Software Test Plan for:

Kirkland Signature Online Survey Tool

Prepared by:
Josh Pfeffer, Jeremy Koletar, Joe Thomas, Patrick Cook, Tim Wong

Version 1.1

December 10, 2016

Revision History

Date	Changes	Version
12/5/16	First draft	1.0
12/10/16	Completed composition of the doc and made Kearns-requested changes (final draft)	1.1

Table of Contents

1.0 Introduction

Authors: Timothy Wong

This section provides an overview of the entire testing document. This document describes all testing methods and test cases that will be used to test our survey tool.

1.1 System Overview, Purpose of this Document

In this document, we will be going over how we will be testing our survey tool in order to make sure our survey tool is functioning as predicted and to find any faults or defects that may be in our tool. Currently, our survey tool has the ability to create surveys, send surveys, and analyze surveys. It supports the following types of questions: multiple choice, free response, rating, constant sum, ranking, engine created conjoint questions, and manually created conjoint questions.

1.2 Test Approach

In each of the components of the survey we will be checking to see if the components work when everything is inputted by the user correctly as well as the system responses if the user does not correctly input the information. We will be testing different combinations for each test at least five times to make sure the component is working in good order. The tests will be done manually unless otherwise noted.

1.3 References

We will be referencing our previous documents for the survey tool. These include the vision and scope document, software requirements specification document, and design document.

2.0 Test Plan

Authors: Timothy Wong

2.1 Features to be tested

For this test plan, we will be testing three different major features of our survey tool. They are the following:

1. We will be testing the manual three product conjoint analysis questions. These questions will be based off of attributes and respective levels that will have been inputted into the system already by the survey creator. We will only be testing the creation of such questions.
2. We will also be creating test cases for branching logic. This entails making sure the branching for questions work and testing different situations for when a question can branch.
3. Finally, we will be testing our cross tab analysis results. The tests will confirm to us that the analysis and the format of our analysis is in accordance with the specifications and requirements in our SRS document.

2.2 Features not to be tested

We will not be testing the creation and analysis of the multiple choice questions, free response questions, rating questions, constant sum questions, ranking questions, and engine created conjoint analysis questions.

2.3 Testing Tools and Environment

Currently, we will not be using any external testing tools and will be manually testing the three different features listed above in section 2.1. We will be using a non-released beta version of our survey tool to test these different features. However, noted below in section 5.0 Other Testing we may use a third party program to create scripts that will simulate users taking our survey if we are unable to create the scripts ourselves in a timely manner.

3.0 Unit Testing Test Cases

3.1 TC-1 Manual Conjoint Analysis Creation

Authors: Timothy Wong

3.1.1 Identification and Classification

In this test, we will be testing the creation of three product manual conjoint analysis questions. There are two different factors to consider when creating a conjoint analysis question. They are the following:

1. Can add attributes to each option.
2. Can select levels for each attribute.

There are also different user input failures that our system should account for. We will checking the following user input failures and their respective general outputs:

1. Not choosing a specific attribute (does not allow a level to be picked for the attribute)
2. Not choosing a specific level for an attribute

We are testing this component of our survey tool as it is a major feature that we are including that most other survey tools do not implement. It is also important to make sure that this question works because it is unlike any of the other types of questions we include in our survey and so could contain more bugs since it is not a typical type of question people normally implement.

3.1.2 Instructions

<u>Test 1 - User inputs all the correct information:</u>

1. Create new survey.
2. Click “+Add Question” button.
3. Select “Conjoint Analysis” option from the drop down menu of different types of questions.
4. Input desired question into the question description box.
5. Select “Three Choice” option from the drop down menu of different types of conjoint analysis questions.
6. Click “+Add Attribute” button under each option
7. Select attribute from drop down menu of attributes. The attributes should sync with each other and appear in each option.
8. Repeat steps 6 and 7 one to five times for each option set.
9. Select a level for each attribute.
10. Finish survey and view survey in the respondent’s view

Test 2 - User does not specify attribute(s):

1. Create new survey.
2. Click “+Add Question” button.
3. Select “Conjoint Analysis” option from the drop down menu of different types of questions.
4. Input desired question into the question description box.
5. Select “Three Choice” option from the drop down menu of different types of conjoint analysis questions.
6. Click “+Add Attribute” button under each option
7. Repeat step 6 one to five times for each option set
8. Finish survey and view in the respondent’s view

Test 3 - User does specify levels for each attribute(s):

1. Create new survey.
2. Click “+Add Question” button.
3. Select “Conjoint Analysis” option from the drop down menu of different types of questions.
4. Input desired question into the question description box.
5. Select “Three Choice” option from the drop down menu of different types of conjoint analysis questions.
6. Click “+Add Attribute” button under each option. Another attribute line should appear in each option.
7. Select attribute from drop down menu of attributes. The attributes should sync with each other and appear in each option.
8. Repeat steps 6 and 7 one to five times for each option set.
9. Finish survey and view survey in the respondent’s view

3.1.3 Expected Result

Test 1 - User inputs all the correct information:

The user should see one question in the survey that has three options to choose from. The user will be able to pick a “most preferable” option, “less preferable” option, and “least preferable” option in the form of three user input boxes that allow the user to input numbers from 1-3 in each box to rank the options. There should be the same number of attributes in each option as was added when creating the question. Each attribute and the description that corresponds to the attribute’s level should be stated clearly in each respective option box.

Test 2 - User does not input specific attribute(s):

The system should not let the survey creator complete the survey as the creator did not input the specific attribute(s). Upon trying to finish the survey, the system

will give the creator an error message and highlight in red the question number in the top bar where all the question numbers are displayed. In the question, the system will also highlight in red the attributes and levels that have not been filled out and will not allow the survey creator to finish the survey until those fields have been filled out.

Test 3 - User does not input specific level(s):
The system should not let the survey creator complete the survey as the creator did not input the specific levels for their respective attributes. Upon trying to finish the survey, the system will give the creator an error message and highlight in red the question number in the top bar where all the question numbers are displayed. In the question, the system will also highlight in red the attributes and levels that were not filled out and will not allow the survey creator to finish the survey until those fields have been filled out.

3.1.4 Clean-up

After finishing this, the survey link can be disabled and the survey can be deleted from the “view surveys” screen.

3.2 TC-2 Branching Logic Creation and Behavior

Authors: Jeremy Koletar

3.2.1 Identification and Classification

In this test, we’re testing the behavior of branching logic. Namely, we want to make sure that branches can be taken and untaken, and that the basic chaining of conditionals functions properly.

Preconditions: A mailing list has been created. The mailing list contains at least three respondents. Each respondent’s email address should be accessible by the tester. An empty survey has been created.

3.2.2 Instructions

Setup

1. Add a multiple choice question with three possible responses: “Branch now”, “Pass”, and “Fail”.
2. Add a multiple choice question with two possible responses: “Pass dependent branch” and “Fail dependent branch”.
3. Add a free response question with the prompt “No branching occurred.”
4. Add a free response question with the prompt “Branched immediately from question 1.”

5. Add a free response question with the prompt “Branched from question 2 by passing a combined conditional.”
6. Navigate to question 1, and tick the branching logic checkbox.
7. Add a new row of branching logic. Select Question 4 as the destination question, leaving the “next conditional” field blank. Select question 1 as the test question. The “conditional” field should be set to “equal to”, and the “test value” field should be set to “Branch now”.
8. Navigate to question 2, and tick the branching logic checkbox.
9. Add two rows of branching logic.
10. For the first row of branching logic, do as follows: Leave the destination question blank. In the “next conditional” field, from the dropdown menu, select the *down arrow* icon. Select question 1 as the test question. The “conditional” field should be set to “equal to”, and the “test value” field should be set to “Pass”.
11. For the second row of branching logic, do as follows: Set the destination question to question 5, leaving the “next conditional” field blank. Select question 2 as the test question. The “conditional” field should be set to “equal to”, and the “test value” field should be set to “Pass dependent branch”.
12. Publish the survey to the group of test respondents.

Test 1 - No branching

1. Click on an unused link sent to a test repondent account.
2. Select “Fail” for the first question.
3. Select “Pass dependent branch” for the second question.

Test 2 - Immediate branching

1. Click on an unused link sent to a test repondent account.
2. Select “Branch now” for the first question.

Test 3 - Dependent branching

1. Click on an unused link sent to a test repondent account.
2. Select “Pass” for the first question.
3. Select “Pass dependent branch” for the second question.

3.2.3 Expected Result

Test 1 - No branching

After submitting “Pass dependent branch” for the second question, the free response question with the prompt “No branching occurred” should be shown.

Test 2 - Immediate branching

After selecting “Branch now” for the first question, the free response question with the prompt “Branched immediately from question 1” should be shown.

Test 3 - Dependent branching
After selecting “Pass dependent branch” for the second question, the free response question with the prompt “Branched from question 2 by passing a combined conditional” should be shown.

3.3 TC-3 Cross Tab Analysis

Authors: Patrick Cook

3.3.1 Identification and Classification

In this test, we will be testing the functionality of our cross tab analysis. There are two different cases that our system will need to handle. They are the following:

1. Two questions are selected for cross tab analysis
2. A third question is added to the analysis which provides additional statistics

There are a few system failures that we must account for which include the following:

1. Attempting to analyze a question that cannot be quantified (ex. Text input)
2. System alerts user if less than 2 questions are selected for analysis

3.3.2 Instructions

Test 1 - User selects two questions to be used for analysis:

1. Click ‘View Survey’ tab on the home page
2. Click ‘Cross Tabulation’ option
3. Select first question to use in analysis by selecting the question number from the drop down menu
4. Select the second question to use in the analysis by selecting the question number from the drop down menu
5. Repeat steps 3 and 4 to view crosstab analysis using other questions

Test 2 - User selects three questions to be used for analysis:

1. Click ‘View Survey’ tab on the home page
2. Click ‘Cross Tabulation’ option
3. Select first question to use in analysis by selecting the question number from the drop down menu
4. Select the second question to use in the analysis by selecting the question number from the drop down menu
5. Select the third question to use in the analysis by selecting the question number from the drop down menu which will used by the system to provide analysis using all three questions instead of only two questions

6. Repeat steps 3-5 to view cross tab analysis using other questions

Test 3 - User does not select at least 2 questions for analysis:

1. Click ‘View Survey’ tab on the home page
2. Click ‘Cross Tabulation’ option
3. Select first question to use in analysis by selecting the question number from the drop down menu
4. User does not select a second question for analysis

Test 4 - User selects question type that cannot be used for analysis:

1. Click ‘View Survey’ tab on the home page
2. Click ‘Cross Tabulation’ option
3. Select first question which is of the question type ‘text input’ to use in analysis by selecting the question number from the drop down menu.
4. Select second question which is of the question type ‘text input’ to use in analysis by selecting the question number from the drop down menu.

3.3.3 Expected Result

Test 1 - User selects two questions to be used for analysis:
The system will present a 2x2 grid to the user containing data which corresponds to the two questions selected by the user. The options from question 1 will be listed on the left column and the options from question 2 will be listed on the top row. The boxes in between will show the percentage of respondents who answered the respective options.

Test 2 - User selects three questions to be used for analysis:
The system will present a 2x2 grid to the user containing data which corresponds to the two questions selected by the user. The options from question 1 will be listed on the left column and the options from question 2 will be listed on the top row. The boxes in between will show the percentage of respondents who answered the respective options. The third question in the bottom will be an optional question that can be tested against each box in the table.

Test 3 - User does not select at least 2 questions for analysis:
The system will not perform the analysis if only one question is selected. When/If the user selects another question to perform the analysis with then this case will follow the behaviour of test case 1 above. The analysis will be performed and the 2x2 grid will be populated with the results.

Test 4 - User selects question type that cannot be used for analysis:

The system does not support cross tab analysis with questions that do not have quantitative responses. This includes free response and text input question types. The system will alert the user that they have selected an incompatible question type and will not continue with the analysis.

4.0 Integration Testing Strategy

Author: Josh Pfeffer

4.1 Bottom-Up Testing

With this approach, we will test the lower-level components first. This is advantageous because it, in theory, will help us find bugs more easily in our system.

We will utilize this testing strategy by testing low-level components first, such as database interactions (e.g. fetching/retrieving data) and communication among classes. We will gradually test higher level functionality, such as individual class tests along with various UI tests that will test the system's error detection if a user does not input data correctly.

4.2 Top-Down Testing

With this approach, we will test the higher-level components first. This method will allow our team to find a missing link between components more easily.

Per the top-down testing model, Team Costco will collectively test high-level components first--such as user input error detection--when using this strategy. Once these tests are complete and validated, we will test individual class functionality. Then, we will test lower level interactions, such as communication between classes and various database interactions.

5.0 Other Testing

Authors: Timothy Wong

5.1 Performance Testing

Our performance test will be the test of our system under normal use. We will be defining normal use with respect to three different aspects of our system:

1. Number of questions in a survey.
2. Number of respondents taking a survey.
3. Number of active surveys in use.

The number of questions we estimate a survey to have normally would be 25 questions. This is assuming that our survey creator is utilizing engine created conjoint analysis questions as they will comprise of more than one question.

The number of respondents we estimate to taking surveys at one time would normally be around 400 respondents. While many more than 400 respondents will be asked to take surveys, we do not expect all of the respondents to take their respective surveys at the same time.

The number of active surveys we expect to be active at one time on average would be around 20 active surveys at one time. Note that we are not accounting done surveys as they should not be much of a burden on the system since only one or two analysts would be analyzing a survey at one time.

Our performance test will be focused on the amount of time it takes for our system to fetch questions for a survey and display that question for a respondent. From the respondent's view, this equates to how long it takes to get from question to question. We consider this test to be passed if it takes less than two seconds to get from question to question.

5.2 Stress Testing

Our stress tests will consist of testing three different aspects of our system. They are the following:

1. Number of questions in a survey.
2. Number of respondents taking a survey.
3. Number of active surveys in use.

Test 1 - Number of questions:

In this test, we will be testing the upper bound of the number of questions a single survey can have and how well our system can output the questions to the respondent. The requirement that we are putting on our system is that our system should be able to handle a survey that has at least 150 questions. We will do this by first performing the performance tests above and then creating a survey with around 150 questions and comparing how long it takes to get from question to question. If the difference between times is under two seconds between a normal performance and our stressed performance, then we consider this test to be passed.

Test 2 - Number of respondents:
In this test, we will be testing the upper bound of the numbers of respondents our system can handle at one time. The requirement that we are putting on our system is to be running when 1,000 users are using our tool at the same time. We will accomplish this test by using a script to simulate the traffic our tool would encounter if around 800 users were answering surveys simultaneously. Currently, we will be writing these scripts ourselves, but we may use a third party program to create these scripts if complications arise. The success of this test will be determined by the difference in the time it takes to move from question to question in the performance test and this test. If the difference in times is under five seconds, then we consider this test to be passed.

Test 3 - Number of surveys in use:
In this test, we will be testing the upper bound of the number of surveys in use our system can handle at one time. The requirement that we are putting on our system is that it can manage at least 40 active surveys at a time. We will accomplish this test by creating 40 active surveys and having a script to simulate an average amount of people answering the surveys at one time. Similarly in test two, we will be writing these scripts ourselves, but we may use a third party program to create these scripts if complications arise. The success of this test will be determined by the difference in the time it takes to move from question to question in the performance test and this test. If the difference in times is under three seconds, then we consider this test to be passed.

5.3 Other Tests

We will also test that our system works on the different browsers mentioned in our Software Requirements Specification document. This test can be accomplished by trying to create, respond to, and analyze surveys using our tools on the different browsers.

6.0 Test record keeping and test log

Authors: Timothy Wong

Test	Date Tested	Passed?
3.1 TC-1 Manual Conjoint Analysis Creation: Test 1		
3.1 TC-1 Manual Conjoint Analysis Creation: Test 2		
3.1 TC-1 Manual Conjoint Analysis Creation: Test 3		
3.2 TC-2 Branching Logic Creation and Behavior: Test 1		
3.2 TC-2 Branching Logic Creation and Behavior: Test 2		
3.2 TC-2 Branching Logic Creation and Behavior: Test 3		
3.3 TC-3 Cross Tab Analysis: Test 1		
3.3 TC-3 Cross Tab Analysis: Test 2		
3.3 TC-3 Cross Tab Analysis: Test 3		
3.3 TC-3 Cross Tab Analysis: Test 4		
Performance Test		
Stress Testing: Number of questions in a survey		
Stress Testing: Number of respondents answering		
Stress Testing: Number of active surveys		