**Apex Specialist SuperBadge**

**1.Get Started with Apex Unit Tests**

**Q. Create and install a simple Apex class to test if a date is within a proper range, and if not, returns a date that occurs at the end of the month within the range. You'll copy the code for the class from GitHub. Then write unit tests that achieve 100% code coverage.**

**Output:**

public class VerifyDate {

//method to handle potential checks against two dates

public static Date CheckDates(Date date1, Date date2) {

//if date2 is within the next 30 days of date1, use date2. Otherwise use the end of the month

if(DateWithin30Days(date1,date2)) {

return date2;

} else {

return SetEndOfMonthDate(date1);

}

}

//method to check if date2 is within the next 30 days of date1

private static Boolean DateWithin30Days(Date date1, Date date2) {

//check for date2 being in the past

if( date2 < date1) { return false; }

//check that date2 is within (>=) 30 days of date1

Date date30Days = date1.addDays(30); //create a date 30 days away from date1

if( date2 >= date30Days ) { return false; }

else { return true; }

}

//method to return the end of the month of a given date

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;

}

}

**TestVerifyDate**

@isTest

public class TestVerifyDate {

@isTest Static void test1(){

Date d = verifyDate.CheckDates(date.parse('01/01/2020'), date.parse('01/03/2020'));

System.assertEquals(Date.parse('01/03/2020'),d);

}

@isTest Static void test2(){

Date d = verifyDate.CheckDates(date.parse('01/01/2020'), date.parse('03/03/2020'));

System.assertEquals(Date.parse('01/31/2020'),d);

}

}

**2.Test Apex Triggers**

**Q. Create and install a simple Apex trigger which blocks inserts and updates to any contact with a last name of 'INVALIDNAME'. You'll copy the code for the class from GitHub. Then write unit tests that achieve 100% code coverage.**

**Output:**

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

//check contacts prior to insert or update for invalid data

For (Contact c : Trigger.New) {

if(c.LastName == 'INVALIDNAME') { //invalidname is invalid

c.AddError('The Last Name "'+c.LastName+'" is not allowed for DML');

}

}

}

**Test Class**

@isTest

public class TestRestrictContactByName {

@isTest

public static void testContact(){

Contact ct =new Contact();

ct.LastName = 'INVALIDNAME';

Database.SaveResult res = Database.insert(ct,false);

System.AssertEquals('The Last Name "INVALIDNAME" is not allowed for DML',res.getErrors()[0].getMessage());

}

}

**3.Create Test Data for Apex Tests**

**Q. Create an Apex class that returns a list of contacts based on two incoming parameters: the number of contacts to generate and the last name. Do not insert the generated contact records into the database.**

**Output:**

public class RandomContactFactory {

public static List<Contact> generateRandomContacts(Integer num, String lastName){

List<Contact> ContactList = new List<Contact>();

for(Integer i = 0;i<num;i++){

Contact ct = new Contact(FirstName ='Test ' +i, LastName =lastName);

ContactList.add(ct);

}

return contactList;

}

}