Queensway Taxis Case Study

Difficulty Rating: Intermediate

25 June 2015

KnowledgeKube

Introduction

Queensway Taxis is a private hire firm that operates a fleet of cars in Birmingham, England. Their current booking system relies on staff members taking customer requests by phone and writing down the details using a pen and paper - a system that is both slow and prone to user error. The firm has decided to commission an electronic application built using KnowledgeKube in order to book journeys faster and more efficiently.

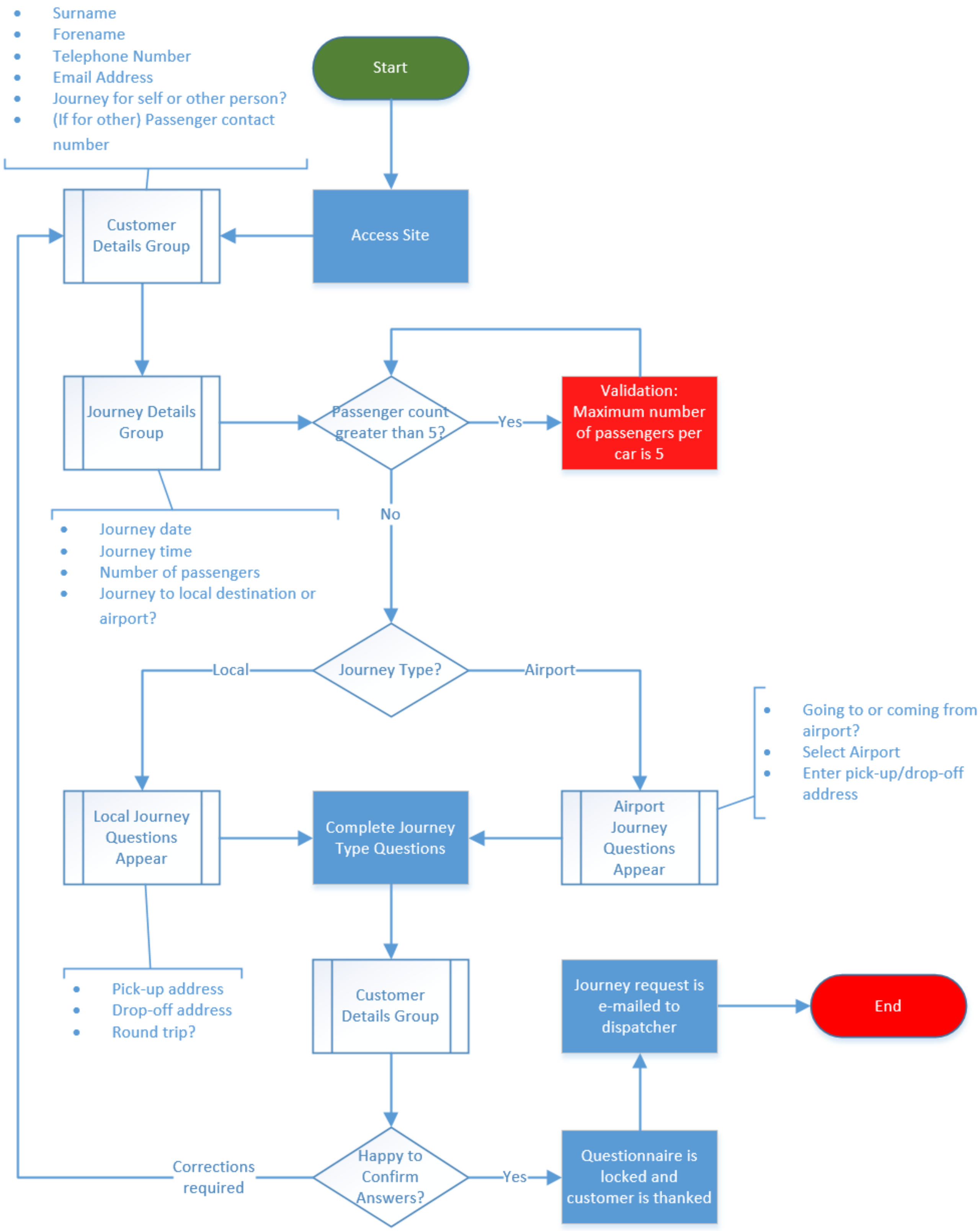
Surname
Forename
Telephone Number
Email Address
Journey for self or other person?
(If for other) Passenger contact number
Start
Access Site
Customer Details Group
Journey Details Group
Passenger count greater than 5?
Yes
Validation: Maximum number of passengers per car is 5
No
Journey date
Journey time
Number of passengers
Journey to local destination or airport?
Journey Type?
Local
Airport
Going to or coming from airport?
Select Airport
Enter pick-up/drop-off address
Local Journey Questions Appear
Complete Journey Type Questions
Airport Journey Questions Appear
Pick-up address
Drop-off address
Round trip?
Customer Details Group
Journey request is e-mailed to dispatcher
End
Corrections required
Happy to Confirm Answers?
Yes
Questionnaire is locked and customer is thanked

Figure 1-1: The proposed Queensway Taxis questionnaire flow.

Building the Basic Model Structure

The Queensway Taxis questionnaire model will contain five question groups, two of which will be used to gather the following information:

- The customer's contact details including their name, address, email address and contact number.
- Details of the intended journey, including a date and time, number of passengers and whether the person filling in the questionnaire is requesting a journey for themselves or a third party.

The remaining groups will serve as two supplemental question sets and a summary page, respectively.

Creating the Initial Question Groups

Start by adding two **Form**-type question groups to your model. The first will be used to gather customer contact details, while the second will capture the customer's journey requirements. Give each group a name that suits and identifies its purpose.

As with all new question groups, your forms will be **Checked In** and unavailable for edit, so make sure you check them out before attempting to edit them. Remember to check all question groups in afterwards, to get yourself into the habit of making content available to other editors when you have finished with it.

Before you start to populate your forms with questions, you should add **Navigation Buttons** that will allow people to move back and forth between the groups. Queensway Taxis have asked for the contact details form to appear first, so make sure you set this form as your model's **Startup Group**.

Adding Questions and Responses

Now you have the foundation of your questionnaire, you can start to create content for the end user to read and fill in. You have been asked to add questions to the **Contact Details** form that collect the following information about the passenger:

Information	Question Type
Surname	Free Text
Forename	Free Text
E-mail address	Email Address
Contact Number	Number
Whether the person filling in the questionnaire is booking on behalf of themselves.	Yes/No

The **Journey Details** form, meanwhile, should contain questions that gather the following information about the intended journey:

Information	Question Type
Journey date	Date
Journey time	Time
Number of passengers	Number
Whether the journey is to a local destination or an airport.	Multiple choice

Set the default value of the passenger number question to "1", and add the following responses to the multiple choice question:

- -- Please Select One --
- Local Fare
- Airport Fare

Ensure that the first response is set as the question's default answer, and that it cannot be left as the chosen response by the person filling in the questionnaire. The response given here will determine the remaining questions asked of the customer, as described in the next section of this case study.

Add all questions in your model to the **Expression Parser**, so their values can be used later when you create your questionnaire's summary form. Next, add narrative questions to your forms, so Queensway's customers know what they are filling in, and why.

Enhancing the Questionnaire

Now you have a basic questionnaire structure, you are going to add a layer of automation to your model. This stage includes the following:

- Creating nested question groups for displaying questions relevant to the type of journey being booked.
- Using a validator to ensure the number of passengers does not exceed the maximum vehicle capacity available.
- Using dynamic visibility to hide content that is irrelevant to the customer's journey.
- Overriding button validation to allow the customer to return to an earlier form without completing the current form.

Creating Nested Question Groups

Add two further **Form**-type question groups, one to request details about local fares, the other concerned with airport fares. When the model is complete, only one of these forms will appear to the customer at any one time, with the visible form dependent on the answer given to the multiple choice question under **Journey Details**. Both of these new groups should be assigned two columns, and all constituent questions in need of a response should be added to the expression parser.

Because these new question groups are to be nested inside the **Journey Details** group, there is no need to add navigation buttons to either of them.

The local fares group should contain **Free Text** questions to gather the house number, street and postcode of both the pick-up and drop-off locations. Assign each set of questions to a different column in the question group, and write a **Narrative** questions above both that make it clear to the customer which one is which. Lastly, add a **Checkbox** question that asks whether the journey will be a round trip that returns to the pick-up location.

You may find it easier to add **Free Text** questions to this group using the **Create Questions (From CSV)** method.

The airport fares group requires two **Multiple Choice** questions:

- The first question asks whether the journey to the airport is to pick the passenger up or to drop them off. Add two suitable responses to this question.
- The second question asks the customer to name the airport to which the taxi will drive. Add a suitable list of airport names as responses to this question.

Add a default place holder response to both of these questions, and ensure that in each case the default response cannot be left as the customer's chosen answer.

In addition to multiple choice questions, the airport fares group should also include a set of free text questions for gathering the address at the other end of the airport journey. This will serve as either the pick-up or drop-off location, depending on the customer's response to the first multiple choice question described above. Place the free text questions in a separate column to that of the multiple choice ones.

Finally, add two **Place Holder** questions to the **Journey Details** form, and add an attribute to each so that one of them nests the local fares group, and the other one nests the airport fares group.

Applying Visibility Expressions

To ensure that the correct journey type questions are asked, you need to add **Visibility Expression** attributes to the two place holder questions found in the **Journey Details** form. By applying dynamic visibility to the place holders instead of individual questions, either the **Local Journey** group or the **Airport Journey** group will be hidden entirely. To achieve this, ensure the visibility attributes use the following logic:

Place holder	Only appear when...
Airport Journey	Customer selects **Airport** as their journey type.
Local Journey	Customer selects **Local** as their journey type.

Lastly, add a new **Number**-type question to the **Customer Details** form. This question should have a visibility expression attribute that ensures it only appears if the customer has indicated they are ordering a taxi on behalf of someone else. The purpose of this question is to record the contact telephone number of the person for whom the taxi is being booked.

Remember that in order for a visibility expression to work correctly, the respective trigger question must be added to the expression parser and made to post back to the web server.

Adding a Validator

The largest cars in the Queensway Taxis fleet can seat a maximum of five passengers. Due to this fact, the customer should not be able to proceed from the **Journey Details** form if they have specified a number of passengers greater than five.

Create a **Validator** in the journey details form, and apply a validation attribute that checks to see whether the customer has entered a valid number of passengers. The validation text should make it clear to the customer that there is a limit to the number of passengers, letting them make the necessary adjustments before they proceed.

Overriding Button Validation

As the customer completes the questionnaire, they may wish to return to the **Customer Details** form before they have finished filling in the **Journey Details** form. Because all of the journey details questions are mandatory, attempting to leave the form without completing all questions would normally cause a validation error. To get around this, apply a **Cause Button Validation** attribute to the "Back" navigation button, and have it evaluate to 0 (or *False*).

The "Next" button will be left as-is, ensuring that validation takes place when the customer attempts to progress to a new page in the questionnaire.

Concluding Your Questionnaire

When the customer has finished filling in your questionnaire, they should be provided with some feedback. Specifically, they will be presented with a form that summarises their stated journey requirements and gives them the option to go back and change those requirements if necessary.

As soon as the customer is happy that their requirements are correct, they should click a button to both confirm this fact and submit their order. Submitted orders will be e-mailed to Queensway Taxis, who can then arrange for a taxi to be dispatched.

A second e-mail will be sent to the customer, confirming their order and letting them know that as soon as a taxi is assigned to them the driver will be in touch by phone.

Writing a Confirmation E-mail Template in CDS

When a user confirms their booking, an e-mail will be sent to both the customer and the dispatcher at Queensway Taxis. Both e-mails will simply confirm the details of the customer's journey request, and as such both can be sent using the same e-mail template. This template will be created using the Mercato **Content Delivery System** application, and called upon when the customer submits their request.

Refer to the separate CDS documentation for more information about creating CDS e-mail templates.

When you create your template, ensure that the e-mail's body includes references to every question relating to the requested journey. Ensure that the information is laid-out in a logical and readable manner. Give the template a suitable name so you can easily refer to it on your questionnaire's summary page.

Building Your Summary Form

Create another **Form**-type question group to serve as your questionnaire's summary. Place a narrative at the top of the page to thank the customer for choosing Queensway Taxis. The narrative should also ask the customer to review their journey details before either returning to an earlier page to make corrections, or confirming they want to proceed with the taxi booking.

Next, add a series of Read-Only Text items to the summary. Each one should correspond to a question concerning the customer's journey. Using attributes, map the answer of each question to the corresponding read-only text. There is a lot of information to display, so consider carefully the order in which the answers should appear, and whether or not the summary would be best split into multiple columns.

Add another navigation button to the Journey Details form. The customer should be able to use this button to progress to the Summary Form. Then add a navigation button to the summary form to allow the customer to return to an earlier part of the questionnaire to change their answers as needed. Give the second button a cpation that makes it clear the customer can change their answers.

Finally, add a regular button to the bottom of your summary page. This button allows the customer to submit their journey request, resulting in the following:

- The buttons for returning to an earlier part of the questionnaire and submitting the journey request should be disabled using **Enable Expression** attributes, to prevent changes being made or the request being submitted a second time.
- Two e-mails should be sent using *CMSEmail* expressions. Both e-mails should be based on your CDS template, with one sent to the customer and the other sent to a dedicated e-mail address at Queensway Taxis.
- An additional narrative question should appear, confirming the journey request has been submitted and thanking the customer for choosing Queensway. A suitable **Visibility Expression** will need to be applied to this narrative after you have created it.

> Enable expression and visibility expression attributes rely on a particular condition being met. One of the best ways of implementing them in this example is to create a variable whose value starts off as 0, and changes to 1 when the customer submits their request. Both types of attribute can rely on the value of this single variable.

Create all of the questionnaire elements required to fulfil the submission process. Don't forget you can use semicolons to create **Multi-line Expressions** that accomplish all of the above with a single button click.

[THIS PAGE INTENTIONALLY LEFT BLANK]

Actions	
Attributes	✓
CDS	✓
Data Processing	
Data Source Grids	
Data Sources	
Documents	
E-Mail Templates	✓
Language Translations	
Model Scheduler	
Model Variables	✓
Rating Definitions	
Repeating Question Groups	
States	
Users/Roles/Workgroups	
Validators	✓

Abstract

The Queensway Taxi Company case study is intended to demonstrate how KnowledgeKube can be used to create useful, real-world applications.

By following this study, you will be shown how to construct a simple booking system for a fictional taxi firm. It will allow users to book taxis for local and airport journeys.

The implementation of this model involves the use of question attributes that change the flow of the questionnaire according to user responses, and deploys external CDS resources to generate dynamic e-mails.

Prerequisites

In addition to an understanding of the KnowledgeKube features covered herein, you will need the following in order to complete this case study:

- Access to the Mercato **Content Delivery System** application.
- The ability to create a dynamic e-mail template using CDS.