



AguilarTech.io

A CONVERSATION BOT TO HANDLE RETURNS

2 November 2022

Daniel Aguilar

AguilarTech.io

+61 432 992 862

daniel@aguilartech.io



Task 1: Power Virtual Agent - Building a Conversational Bot to handle return

STEP 1 – SETTING UP THE TASK

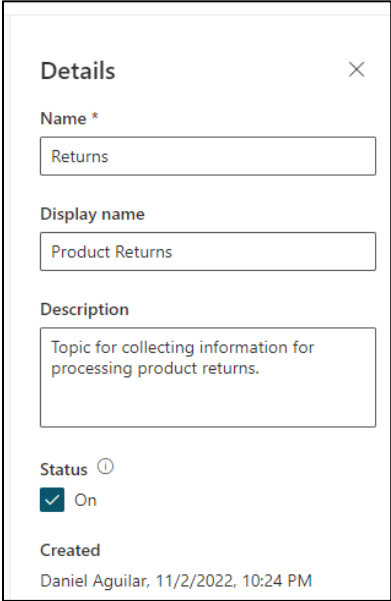
The first step that I took was to understand the requirements and connect that to how I will go about fulfilling the requirements, making sure that I understand what is required end to end without missing anything.

Then I connected these requirements to the best way to fulfill them using the required tech stack. Which in this case is Power Virtual Agent and Power Automate in the Microsoft 365 suite of apps.

I also made sure at this point to ensure that I am up to date with how to best use Power Virtual Agent to its full feature capability, by reading the documentation and watching the demo guide videos.

STEP 2 – INITIALIZING THE CHATBOT

Using Power Virtual Assistant I create a new bot instances, gave it the name of “Returns” and a good description; ensuring that there would be no confusion with future colleagues as to what this bot does. I also checked that the correct settings were selected for the current user case scenario.



The screenshot shows a 'Details' dialog box for a Power Virtual Agent bot. The dialog has a close button (X) in the top right corner. It contains the following fields and settings:

- Name ***: Returns
- Display name**: Product Returns
- Description**: Topic for collecting information for processing product returns.
- Status**: On (checked)
- Created**: Daniel Aguilar, 11/2/2022, 10:24 PM

By following examples in the Microsoft documentation I carefully selected the best trigger phrases that I could think of to match the topic of returns.

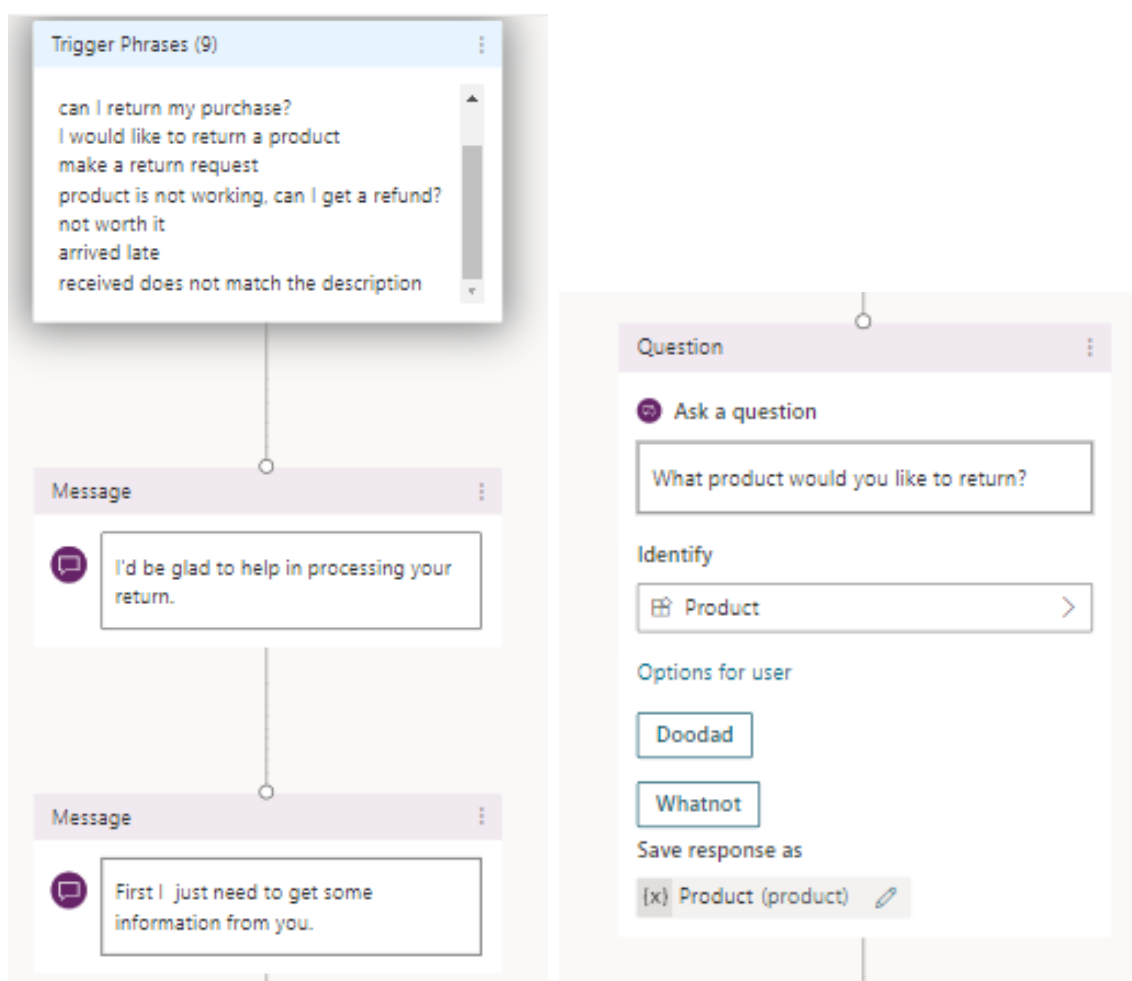
Trigger phrases (9)
return
how to return?
can I return my purchase?
I would like to return a product
make a return request
product is not working, can I get a refund?
not worth it
arrived late

I did this by brainstorming as many ways as possible that users might communicate an intention to request a product return. This, I believe, will minimize any issue with the bot not being able to understand human user intent and therefore ensuring user experience remains high.

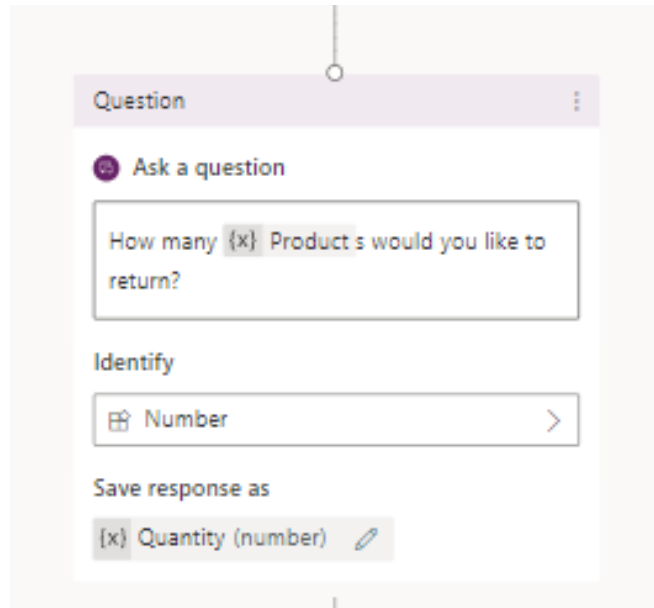
STEP 3 – THINK OF THE USER EXPERIENCE

The next step for me was to stand back and plan out how to maximise the experience for the user through a returns request process by making it as seamless as possible, asking the minimal amount of back and forth between bot and human.

I decided that the best start to the process would be to first give acknowledgement feedback to the user that the returns request process has started and to first ask for the name of the product that the user would like to return.



Knowing this we can then ask the user what quantity they would like to return by using the product name string variable in this question. I think this improves the user experience by being clearer on exactly which product is being returned i.e “How many whatnots would you like to return?” instead of “what quantity would you like to return?”



STEP 4 – CREATED ENTITIES

To ensure data integrity and understanding Power Virtual Agent uses 'entities' that act like data prototypes. I created an entity for 'Product' as a Closed List of the two products: 'doodad' and 'whatnot'. I created Synonyms for these by splitting these into two words and adding a plural version for each. I did this to make sure I am capturing as many variations of what users might type when referencing the products as possible.

Product							
<p>Name *</p> <input type="text" value="Product"/>	<p>List items</p> <p>Enter item <input type="text"/> <input type="button" value="Add"/></p>						
<p>Description</p> <input type="text" value="Company Products"/>	<table border="1"> <thead> <tr> <th>Item</th> <th>Synonyms</th> </tr> </thead> <tbody> <tr> <td>Doodad</td> <td>doodads, doo dad</td> </tr> <tr> <td>Whatnot</td> <td>whatnots, what not</td> </tr> </tbody> </table>	Item	Synonyms	Doodad	doodads, doo dad	Whatnot	whatnots, what not
Item	Synonyms						
Doodad	doodads, doo dad						
Whatnot	whatnots, what not						
<p>Method</p> <p>List</p> <p>The bot will try to match an item on the list based on what the customer says.</p>							

I also created an entity for the list of Return Reasons. Also making sure that these options had synonyms to capture variations.

Return Reason

Name *

List items

Description

Method

List

The bot will try to match an item on the list based on what the customer says.

Modified by

5 hours ago

Smart matching



on

Item

Synonyms

damage or broken

working, does not work, not work

does not match description

inaccurate description

I do not like the item

disappointed

poor value

overpriced

arrived late

delay

STEP 5 – ENSURE ROBUSTNESS

After this I continued to create the flow for the process of the bot going through asking all of the pieces of required information: product, quantity invoice number, email, reason for return.

When creating each of this I double checked data integrity by ensuring that each step was in the correct order, the correct type and that the correct entity type was being used for each question. This will ensure that there are no bugs and that the flow runs exactly as intended.

The image displays three examples of a 'Question' form component in a mobile application. Each form has a title bar 'Question' with a three-dot menu icon on the right. Below the title bar is a section 'Ask a question' with a speech bubble icon. The main input area contains a text field with a question. Below the text field is an 'Identify' section with a dropdown menu. At the bottom is a 'Save response as' section with a text input field and an edit icon.

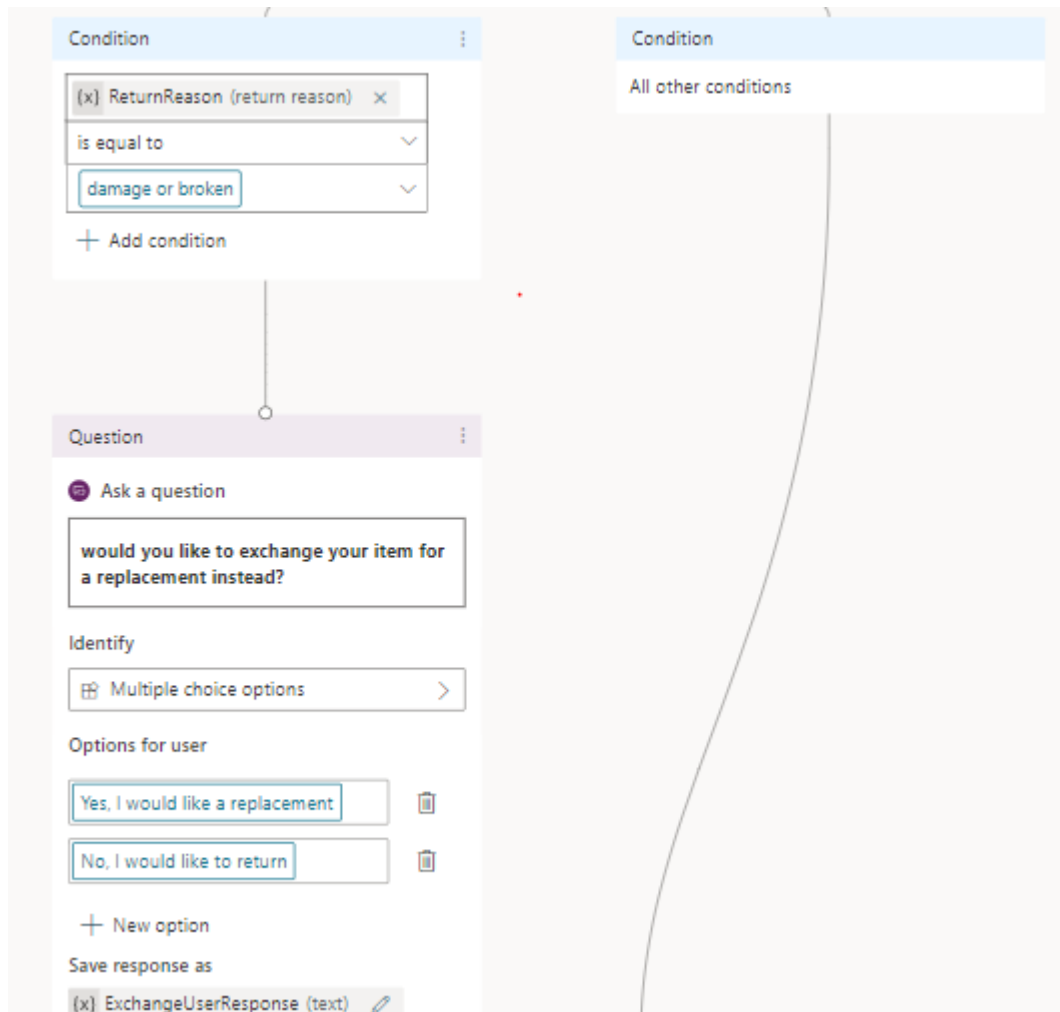
- Form 1:** Question: 'What is your Invoice Number?'. Identify: 'Number'. Save response as: '{x} InvoiceNumber (number)'. Edit icon.
- Form 2:** Question: 'What is your Email Address?'. Identify: 'Email'. Save response as: '{x} Email (email)'. Edit icon.
- Form 3:** Question: 'What is the reason for your return?'. Identify: 'Return Reason'. Options for user: 'damage or broken', 'does not match description', 'I do not like the item', 'poor value', 'arrived late'. Save response as: '{x} ReturnReason (return reason)'. Edit icon.

STEP 6 – ENSURE CLEAN & EFFICIENT CODE

I also made sure to name the variable with descriptive names and in a consistent format such as camel case. This will also help this flow be able to be worked on and debugged by others easily in the future.

STEP 7- GIVING THE USER THE OPTION TO ACCEPT AN EXCHANGE

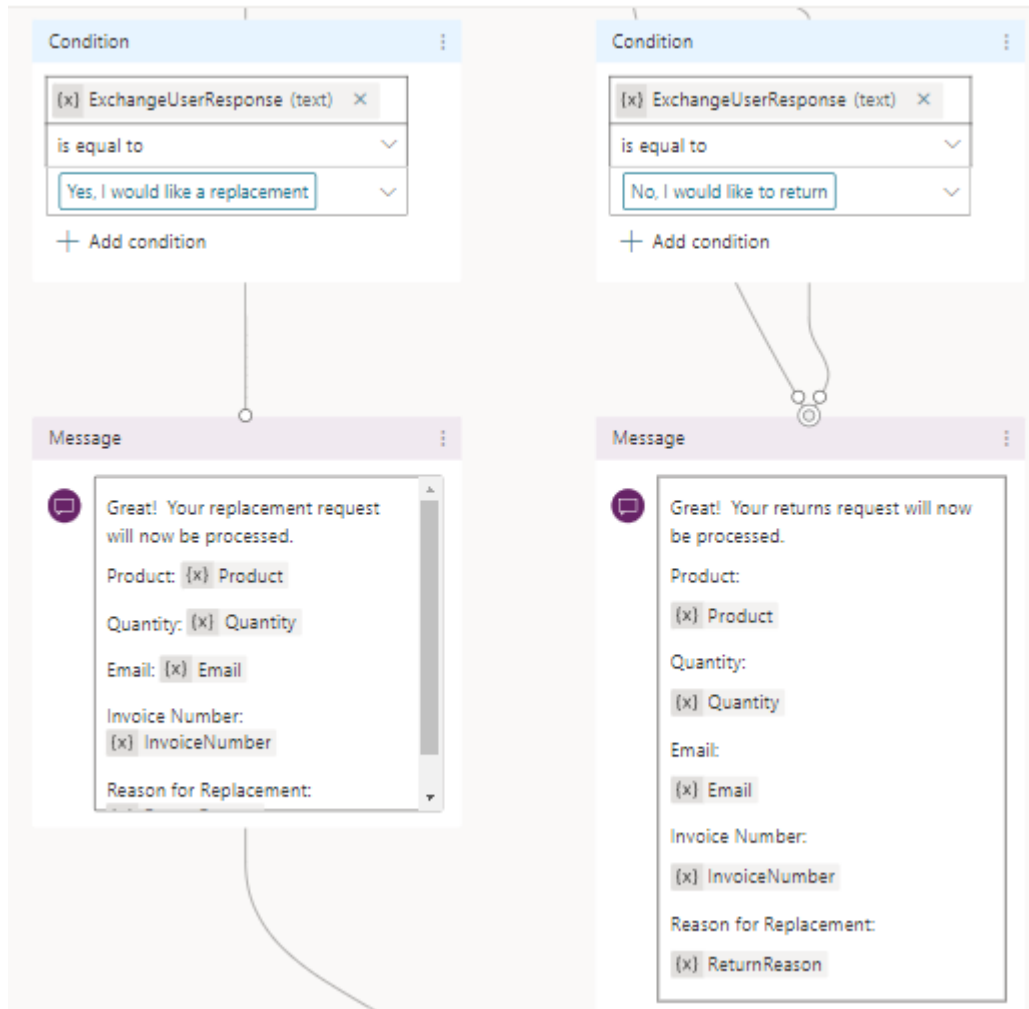
One of the requirements of the task is to encourage to user to accept a return if the goods are damaged. I created this conditional if the user selects that the reason for the return is due to damage. The user's response to this question is recorded in the variable: *ExchangeUserResponse*. Which also sends the user down a tree of whether they are to receive a 'replacement' or a remain a 'return'.



If the user, having selected 'damaged' as the return reason confirms that they would like to continue with their 'return' then I joined the flow back with the main flow for all other return reasons. The reason for doing this was to eliminate code repetition.

STEP 8– FINILIZING THE CHATBOT TOPIC

As the last step of the flow when all required information has been collected by the chat bot I created a message back to the users confirming the details of their return. This will also be good for user experience as users will get some feedback and confirmation of their return request.

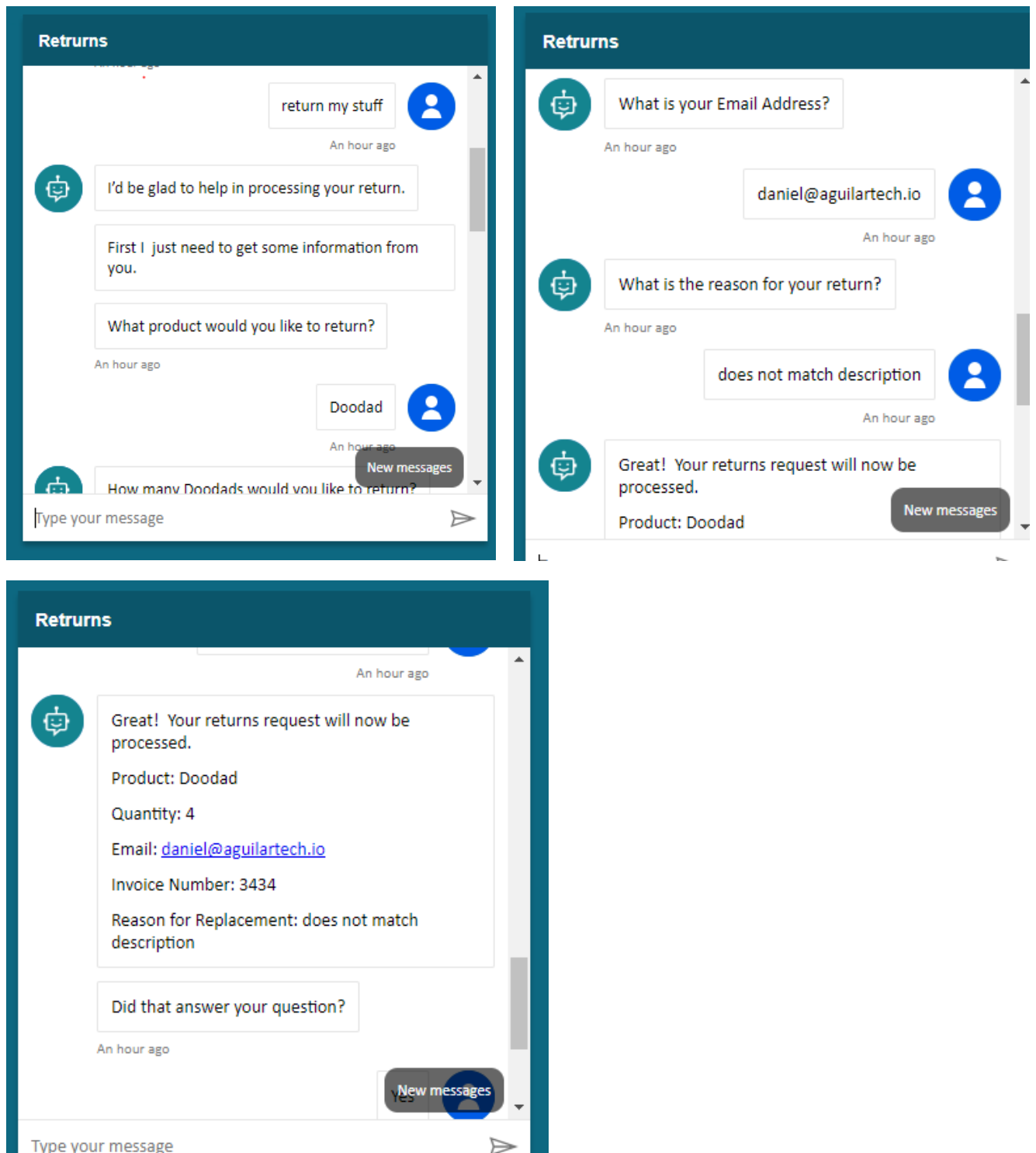


STEP 9– TESTING, TEST PUBLISHING AND ITERATING

After completing the first iteration of the bot, I tested it with as many different variations of what users might say and do as possible. Seeking for any bugs or problems with the flow, any spelling mistakes or incorrect steps.

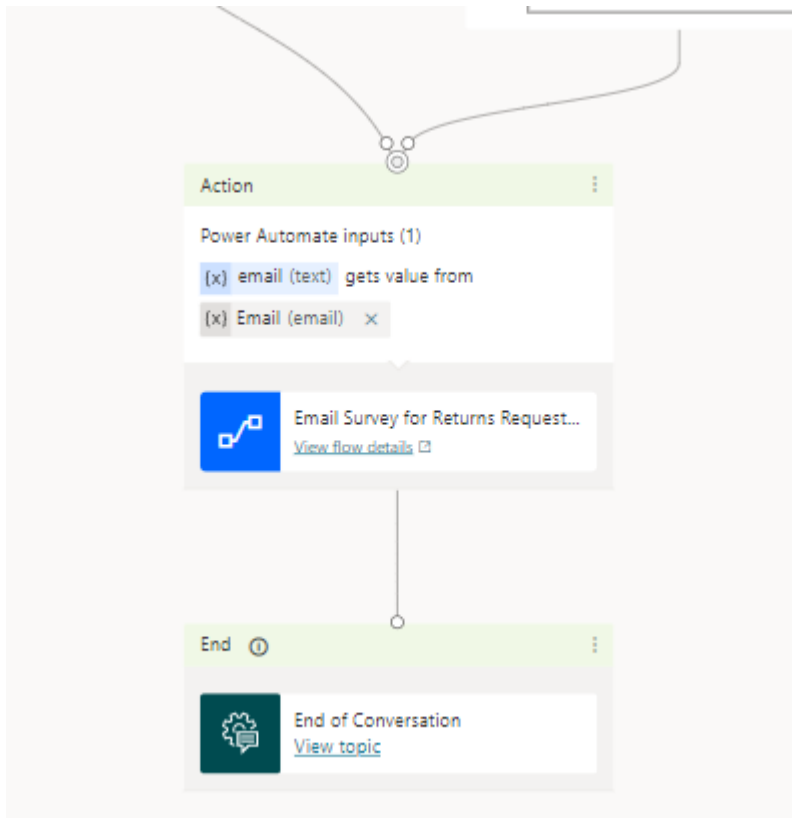
I only found one spelling mistake and went back and corrected that.

To finalize, I took one last check of the task requirements, checking that all requirements had been completed and I was ready to present the Chatbot to the customer.



Task 2: Power Automate - Sending out a survey form

Reading the requirements of task 2, I understood it to require me to create a Power Automate flow that gets triggered at the end of a Chatbot session requesting a return.



The flow simply needed to send a link to a survey to the email of customers that had submitted a return request so I created an input variable for 'email' that gets passed down from the email collected by the bot.

This 'email' variable does not need to be validated as a valid email as it has already been validated by the "Email" entity. Which means that it would only accept valid emails at the stage of the chat conversation where the bots requests an email from the user.

Using this email variable, I created a “Send Email” action of the Office 365 Outlook connector to send an email with a link to the Microsoft Forms survey asking them to rate their experience.

The screenshot shows the configuration of a 'Send an email (V2)' action in Power Automate. The 'To' field is populated with the 'email' variable. The subject is 'Thank you for your return request'. The body text is as follows:

Dear Customer,
Thank you for your return request.
Please complete the below survey to rate your experience:
[Customer Experience Survey](#)
Thank you

The email being received:

The screenshot shows an Outlook inbox with the following email:

Focused | Other 1 | Filter

Other: New conversations
Microsoft Viva

Daniel Aguilar
> Thank you for your return req... 02:10
Dear Customer, Thank you for your ret...

Thank you for your return request

DA Daniel Aguilar
To: Daniel Aguilar

Dear Customer,
Thank you for your return request.
Please complete the below survey to rate your experience:
[Customer Experience Survey](#)
Thank you

Questions

Responses

Returns Experience Survey

1. How likely are you to recommend our company after your experience with us today? *

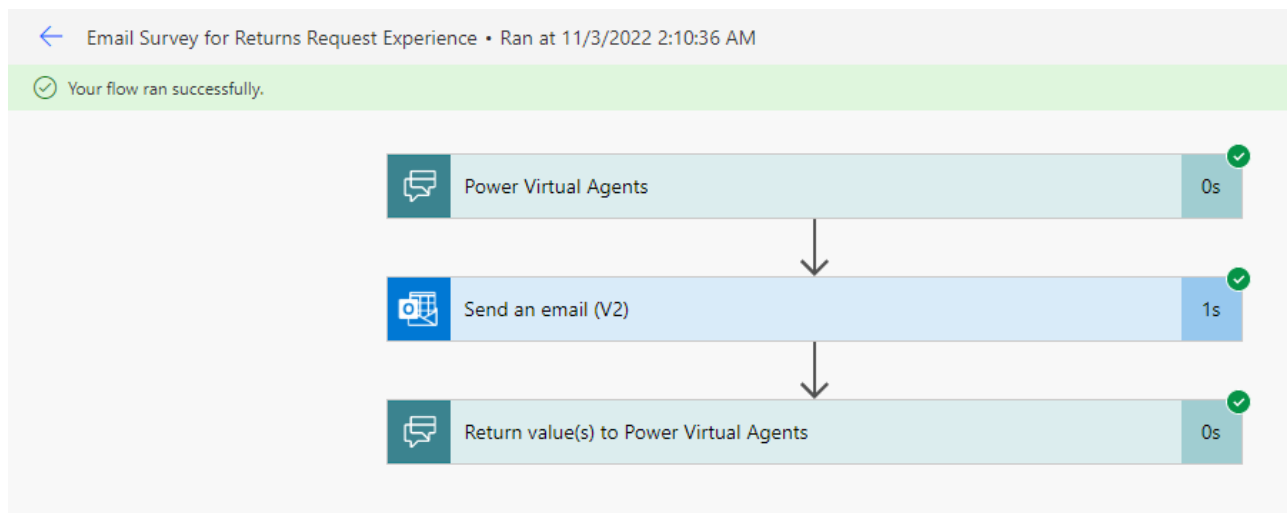
0	1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	---	----

Not at all likely

Extremely likely

+ Add new

Testing the Power Automate flow by running a chat bot returns request:



Having tested all parts of each of the two tasks and double checking the assessment required I believe that the tasks are finalized at this point.

The links to the demos are:

[Power Virtual Agent Demo Site](#) & [Experience Survey](#)

Ps. My Outlook is not set up for outgoing SMTP through O365. But I am happy to provide confirmation that the **flow** has run in a test.

Assessment by

Daniel Aguilar

+61 432 992 862

daniel@aguilartech.io