024';
308 279 static testmethod void testScheduledJob() {
```

```
309 280 List<Lead> leads = new List<Lead>();
310 281 for(Integer i = 0; i < 200; i++) {
311 282 Lead l = new Lead(
312 283 FirstName = 'First' + i,
313 284 LastName = 'LastName',
314 285 Company = 'The Inc'
315 286 );
316 287 leads.add(l);
317 288 }
318 289 insert leads;
319 290 Test.startTest();
320 291 String jobId =
    System.schedule('ScheduledApexTest',CRON_EXP,new
321 DailyLeadProcessor());
322 292 Test.stopTest();
323 293 List<Lead> checkleads = new List<Lead>();
324 294 checkleads = [Select Id From Lead Where LeadSource =
325 'Dreamforce' and Company = 'The Inc'];
326 295 System.assertEquals(200,checkleads.size(),'Leads were not
327 296 }
328 297 }
329 298 APEX INTEGRATION SERVICES
330 299 AnimalLocator.apxc:
331 300 public class AnimalLocator{
332 301 public static String getAnimalNameById(Integer x){
333 302 Http http = new Http();
334 303 HttpRequest req = new HttpRequest();
335 304 req.setEndpoint('https://th-apex-http⊠305

336 306 Map<String, Object> animal= new Map<String, Object>();
337 307 HttpResponse res = http.send(req);
338 308 if (res.getStatusCode() == 200) {
339 309 SPSGP-10445-Salesforce Developer Catalyst
340 310 Self-Learning & Super Badges
341 311 9
342 312 APEX SPECIALIST SUPER BADGE CODES
343 313 Map<String, Object> results = (Map<String,
344 Object>)JSON.deserializeUntyped(res.getBody());
345 314 animal = (Map<String, Object>) results.get('animal');
346 315 }
```

```
347 316 return (String)animal.get('name');
348 317 }
349 318 }
350 319 AnimalLocatorTest.apxc:
351 320 @isTest
352 321 private class AnimalLocatorTest{
353 322 @isTest static void AnimalLocatorMock1() {
354 323 Test.setMock(HttpCalloutMock.class, new
    AnimalLocatorMock());
355 324 string result = AnimalLocator.getAnimalNameById(3);
356 325 String expectedResult = 'chicken';
357 326 System.assertEquals(result,expectedResult );
358 327 }
359 328 }
360 329 AnimalLocatorMock.apxc:
361 330 @isTest
362 331 global class AnimalLocatorMock implements HttpCalloutMock {
363 332 // Implement this interface method
364 333 global HTTPResponse respond(HTTPRequest request) {
365 334 // Create a fake response
366 335 HttpResponse response = new HttpResponse();
367 336 response.setHeader('Content-Type', 'application/json');
368 337 response.setBody('{"animals": ["majestic badger", "fluffy
369 bunny", "scary bear", "chicken", "mighty
370 338 moose"]}');
371 339 response.setStatusCode(200);
372 340 return response;
373 341 }
374 342 }
375 343 ParkLocator.apxc:
376 344 public class ParkLocator {
377 345 public static string[] country(string theCountry) {
378 346 ParkService.ParksImplPort parkSvc = new
379 ParkService.ParksImplPort(); // remove space
380 347 return parkSvc.byCountry(theCountry);
381 348 }
382 349 }
383 350 SPSGP-10445-Salesforce Developer Catalyst
384 351 Self-Learning & Super Badges
385 352 10
```

APEX SPECIALIST SUPER BADGE CODES
ParkLocatorTest.apxc:

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

ParkServiceMock.apxc:

```
@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();
response_x.return_x = new List<String>{'Yellowstone',
  'Mackinac
// end
response.put('response_x', response_x);
}
}
```

AccountManager.apxc:

```
@RestResource(urlMapping='/Accounts/*/contacts')
global class AccountManager {
@HttpGet
global static Account getAccount() {
```

```
424 390 RestRequest req = RestContext.request;
425 391 String accId = req.requestURI.substringBetween('Accounts/',
426 '/contacts');
427 392 SPSGP-10445-Salesforce Developer Catalyst
428 393 Self-Learning & Super Badges
429 394 11
430 395 APEX SPECIALIST SUPER BADGE CODES
431 396 Account acc = [SELECT Id, Name, (SELECT Id, Name FROM
   Contacts)
432 397 FROM Account WHERE Id = :accId];
433 398 return acc;
434 399 }
435 400 }
436 401 AccountManagerTest.apxc:
437 402 @isTest
438 403 private class AccountManagerTest {
439 404 private static testMethod void getAccountTest1() {
440 405 Id recordId = createTestRecord();
441 406 // Set up a test request
442 407 RestRequest request = new RestRequest();
443 408 request.requestUri =
444 'https://na1.salesforce.com/services/apexrest/Accounts/'+
445 recordId
446 409 +'/contacts' ;
447 410 request.httpMethod = 'GET';
448 411 RestContext.request = request;
449 412 // Call the method to test
450 413 Account thisAccount = AccountManager.getAccount();
451 414 // Verify results
452 415 System.assert(thisAccount != null);
453 416 System.assertEquals('Test record', thisAccount.Name);
454 417 }
455 418 // Helper method
456 419 static Id createTestRecord() {
457 420 // Create test record
458 421 Account TestAcc = new Account(
459 422 Name='Test record');
460 423 insert TestAcc;
461 424 Contact TestCon= new Contact(
462 425 LastName='Test',
```

```
463 426 AccountId = TestAcc.id);
464 427 return TestAcc.Id;
465 428 }
466 429 }
467 430 SPSGP-10445-Salesforce Developer Catalyst
468 431 Self-Learning & Super Badges
469 432 12
470 433 APEX SPECIALIST SUPER BADGE CODES
471 434 APEX SPECIALIST SUPER BADGE
472 435 Challenge-1
473 436 MaintenanceRequestHelper.apxc:
474 437 public with sharing class MaintenanceRequestHelper {
475 438 public static void updateworkOrders(List<Case>
    updWorkOrders,
476 Map<Id,Case> nonUpdCaseMap) {
477 439 Set<Id> validIds = new Set<Id>();
478 440 For (Case c : updWorkOrders){
479 441 if (nonUpdCaseMap.get(c.Id).Status != 'Closed' && c.Status
    ==
480 'Closed'){
481 442 if (c.Type == 'Repair' || c.Type == 'Routine Maintenance'){
482 443 validIds.add(c.Id);
483 444 }
484 445 }
485 446 }
486 447 if (!validIds.isEmpty()){
487 448 List<Case> newCases = new List<Case>();
488 449 Map<Id,Case> closedCasesM = new Map<Id,Case>([SELECT Id,
489 Vehicle__c, Equipment__c,
490 450 Equipment__r.Maintenance_Cycle__c,(SELECT
491 Id,Equipment__c,Quantity__c FROM
492 451 Equipment_Maintenance_Items__r)
493 452 FROM Case WHERE Id IN :validIds]);
494 453 Map<Id,Decimal> maintenanceCycles = new Map<ID,Decimal>();
495 454 AggregateResult[] results = [SELECT Maintenance_Request__c,
496 455 MIN(Equipment__r.Maintenance_Cycle__c)cycle FROM
497 Equipment_Maintenance_Item__c WHERE
498 456 Maintenance_Request__c IN :ValidIds GROUP BY
499 Maintenance_Request__c];
500 457 for (AggregateResult ar : results){
```

```
501 458 maintenanceCycles.put((Id) ar.get('Maintenance_Request__c'),
502 (Decimal) ar.get('cycle'));
503 459 }
504 460 for(Case cc : closedCasesM.values()){
505 461 Case nc = new Case (
506 462 ParentId = cc.Id,
507 463 Status = 'New',
508 464 SPSGP-10445-Salesforce Developer Catalyst
509 465 Self-Learning & Super Badges
510 466 13
511 467 APEX SPECIALIST SUPER BADGE CODES
512 468 Subject = 'Routine Maintenance',
513 469 Type = 'Routine Maintenance',
514 470 Vehicle__c = cc.Vehicle__c,
515 471 Equipment__c =cc.Equipment__c,
516 472 Origin = 'Web',
517 473 Date_Reported__c = Date.Today()
518 474 );
519 475 If (maintenanceCycles.containskey(cc.Id)){
520 476 nc.Date_Due__c = Date.today().addDays((Integer)
521 maintenanceCycles.get(cc.Id));
522 477 }
523 478 newCases.add(nc);
524 479 }
525 480 insert newCases;
526 481 List<Equipment_Maintenance_Item__c> clonedWPs = new
527 482 List<Equipment_Maintenance_Item__c>();
528 483 for (Case nc : newCases){
529 484 for (Equipment_Maintenance_Item__c wp :
530 485
   closedCasesM.get(nc.ParentId).Equipment_Maintenance_Items__r){
531 486 Equipment_Maintenance_Item__c wpClone = wp.clone();
532 487 wpClone.Maintenance_Request__c = nc.Id;
533 488 ClonedWPs.add(wpClone);
534 489 }
535 490 }
536 491 insert ClonedWPs;
537 492 }
538 493 }
539 494 }
```

```
540 495 SPSGP-10445-Salesforce Developer Catalyst
541 496 Self-Learning & Super Badges
542 497 14
543 498 APEX SPECIALIST SUPER BADGE CODES
544 499 MaintenanceRequest.apxt:
545 500 trigger MaintenanceRequest on Case (before update, after
    update)
546 {
547 501 if(Trigger.isUpdate && Trigger.isAfter){
548 502 MaintenanceRequestHelper.updateWorkOrders(Trigger.New,
549 Trigger.OldMap);
550 503 }
551 504 }
552 505 MaintenanceRequestHelperTest.apxc:
553 506 @istest
554 507 public with sharing class MaintenanceRequestHelperTest {
555 508 private static final string STATUS_NEW = 'New';
556 509 private static final string WORKING = 'Working';
557 510 private static final string CLOSED = 'Closed';
558 511 private static final string REPAIR = 'Repair';
559 512 private static final string REQUEST_ORIGIN = 'Web';
560 513 private static final string REQUEST_TYPE = 'Routine
561 514 private static final string REQUEST_SUBJECT = 'Testing

562 515 PRIVATE STATIC Vehicle__c createVehicle(){
563 516 Vehicle__c Vehicle = new Vehicle__C(name = 'SuperTruck');
564 517 return Vehicle;
565 518 }
566 519 PRIVATE STATIC Product2 createEq(){
567 520 product2 equipment = new product2(name = 'SuperEquipment',
568 521 lifespan_months__C = 10,
569 522 maintenance_cycle__C = 10,
570 523 replacement_part__c = true);
571 524 return equipment;
572 525 }
573 526 PRIVATE STATIC Case createMaintenanceRequest(id vehicleId,
    id
574 equipmentId){
575 527 case cs = new case(Type=REPAIR,
576 528 Status=STATUS_NEW,
```

```
577 529 Origin=REQUEST_ORIGIN,
578 530 Subject=REQUEST_SUBJECT,
579 531 Equipment__c=equipmentId,
580 532 SPSGP-10445-Salesforce Developer Catalyst
581 533 Self-Learning & Super Badges
582 534 15
583 535 APEX SPECIALIST SUPER BADGE CODES
584 536 Vehicle__c=vehicleId);
585 537 return cs;
586 538 }
587 539 PRIVATE STATIC Equipment_Maintenance_Item__c
    createWorkPart(id
588 equipmentId,id requestId){
589 540 Equipment_Maintenance_Item__c wp = new
590 Equipment_Maintenance_Item__c(Equipment__c =
591 541 equipmentId,
592 542 Maintenance_Request__c = requestId);
593 543 return wp;
594 544 }
595 545 @istest
596 546 private static void testMaintenanceRequestPositive(){
597 547 Vehicle__c vehicle = createVehicle();
598 548 insert vehicle;
599 549 id vehicleId = vehicle.Id;
600 550 Product2 equipment = createEq();
601 551 insert equipment;
602 552 id equipmentId = equipment.Id;
603 553 case somethingToUpdate =
604 createMaintenanceRequest(vehicleId,equipmentId);
605 554 insert somethingToUpdate;
606 555 Equipment_Maintenance_Item__c workP =
607 createWorkPart(equipmentId,somethingToUpdate.id);
608 556 insert workP;
609 557 test.startTest();
610 558 somethingToUpdate.status = CLOSED;
611 559 update somethingToUpdate;
612 560 test.stopTest();
613 561 Case newReq = [Select id, subject, type, Equipment__c,
614 Date_Reported__c, Vehicle__c,
615 562 Date_Due__c
```

```
616 563 from case
617 564 where status =:STATUS_NEW];
618 565 SPSGP-10445-Salesforce Developer Catalyst
619 566 Self-Learning & Super Badges
620 567 16
621 568 APEX SPECIALIST SUPER BADGE CODES
622 569 Equipment_Maintenance_Item__c workPart = [select id
623 570 from Equipment_Maintenance_Item__c
624 571 where Maintenance_Request__c =:newReq.Id];
625 572 system.assert(workPart != null);
626 573 system.assert(newReq.Subject != null);
627 574 system.assertEquals(newReq.Type, REQUEST_TYPE);
628 575 SYSTEM.assertEquals(newReq.Equipment__c, equipmentId);
629 576 SYSTEM.assertEquals(newReq.Vehicle__c, vehicleId);
630 577 SYSTEM.assertEquals(newReq.Date_Reported__c,
   system.today());
631 578 }
632 579 @istest
633 580 private static void testMaintenanceRequestNegative(){
634 581 Vehicle__C vehicle = createVehicle();
635 582 insert vehicle;
636 583 id vehicleId = vehicle.Id;
637 584 product2 equipment = createEq();
638 585 insert equipment;
639 586 id equipmentId = equipment.Id;
640 587 case emptyReq =
   createMaintenanceRequest(vehicleId,equipmentId);
641 588 insert emptyReq;
642 589 Equipment_Maintenance_Item__c workP =
643 createWorkPart(equipmentId, emptyReq.Id);
644 590 insert workP;
645 591 test.startTest();
646 592 emptyReq.Status = WORKING;
647 593 update emptyReq;
648 594 test.stopTest();
649 595 list<case> allRequest = [select id
650 596 from case];
651 597 Equipment_Maintenance_Item__c workPart = [select id
652 598 from Equipment_Maintenance_Item__c
653 599 SPSGP-10445-Salesforce Developer Catalyst
```

656 602 APEX SPECIALIST SUPER BADGE CODES

```
657 603 where Maintenance_Request__c = :emptyReq.Id];
658 604 system.assert(workPart != null);
659 605 system.assert(allRequest.size() == 1);
660 606 }
661 607 @istest
662 608 private static void testMaintenanceRequestBulk(){
663 609 list<Vehicle__C> vehicleList = new list<Vehicle__C>();
664 610 list<Product2> equipmentList = new list<Product2>();
665 611 list<Equipment_Maintenance_Item__c> workPartList = new
666 612 list<Equipment_Maintenance_Item__c>();
667 613 list<case> requestList = new list<case>();
668 614 list<id> oldRequestIds = new list<id>();
669 615 for(integer i = 0; i < 300; i++){
670 616 vehicleList.add(createVehicle());
671 617 equipmentList.add(createEq());
672 618 }
673 619 insert vehicleList;
674 620 insert equipmentList;
675 621 for(integer i = 0; i < 300; i++){
676 622
   requestList.add(createMaintenanceRequest(vehicleList.get(i).id,
677 equipmentList.get(i).id));
678 623 }
679 624 insert requestList;
680 625 for(integer i = 0; i < 300; i++){
681 626 workPartList.add(createWorkPart(equipmentList.get(i).id,
682 requestList.get(i).id));
683 627 }
684 628 insert workPartList;
685 629 test.startTest();
686 630 for(case req : requestList){
687 631 req.Status = CLOSED;
688 632 oldRequestIds.add(req.Id);
689 633 }
690 634 update requestList;
```

691 635 SPSGP-10445-Salesforce Developer Catalyst

```
693 637 18
694 638 APEX SPECIALIST SUPER BADGE CODES
695 639 test.stopTest();
696 640 list<case> allRequests = [select id
697 641 from case
698 642 where status =: STATUS_NEW];
699 643 list<Equipment_Maintenance_Item__c> workParts = [select id
700 644 from Equipment_Maintenance_Item__c
701 645 where Maintenance_Request__c in: oldRequestIds];
702 646 system.assert(allRequests.size() == 300);
703 647 }
704 648 }
705 649 Challenge-2
706 650 WarehouseCalloutService.apxc:
707 651 public with sharing class WarehouseCalloutService implements
708 Queueable {
709 652 private static final String WAREHOUSE_URL = 'https://th-
710 653 //class that makes a REST callout to an external warehouse
711 system to get a list of equipment that
712 654 needs to be updated.
713 655 //The callout's JSON response returns the equipment records
   that
714 you upsert in Salesforce.
715 656 @future(callout=true)
716 657 public static void runWarehouseEquipmentSync(){
717 658 Http http = new Http();
718 659 HttpRequest request = new HttpRequest();
719 660 request.setEndpoint(WAREHOUSE_URL);
720 661 request.setMethod('GET');
721 662 HttpResponse response = http.send(request);
722 663 List<Product2> warehouseEq = new List<Product2>();
723 664 if (response.getStatusCode() == 200){
724 665 List<Object> jsonResponse =
725 (List<Object>)JSON.deserializeUntyped(response.getBody());
726 666 SPSGP-10445-Salesforce Developer Catalyst
727 667 Self-Learning & Super Badges
728 668 19
729 669 APEX SPECIALIST SUPER BADGE CODES
730 670 System.debug(response.getBody());
731 671 //class maps the following fields: replacement part (always
```

```
732 true), cost, current inventory,
733 672 lifespan, maintenance cycle, and warehouse SKU
734 673 //warehouse SKU will be external ID for identifying which
735 equipment records to update within
736 674 Salesforce
737 675 for (Object eq : jsonResponse){
738 676 Map<String,Object> mapJson = (Map<String,Object>)eq;
739 677 Product2 myEq = new Product2();
740 678 myEq.Replacement_Part__c = (Boolean)
    mapJson.get('replacement');
741 679 myEq.Name = (String) mapJson.get('name');
742 680 myEq.Maintenance_Cycle__c = (Integer)
743 mapJson.get('maintenanceperiod');
744 681 myEq.Lifespan_Months__c = (Integer) mapJson.get('lifespan');
745 682 myEq.Cost__c = (Integer) mapJson.get('cost');
746 683 myEq.Warehouse_SKU__c = (String) mapJson.get('sku');
747 684 myEq.Current_Inventory__c = (Double)
    mapJson.get('quantity');
748 685 myEq.ProductCode = (String) mapJson.get('_id');
749 686 warehouseEq.add(myEq);
750 687 }
751 688 if (warehouseEq.size() > 0){
752 689 upsert warehouseEq;
753 690 System.debug('Your equipment was synced with the warehouse
754 691 }
755 692 }
756 693 }
757 694 public static void execute (QueueableContext context){
758 695 runWarehouseEquipmentSync();
759 696 }
760 697 }
761 698 WarehouseCalloutServiceMock.apxc:
762 699 @isTest
763 700 global class WarehouseCalloutServiceMock implements
764 HttpCalloutMock {
765 701 // implement http mock callout
766 702 global static HttpResponse respond(HttpRequest request) {
767 703 SPSGP-10445-Salesforce Developer Catalyst
768 704 Self-Learning & Super Badges
769 705 20
```

```
770 706 APEX SPECIALIST SUPER BADGE CODES
771 707 HttpResponse response = new HttpResponse();
772 708 response.setHeader('Content-Type', 'application/json');
773 709
  response.setBody('[{"_id":"55d66226726b611100aaf741","replacemen
774 t":false,"quantity":5,"name":"Gene
775 710 rator 1000
776 711
  kW","maintenanceperiod":365,"lifespan":120,"cost":5000,"sku":"10
777 712 af742","replacement":true,"quantity":183,"name":"Cooling
778 713
  Fan","maintenanceperiod":0,"lifespan":0,"cost":300,"sku":"100004
779 "},{"_id":"55d66226726b611100aaf743
780 714 ","replacement":true,"quantity":143,"name":"Fuse
781 715
  20A","maintenanceperiod":0,"lifespan":0,"cost":22,"sku":"100005"
782 }]');
783 716 response.setStatusCode(200);
784 717 return response;
785 718 }
786 719 }
787 720 WarehouseCalloutServiceTest.apxc:
788 721 @IsTest
789 722 private class WarehouseCalloutServiceTest {
790 723 // implement your mock callout test here
791 724 @isTest
792 725 static void testWarehouseCallout() {
793 726 test.startTest();
794 727 test.setMock(HttpCalloutMock.class, new
795 WarehouseCalloutServiceMock());
796 728 WarehouseCalloutService.execute(null);
797 729 test.stopTest();
798 730 List<Product2> product2List = new List<Product2>();
799 731 product2List = [SELECT ProductCode FROM Product2];
800 732 System.assertEquals(3, product2List.size());
801 733 System.assertEquals('55d66226726b611100aaf741',
802 product2List.get(0).ProductCode);
803 734 System.assertEquals('55d66226726b611100aaf742',
804 product2List.get(1).ProductCode);
805 735 System.assertEquals('55d66226726b611100aaf743',
```

```
806 product2List.get(2).ProductCode);
807 736 }
808 737 }
809 738 Challenge-3
810 739 WarehouseSyncSchedule.apxc:
811 740 global with sharing class WarehouseSyncSchedule implements
812 Schedulable{
813 741 SPSGP-10445-Salesforce Developer Catalyst
814 742 Self-Learning & Super Badges
815 743 21
816 744 APEX SPECIALIST SUPER BADGE CODES
817 745 global void execute(SchedulableContext ctx){
818 746 System.enqueueJob(new WarehouseCalloutService());
819 747 }
820 748 }
821 749 WarehouseSyncScheduuleTest.apxc:
822 750 @isTest
823 751 public class WarehouseSyncScheduleTest {
824 752 @isTest static void WarehousescheduleTest(){
825 753 String scheduleTime = '00 00 01 * * ?';
826 754 Test.startTest();
827 755 Test.setMock(HttpCalloutMock.class, new
828 WarehouseCalloutServiceMock());
829 756 String jobID=System.schedule('Warehouse Time To Schedule to
830 757 WarehouseSyncSchedule());
831 758 Test.stopTest();
832 759 //Contains schedule information for a scheduled job. CronTrigger
    is similar to a cron job on UNIX
833 
834 760 systems.
835 761 // This object is available in API version 17.0 and later.
836 762 CronTrigger a=[SELECT Id FROM CronTrigger where NextFireTime
    >
837 today];
838 763 System.assertEquals(jobID, a.Id,'Schedule ');
839 764 }
840 765 }
841 766 Challenge-4
842 767 MaintenanceRequestHelperTest.apxc:
843 768 @istest
```

```apex
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(){
SPSGP-10445-Salesforce Developer Catalyst
Self-Learning & Super Badges
22
APEX SPECIALIST SUPER BADGE CODES
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 =
```

```
881 803 equipmentId, Maintenance_Request__c = requestId);
882 804 return wp;
883 805 }
884 806 @istest
885 807 private static void testMaintenanceRequestPositive(){
886 808 Vehicle__c vehicle = createVehicle();
887 809 insert vehicle;
888 810 id vehicleId = vehicle.Id;
889 811 Product2 equipment = createEq();
890 812 insert equipment;
891 813 id equipmentId = equipment.Id;
892 814 SPSGP-10445-Salesforce Developer Catalyst
893 815 Self-Learning & Super Badges
894 816 23
895 817 APEX SPECIALIST SUPER BADGE CODES
896 818 case somethingToUpdate =
897 createMaintenanceRequest(vehicleId,equipmentId);
898 819 insert somethingToUpdate;
899 820 Equipment_Maintenance_Item__c workP =
900 createWorkPart(equipmentId,somethingToUpdate.id);
901 821 insert workP;
902 822 test.startTest();
903 823 somethingToUpdate.status = CLOSED;
904 824 update somethingToUpdate;
905 825 test.stopTest();
906 826 Case newReq = [Select id, subject, type, Equipment__c,
907 Date_Reported__c, Vehicle__c,
908 827 Date_Due__c
909 828 from case
910 829 where status =:STATUS_NEW];
911 830 Equipment_Maintenance_Item__c workPart = [select id
912 831 from Equipment_Maintenance_Item__c
913 832 where Maintenance_Request__c =:newReq.Id];
914 833 system.assert(workPart != null);
915 834 system.assert(newReq.Subject != null);
916 835 system.assertEquals(newReq.Type, REQUEST_TYPE);
917 836 SYSTEM.assertEquals(newReq.Equipment__c, equipmentId);
918 837 SYSTEM.assertEquals(newReq.Vehicle__c, vehicleId);
919 838 SYSTEM.assertEquals(newReq.Date_Reported__c,
   system.today());
```

```apex
839 }
840 @istest
841 private static void testMaintenanceRequestNegative(){
842 Vehicle__C vehicle = createVehicle();
843 insert vehicle;
844 id vehicleId = vehicle.Id;
845 product2 equipment = createEq();
846 insert equipment;
847 id equipmentId = equipment.Id;
848 SPSGP-10445-Salesforce Developer Catalyst
849 Self-Learning & Super Badges
850 24
851 APEX SPECIALIST SUPER BADGE CODES
852 case emptyReq =
  createMaintenanceRequest(vehicleId,equipmentId);
853 insert emptyReq;
854 Equipment_Maintenance_Item__c workP =
createWorkPart(equipmentId, emptyReq.Id);
855 insert workP;
856 test.startTest();
857 emptyReq.Status = WORKING;
858 update emptyReq;
859 test.stopTest();
860 list<case> allRequest = [select id
861 from case];
862 Equipment_Maintenance_Item__c workPart = [select id
863 from Equipment_Maintenance_Item__c
864 where Maintenance_Request__c = :emptyReq.Id];
865 system.assert(workPart != null);
866 system.assert(allRequest.size() == 1);
867 }
868 @istest
869 private static void testMaintenanceRequestBulk(){
870 list<Vehicle__C> vehicleList = new list<Vehicle__C>();
871 list<Product2> equipmentList = new list<Product2>();
872 list<Equipment_Maintenance_Item__c> workPartList = new
873 list<Equipment_Maintenance_Item__c>();
874 list<case> requestList = new list<case>();
875 list<id> oldRequestIds = new list<id>();
876 for(integer i = 0; i < 300; i++){
```

```
959 877 vehicleList.add(createVehicle());
960 878 equipmentList.add(createEq());
961 879 }
962 880 insert vehicleList;
963 881 insert equipmentList;
964 882 SPSGP-10445-Salesforce Developer Catalyst
965 883 Self-Learning & Super Badges
966 884 25
967 885 APEX SPECIALIST SUPER BADGE CODES
968 886 for(integer i = 0; i < 300; i++){
969 887
    requestList.add(createMaintenanceRequest(vehicleList.get(i).id,
970 equipmentList.get(i).id));
971 888 }
972 889 insert requestList;
973 890 for(integer i = 0; i < 300; i++){
974 891 workPartList.add(createWorkPart(equipmentList.get(i).id,
975 requestList.get(i).id));
976 892 }
977 893 insert workPartList;
978 894 test.startTest();
979 895 for(case req : requestList){
980 896 req.Status = CLOSED;
981 897 oldRequestIds.add(req.Id);
982 898 }
983 899 update requestList;
984 900 test.stopTest();
985 901 list<case> allRequests = [select id
986 902 from case
987 903 where status =: STATUS_NEW];
988 904 list<Equipment_Maintenance_Item__c> workParts = [select id
989 905 from Equipment_Maintenance_Item__c
990 906 where Maintenance_Request__c in: oldRequestIds];
991 907 system.assert(allRequests.size() == 300);
992 908 }
993 909 }
994 910 MaintenanceRequestHelper.apxc:
995 911 public with sharing class MaintenanceRequestHelper {
996 912 public static void updateworkOrders(List<Case>
    updWorkOrders,
```

```
997 Map<Id,Case> nonUpdCaseMap) {
998 913 Set<Id> validIds = new Set<Id>();
999 914 For (Case c : updWorkOrders){
1000915 if (nonUpdCaseMap.get(c.Id).Status != 'Closed' && c.Status ==
1001'Closed'){
1002916 SPSGP-10445-Salesforce Developer Catalyst
1003917 Self-Learning & Super Badges
1004918 26
1005919 APEX SPECIALIST SUPER BADGE CODES
1006920 if (c.Type == 'Repair' || c.Type == 'Routine Maintenance'){
1007921 validIds.add(c.Id);
1008922 }
1009923 }
1010924 }
1011925 if (!validIds.isEmpty()){
1012926 List<Case> newCases = new List<Case>();
1013927 Map<Id,Case> closedCasesM = new Map<Id,Case>([SELECT Id,
1014Vehicle__c, Equipment__c,
1015928 Equipment__r.Maintenance_Cycle__c,(SELECT
1016Id,Equipment__c,Quantity__c FROM
1017929 Equipment_Maintenance_Items__r)
1018930 FROM Case WHERE Id IN :validIds]);
1019931 Map<Id,Decimal> maintenanceCycles = new Map<ID,Decimal>();
1020932 AggregateResult[] results = [SELECT Maintenance_Request__c,
1021933 MIN(Equipment__r.Maintenance_Cycle__c)cycle FROM
1022Equipment_Maintenance_Item__c WHERE
1023934 Maintenance_Request__c IN :ValidIds GROUP BY
1024Maintenance_Request__c];
1025935 for (AggregateResult ar : results){
1026936 maintenanceCycles.put((Id) ar.get('Maintenance_Request__c'),
1027(Decimal) ar.get('cycle'));
1028937 }
1029938 for(Case cc : closedCasesM.values()){
1030939 Case nc = new Case (
1031940 ParentId = cc.Id,
1032941 Status = 'New',
1033942 Subject = 'Routine Maintenance',
1034943 Type = 'Routine Maintenance',
1035944 Vehicle__c = cc.Vehicle__c,
```

```
1036945 Equipment__c =cc.Equipment__c,
1037946 Origin = 'Web',
1038947 Date_Reported__c = Date.Today()
1039948 );
1040949 If (maintenanceCycles.containskey(cc.Id)){
1041950 nc.Date_Due__c = Date.today().addDays((Integer)
1042maintenanceCycles.get(cc.Id));
1043951 SPSGP-10445-Salesforce Developer Catalyst
1044952 Self-Learning & Super Badges
1045953 27
1046954 APEX SPECIALIST SUPER BADGE CODES
1047955 }
1048956 newCases.add(nc);
1049957 }
1050958 insert newCases;
1051959 List<Equipment_Maintenance_Item__c> clonedWPs = new
1052960 List<Equipment_Maintenance_Item__c>();
1053961 for (Case nc : newCases){
1054962 for (Equipment_Maintenance_Item__c wp :
1055963
   closedCasesM.get(nc.ParentId).Equipment_Maintenance_Items__r){
1056964 Equipment_Maintenance_Item__c wpClone = wp.clone();
1057965 wpClone.Maintenance_Request__c = nc.Id;
1058966 ClonedWPs.add(wpClone);
1059967 }
1060968 }
1061969 insert ClonedWPs;
1062970 }
1063971 }
1064972 }
1065973 Challenge-5
1066974 WarehouseCalloutService.apxc:
1067975 public with sharing class WarehouseCalloutService implements
1068Queueable {
1069976 private static final String WAREHOUSE_URL = 'https://th 977
   //class that makes a REST callout to an external warehouse
1070system to get a list of equipment that
1071978 needs to be updated.
1072979 //The callout's JSON response returns the equipment records
   that
```

```
1073 you upsert in Salesforce.
1074 980 @future(callout=true)
1075 981 public static void runWarehouseEquipmentSync(){
1076 982 Http http = new Http();
1077 983 HttpRequest request = new HttpRequest();
1078 984 request.setEndpoint(WAREHOUSE_URL);
1079 985 SPSGP-10445-Salesforce Developer Catalyst
1080 986 Self-Learning & Super Badges
1081 987 28
1082 988 APEX SPECIALIST SUPER BADGE CODES
1083 989 request.setMethod('GET');
1084 990 HttpResponse response = http.send(request);
1085 991 List<Product2> warehouseEq = new List<Product2>();
1086 992 if (response.getStatusCode() == 200){
1087 993 List<Object> jsonResponse =
1088 (List<Object>)JSON.deserializeUntyped(response.getBody());
1089 994 System.debug(response.getBody());
1090 995 //class maps the following fields: replacement part (always
1091 true), cost, current inventory,
1092 996 lifespan, maintenance cycle, and warehouse SKU
1093 997 //warehouse SKU will be external ID for identifying which
1094 equipment records to update within
1095 998 Salesforce
1096 999 for (Object eq : jsonResponse){
1097 1000 Map<String,Object> mapJson = (Map<String,Object>)eq;
1098 1001 Product2 myEq = new Product2();
1099 1002 myEq.Replacement_Part__c = (Boolean)
     mapJson.get('replacement');
1100 1003 myEq.Name = (String) mapJson.get('name');
1101 1004 myEq.Maintenance_Cycle__c = (Integer)
1102 mapJson.get('maintenanceperiod');
1103 1005 myEq.Lifespan_Months__c = (Integer) mapJson.get('lifespan');
1104 1006 myEq.Cost__c = (Integer) mapJson.get('cost');
1105 1007 myEq.Warehouse_SKU__c = (String) mapJson.get('sku');
1106 1008 myEq.Current_Inventory__c = (Double)
     mapJson.get('quantity');
1107 1009 myEq.ProductCode = (String) mapJson.get('_id');
1108 1010 warehouseEq.add(myEq);
1109 1011 }
1110 1012 if (warehouseEq.size() > 0){
```

```
1111 1013 upsert warehouseEq;
1112 1014 System.debug('Your equipment was synced with the warehouse
1113 1015 }
1114 1016 }
1115 1017 }
1116 1018 public static void execute (QueueableContext context){
1117 1019 runWarehouseEquipmentSync();
1118 1020 }
1119 1021 SPSGP-10445-Salesforce Developer Catalyst
1120 1022 Self-Learning & Super Badges
1121 1023 29
1122 1024 APEX SPECIALIST SUPER BADGE CODES
1123 1025 }
1124 1026 WarehouseCalloutServiceMock.apxc:
1125 1027 @isTest
1126 1028 global class WarehouseCalloutServiceMock implements
1127 HttpCalloutMock {
1128 1029 // implement http mock callout
1129 1030 global static HttpResponse respond(HttpRequest request) {
1130 1031 HttpResponse response = new HttpResponse();
1131 1032 response.setHeader('Content-Type', 'application/json');
1132 1033 response.setBody('[{"_id":"55d66226726b611100aaf741","replac
     emen
1133 t":false,"quantity":5,"name":"Gene
1134 1034 rator 1000
1135 1035 kW","maintenanceperiod":365,"lifespan":120,"cost":5000,"sku"

1136 1036 af742","replacement":true,"quantity":183,"name":"Cooling
1137 1037 Fan","maintenanceperiod":0,"lifespan":0,"cost":300,"sku":"10
     0004
1138 "},{"_id":"55d66226726b611100aaf743
1139 1038 ","replacement":true,"quantity":143,"name":"Fuse
1140 1039 20A","maintenanceperiod":0,"lifespan":0,"cost":22,"sku":"100

1141 }]');
1142 1040 response.setStatusCode(200);
1143 1041 return response;
1144 1042 }
1145 1043 }
1146 1044 WarehouseCalloutServiceTest.apxc:
```

```apex
1147 1045 @isTest
1148 1046 global class WarehouseCalloutServiceMock implements
1149 HttpCalloutMock {
1150 1047 // implement http mock callout
1151 1048 global static HttpResponse respond(HttpRequest request) {
1152 1049 HttpResponse response = new HttpResponse();
1153 1050 response.setHeader('Content-Type', 'application/json');
1154 1051 response.setBody('[{"_id":"55d66226726b611100aaf741","replac
     emen
1155 t":false,"quantity":5,"name":"Gene
1156 1052 rator 1000
1157 1053 kW","maintenanceperiod":365,"lifespan":120,"cost":5000,"sku"

1158 1054 af742","replacement":true,"quantity":183,"name":"Cooling
1159 1055 Fan","maintenanceperiod":0,"lifespan":0,"cost":300,"sku":"10
     0004
1160 "},{"_id":"55d66226726b611100aaf743
1161 1056 ","replacement":true,"quantity":143,"name":"Fuse
1162 1057 20A","maintenanceperiod":0,"lifespan":0,"cost":22,"sku":"100

1163 }]');
1164 1058 SPSGP-10445-Salesforce Developer Catalyst
1165 1059 Self-Learning & Super Badges
1166 1060 30
1167 1061 APEX SPECIALIST SUPER BADGE CODES
1168 1062 response.setStatusCode(200);
1169 1063 return response;
1170 1064 }
1171 1065 }
1172 1066 Challenge-6
1173 1067 WarehouseSyncSchedule.apxc:
1174 1068 global with sharing class WarehouseSyncSchedule implements
1175 Schedulable{
1176 1069 global void execute(SchedulableContext ctx){
1177 1070 System.enqueueJob(new WarehouseCalloutService());
1178 1071 }
1179 1072 }
1180 1073 WarehouseSyncScheduleTest.apxc:
1181 1074 @isTest
1182 1075 public class WarehouseSyncScheduleTest {
```

```
11831076@isTest static void WarehousescheduleTest(){
11841077String scheduleTime = '00 00 01 * * ?';
11851078Test.startTest();
11861079Test.setMock(HttpCalloutMock.class, new
1187WarehouseCalloutServiceMock());
11881080String jobID=System.schedule('Warehouse Time To Schedule to
11891081WarehouseSyncSchedule());
11901082Test.stopTest();
11911083//Contains schedule information for a scheduled job.
   CronTrigger
1192is similar to a cron job on UNIX
11931084systems.
11941085// This object is available in API version 17.0 and later.
11951086CronTrigger a=[SELECT Id FROM CronTrigger where NextFireTime
   >
1196today];
11971087System.assertEquals(jobID, a.Id,'Schedule ');
11981088}
11991089
```