Zero to Hero with Practical Use Cases

Power Virtual Agents

PRIYARANJAN KS

Contents

Getting Started with Power Virtual Agents

Introduction

Power Virtual Agents helps us create intelligent conversational chat bots without any code implementation. With these bots we can engage with customers and employees in multiple languages across websites, mobile apps, Facebook, Microsoft Teams, or any channel supported by the Azure Bot Framework

PVA lets us create powerful chatbots that can answer questions posed by customers employees, or external visitors which eliminates the need for a dedicated manual intervention to resolve questions. It also helps in creating an automated data acceptance channel so that with a single bot numerous users can input the data which can be used for triggering a back end business workflow.

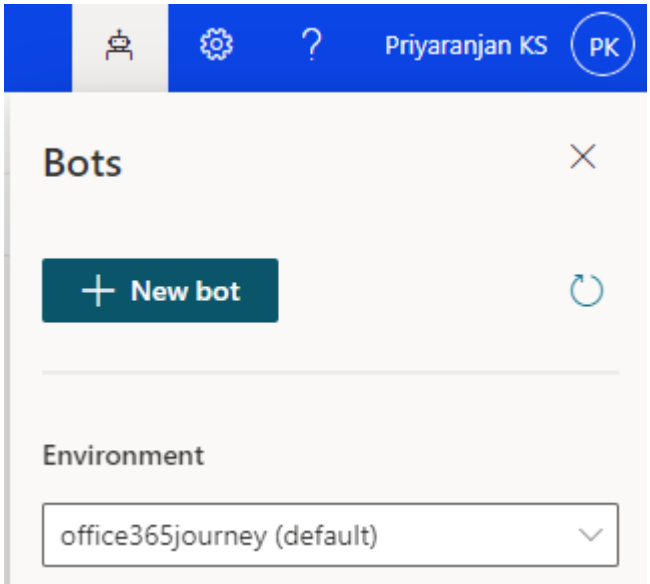
Some of the use cases for Power Virtual Agents Include and not limited to ：

- Understand the Opening and closing hours of a store/hospital/institution
- Fetch the feedback from users/employees
- FAQ systems in Organizations and Public Facing Sites
- Help Desk Support Systems etc

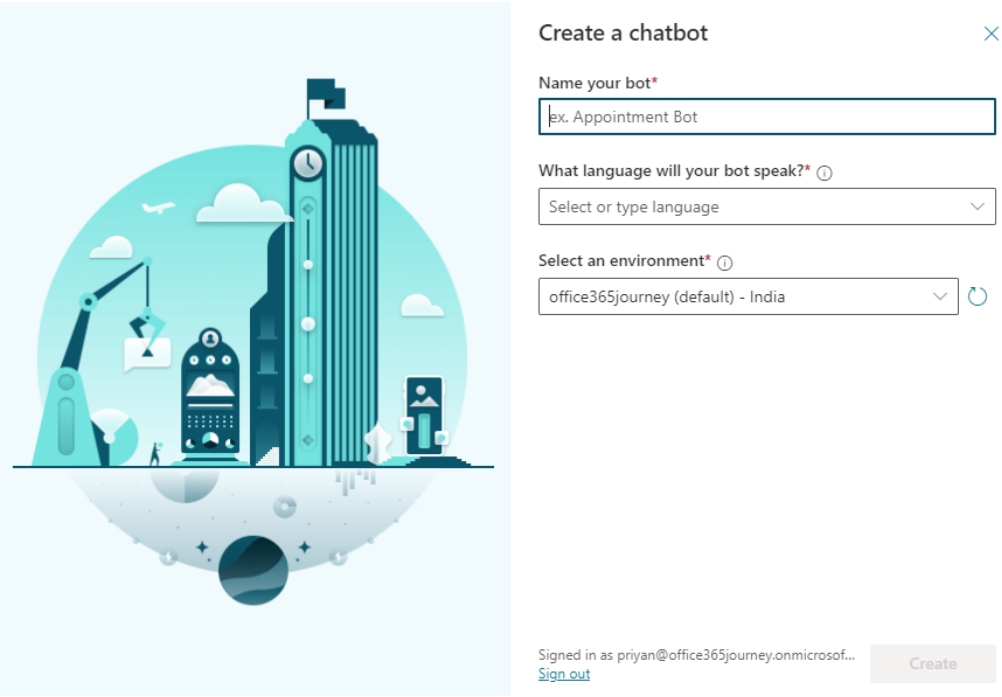
We can create Power Virtual Agents as a Web App as well as directly within Microsoft Teams

Getting Started with Power Virtual Agents

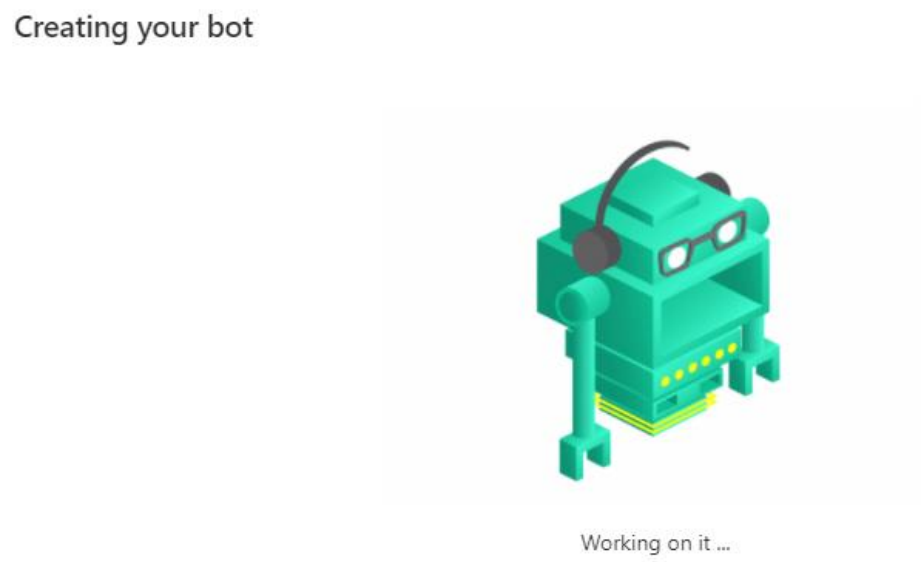
We can easily get started with a Bot creation by selecting New bot from the top right corner of https://web.powerva.microsoft.com/ .



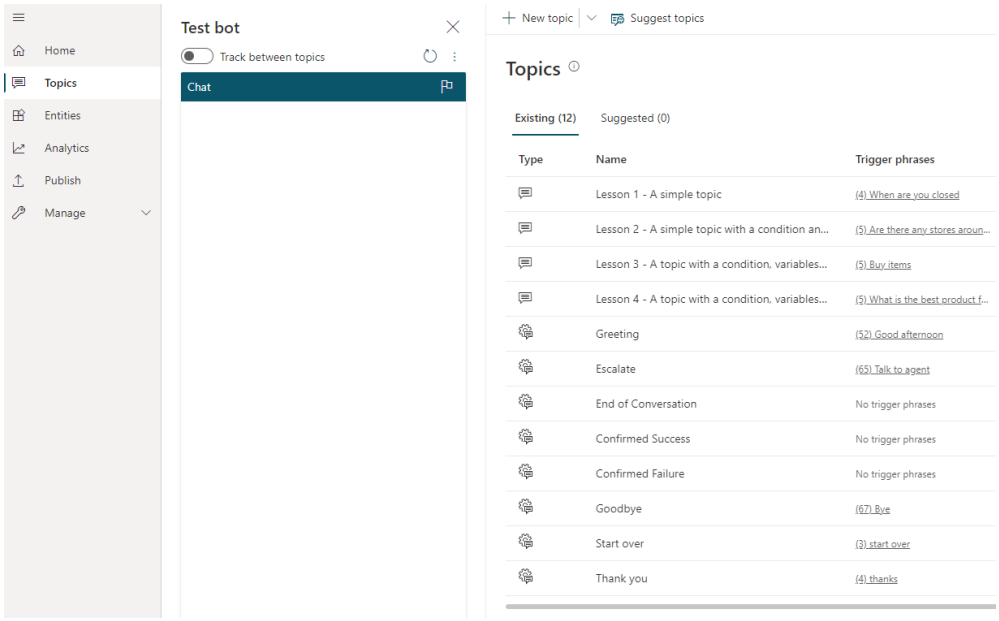
It will give us the option to name the bot and the environment in which it has to be provisioned



After entering the details and clicking on create will provision he bot

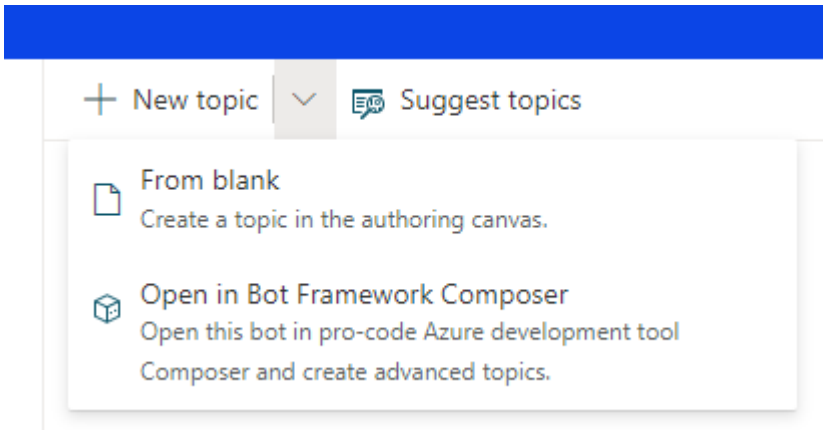


After the bot is provisioned it takes us to the page where we can select the various Navigation options in the left pane. Topics form the core component of a Bot.



Topic

By default we can see predefined topics that takes care of Greeting the user, handle Failure/Success, Escalations etc. We can create New blank topic by selecting New topic



A topic defines how a bot conversation will be initiated and how the bot would respond to user interactions.

A topic is made up of:

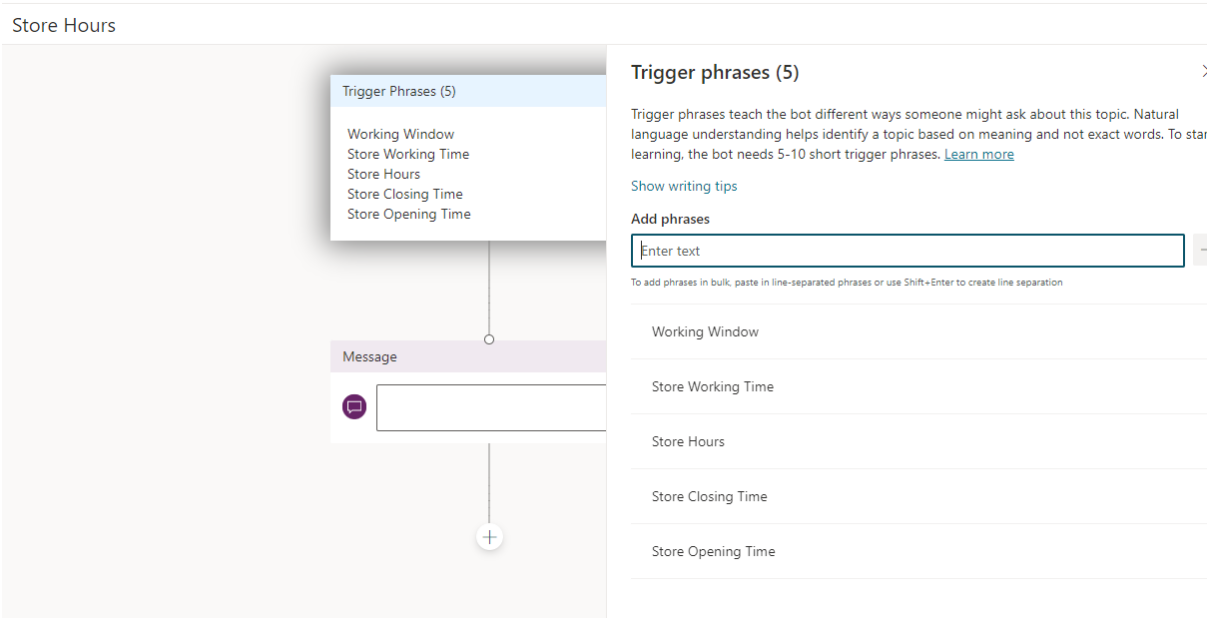
- Trigger: A phrase or keyword which is related to the conversation that the user wants to start. Typing it into the chat window will act as a conversation starter with the bot.
- Conversational Nodes – These are individual actions that will make up a conversation pathway for both. Based on this bot will respond to user interactions

As the first step, we will create a topic that will define a general conversation starter which will pick information about the user. We can have multiple topics created in a bot and based on the way conversation is proceeding, we can navigate between multiple topics as well.

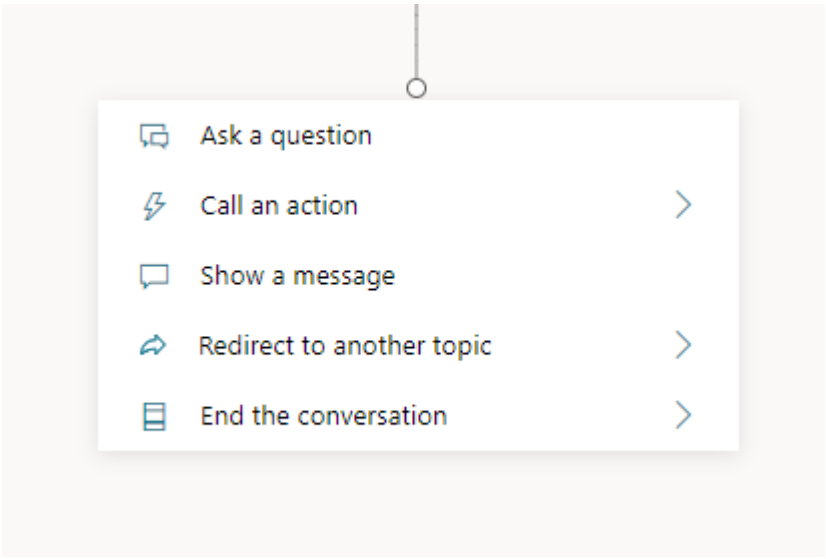
Trigger

For instance, we have create a sample topic called Store Hours that will help users with the store timings.

To trigger the topic, so that the control flows to that specific topic, we will define few words that will transfer the control to these topics. First topic that we will define is Hospital working hours. The trigger phrases can be single keywords or a group of words and to cover a broad spectrum of possible trigger conditions, it is good to mention 5-10 different and still related phrases. We can specify the name, friendly name, and description for the topic in this section.

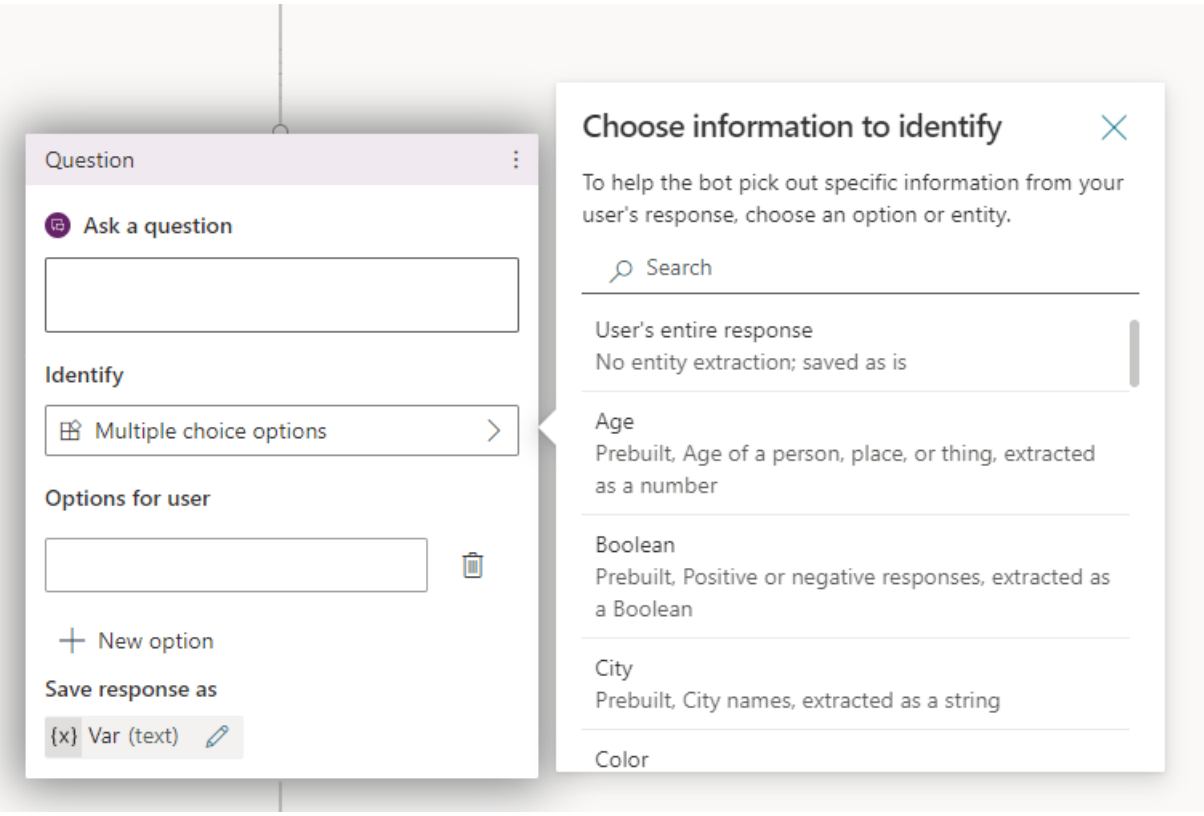


Once the topic and triggers are created, we can add conversational Nodes in the authoring canvas that will define how the communication is driven.

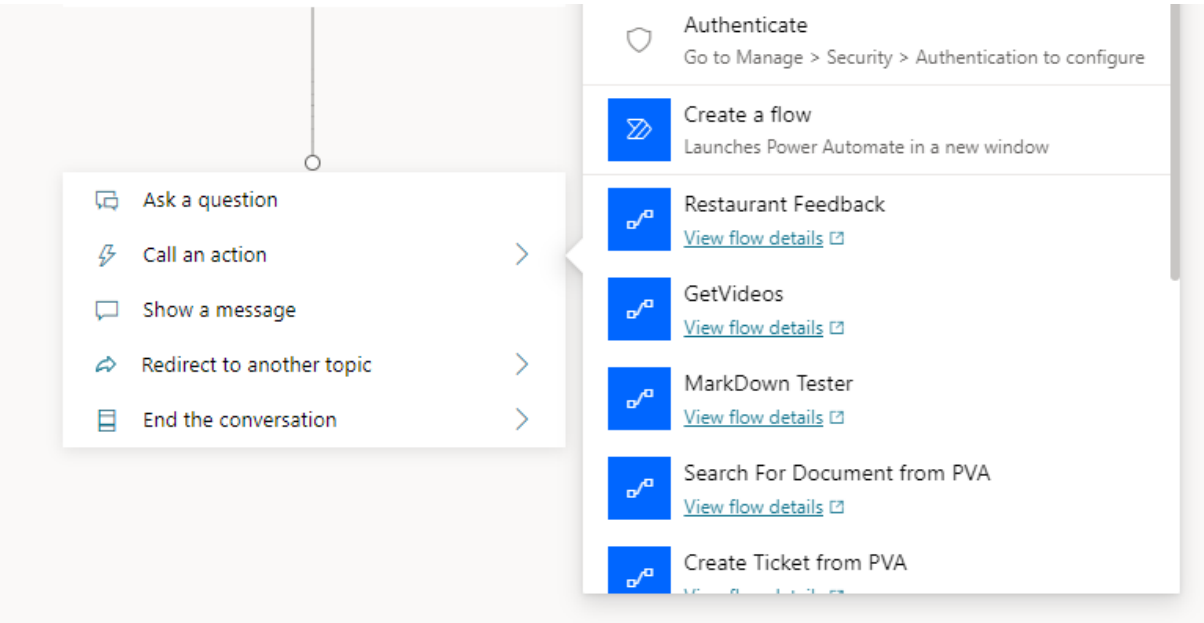


We have the below options that can be added to the canvas:

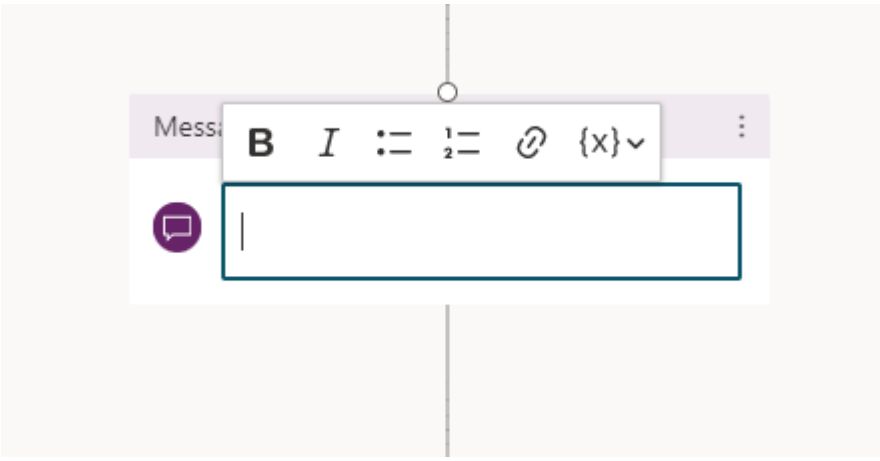
- Ask a question : Lets us ask a question to the user for which he can provide as response which will be stored in a variable which can be later used in the chat. We can accept data across different data types from the user.



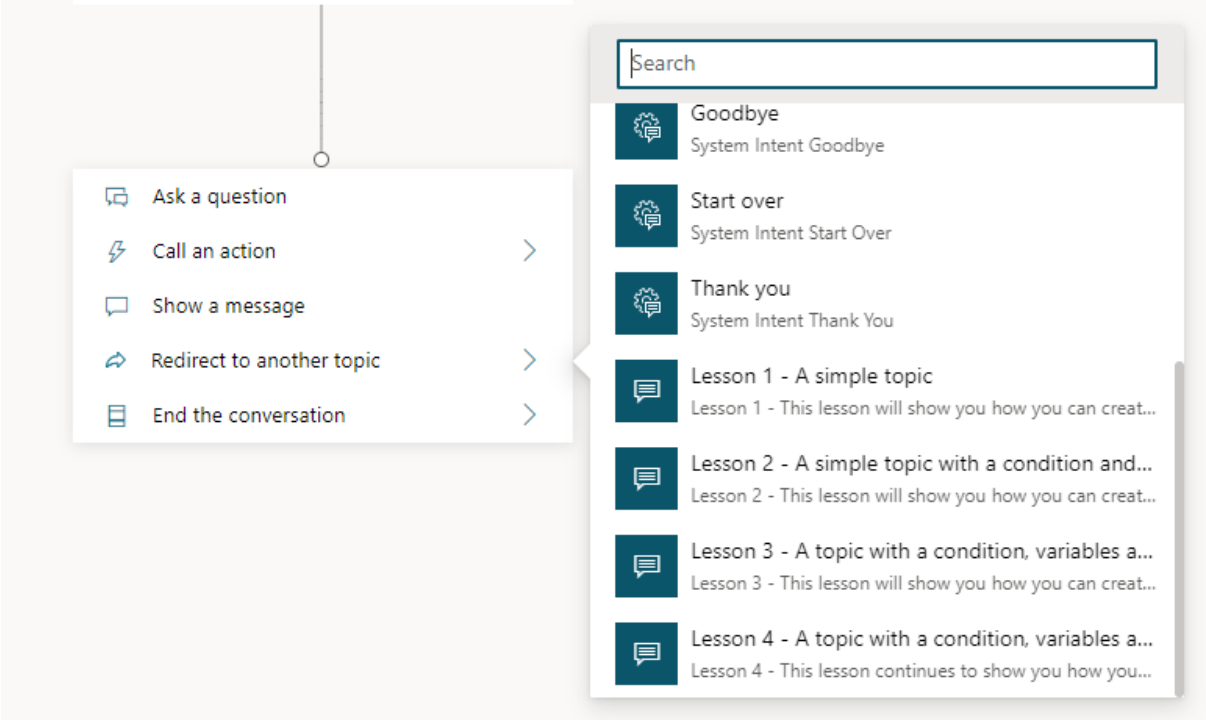
- Call an action: It lets us call a Power Automate to process additional logic using flow. We can pass parameters as well accept output from the flow.



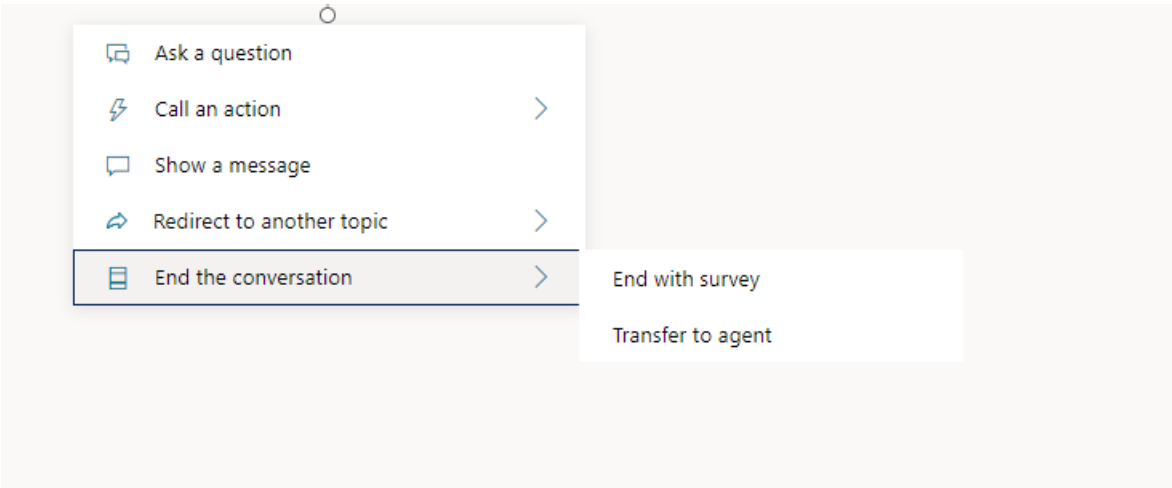
- Show a message : It lets us show a simple message to the end user. It provides basic formatting options as well as to add links and use variables from previous actions.



- Redirect to another topic : It lets you navigate to another topic to handle the questions from the user or to complete the conversation flow.

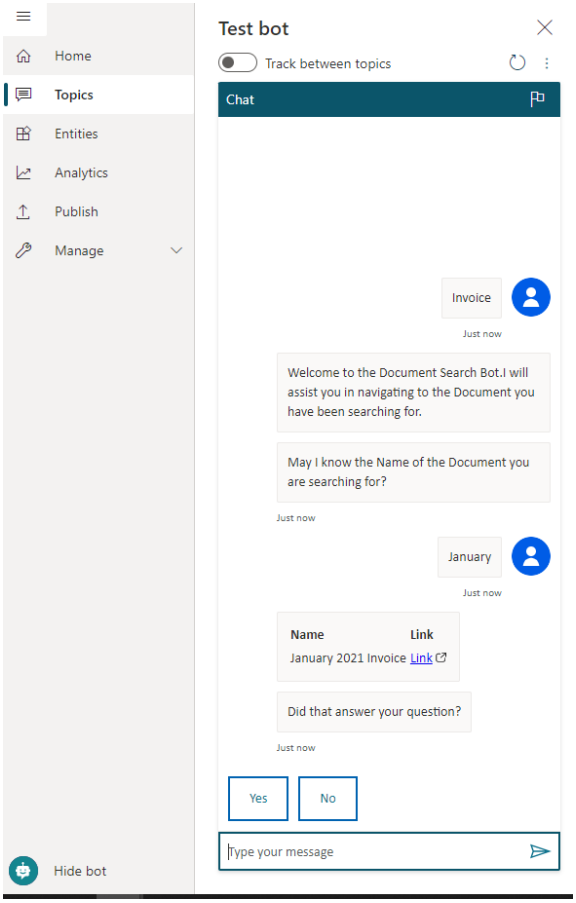


End the conversation: It lets you close the conversation with a survey or lets you hand over the conversation to a human agent waiting in the omnichannel if the topics are not good enough to handle the user queries.



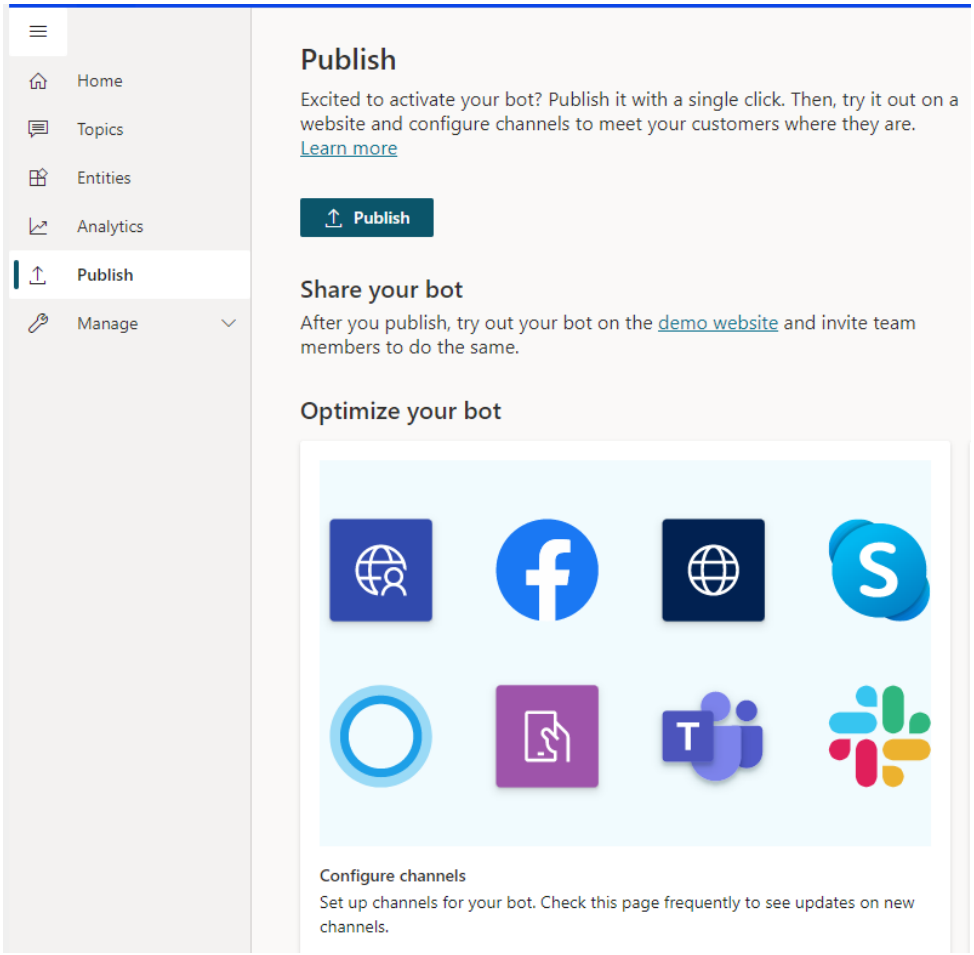
Test the bot

Once the bot is created, we can test it in real time using the testing facility. Once the Topic is saved, triggering it will start the bot conversation and we can see the full flow in action before publishing it to any live channels

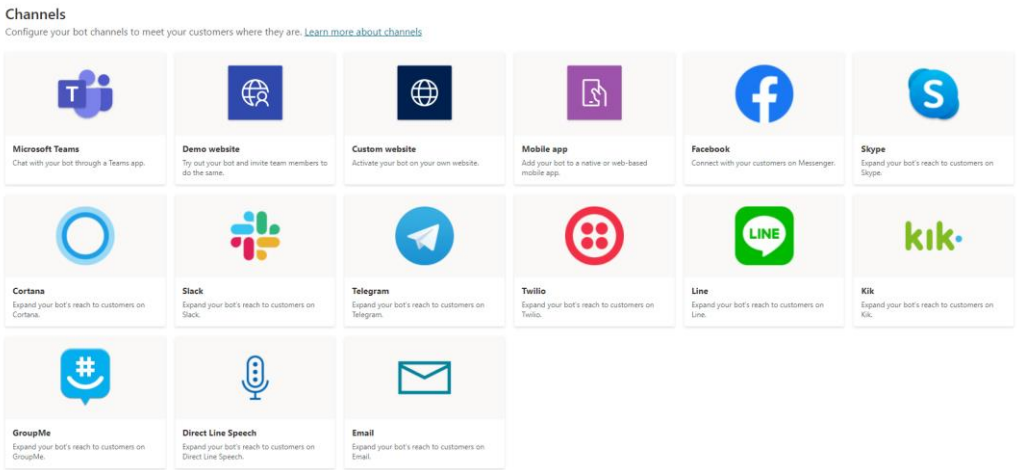


Publish the bot

Once the testing is completed, we can publish it and add it any of the available channels so that users can start using it in real time.

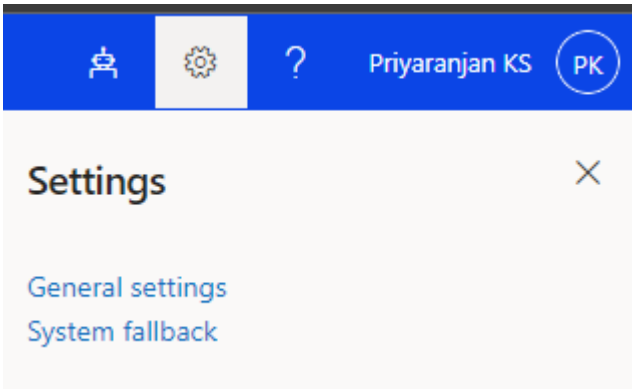


As of now, we have the below channels to which the bot can be added for use.

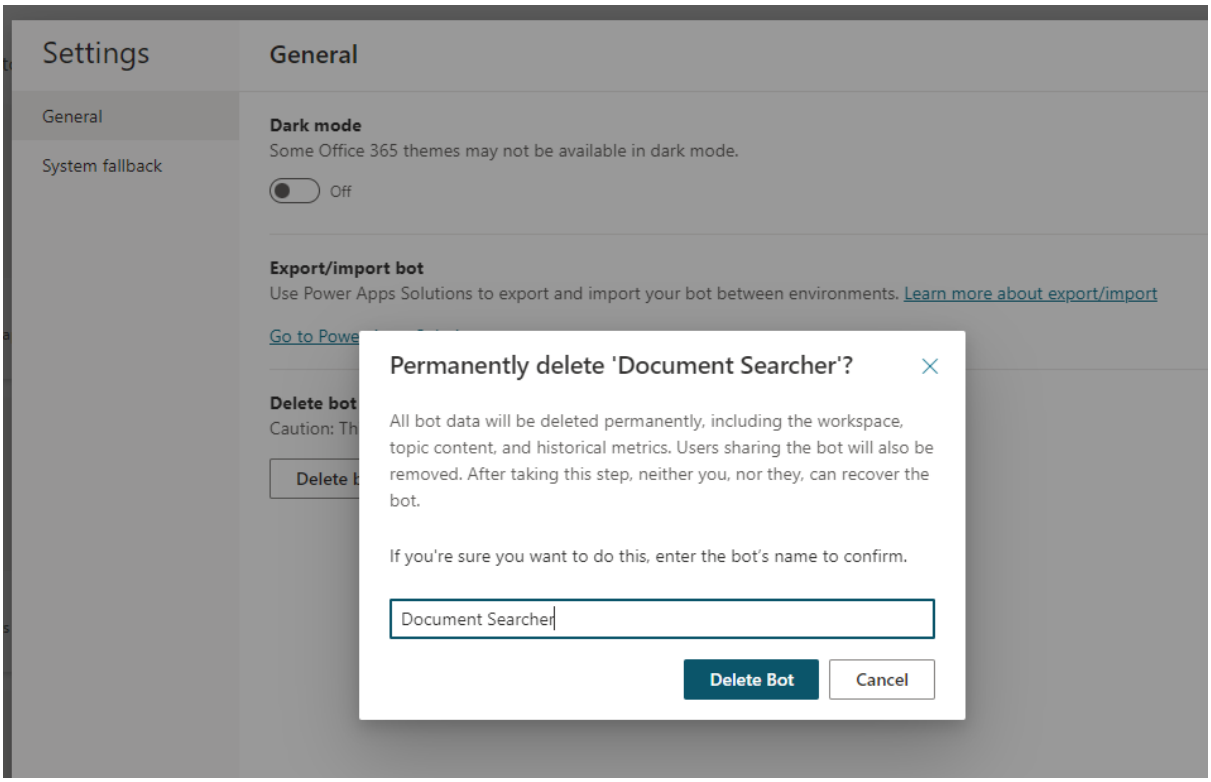


Delete the bot

In case we want to delete the bot, we can head over to the settings in the top right corner. Select General Settings.



Click on Delete bot and it will open a window where you can type in the bot name and if the name is correct, the Delete button will be enabled.



Summary

Thus, we saw a very high level, how we can get started with Power Virtual agents and the basics of creating a bot.

Create a Multi Topic Bot and Publish to Microsoft Teams Channel

Introduction

With Power Virtual Agents we can create Topics to cater to multiple pathways in which a conversation will travel. This helps users to get details about various nonrelated questions that they have resulting in a much more complete user experience when they need to get details that they are looking out for. We will also see how we can publish the bot into a Microsoft Teams Channel.

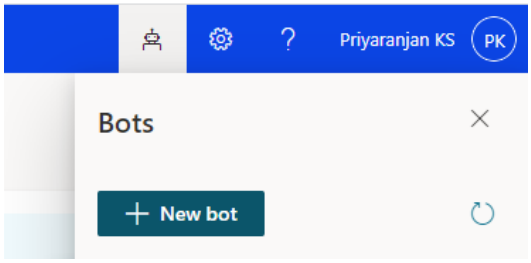
In this article we will see how to create a basic bot that handles multiple topics and publish it in a channel.

Business Use Case

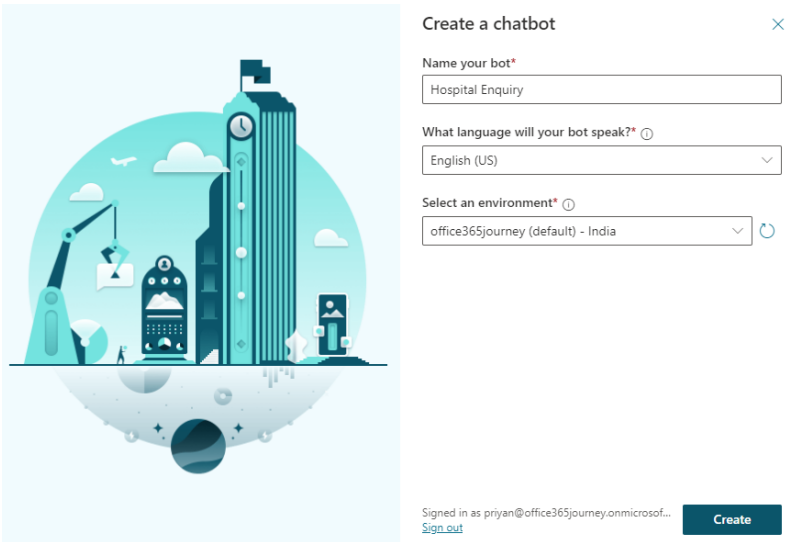
We have a requirement where we need to create an auto response system for an enquiry that come to the Hospital. Having a dedicated user to attend to basic queries can lead to financial overhead and the enquiry service will go down beyond the normal working hours of the employee. To avoid this, we will try to setup an automated response system for some of the commonly asked questions.

Implementation

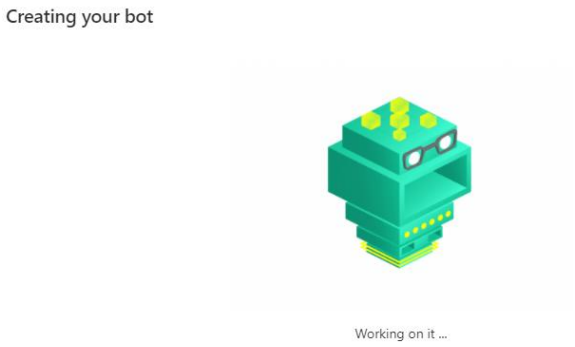
To implement the Automated Enquiry system, we will use Power Virtual Agents where we can create a bot to attend to the queries and give automated responses based on the information we have in the system. To start with we will head over to https://web.powerva.microsoft.com/ and click on the Bot symbol on the top right corner which lets you create a new bot



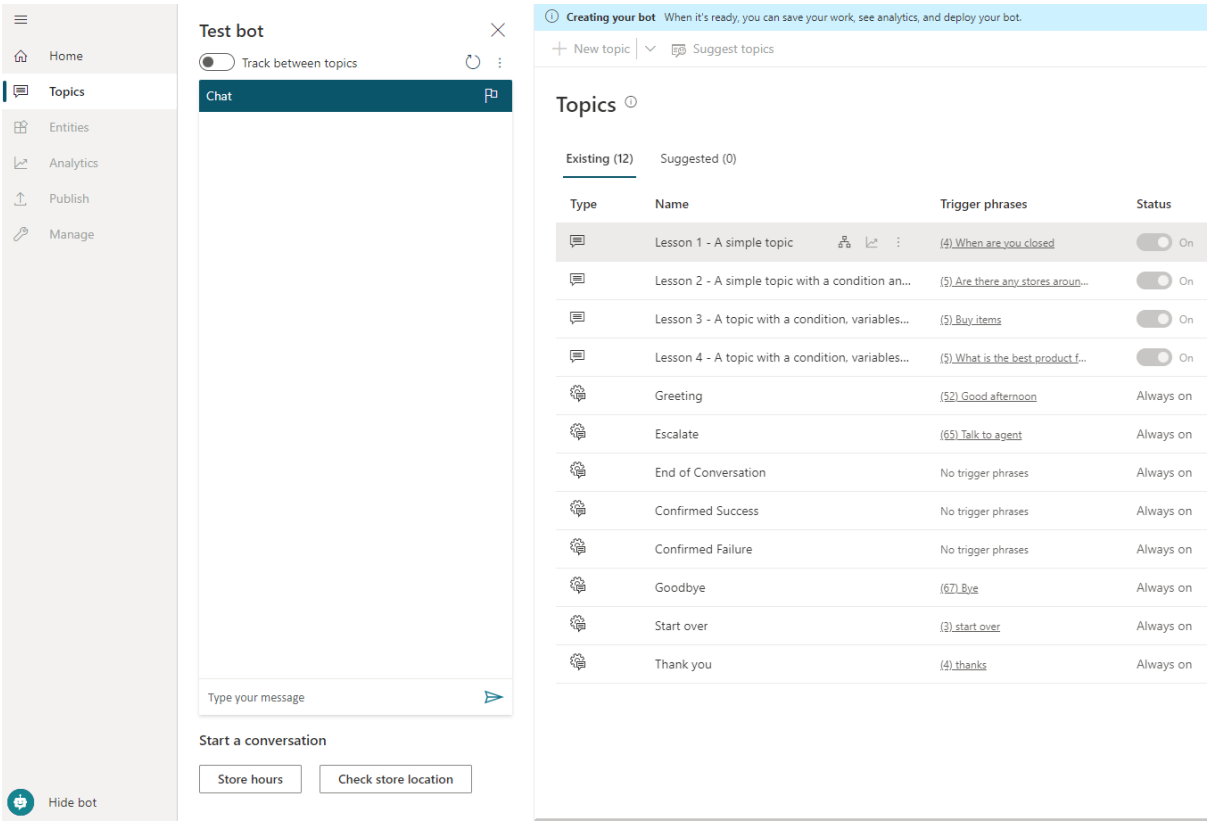
Specify the name and language for use in the bot as well as the environment where it should be provisioned.



It will create a basic bot where we can add the customizations needed for our requirement.



We can see the section Topics which by default lists multiple conversation topic listed out. A topic defines how a bot conversation will be initiated and how the bot would respond to user interactions.

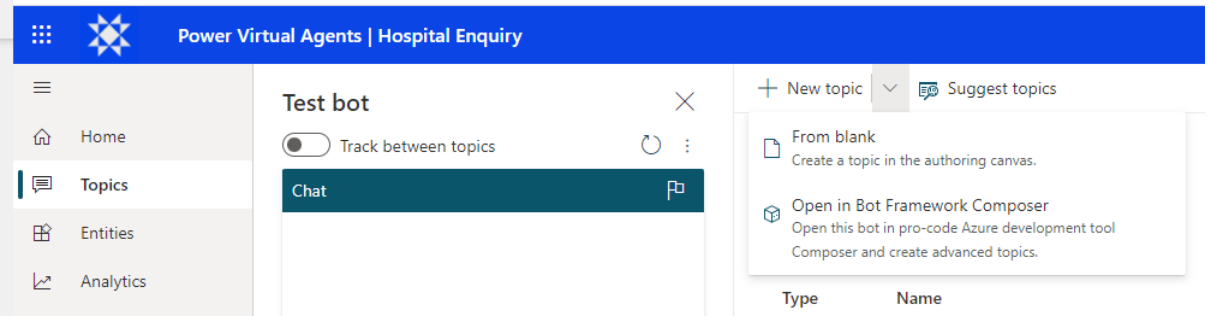


A topic is made up of:

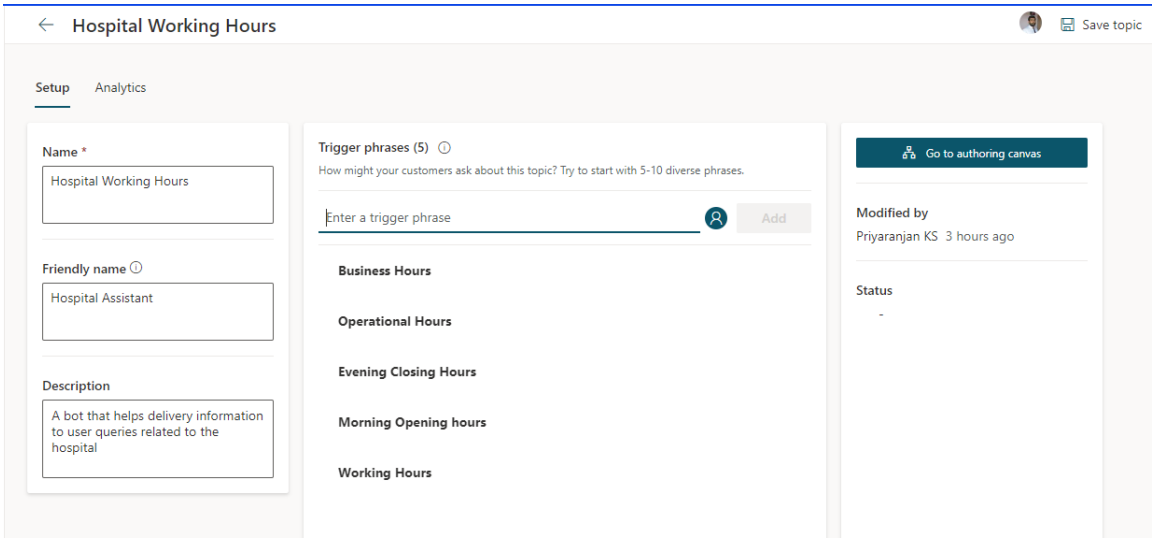
- Trigger: A phrase or keyword which is related to the conversation that the user wants to start. Typing it into the chat window will act as a conversation starter with the bot.
- Conversational Nodes – These are individual actions that will make up a conversation pathway for both. Based on this bot will respond to user interactions

As the first step, lets create a topic that will define a general conversation starter which will pick information about the user. We can have multiple topics created in a bot and based on the way

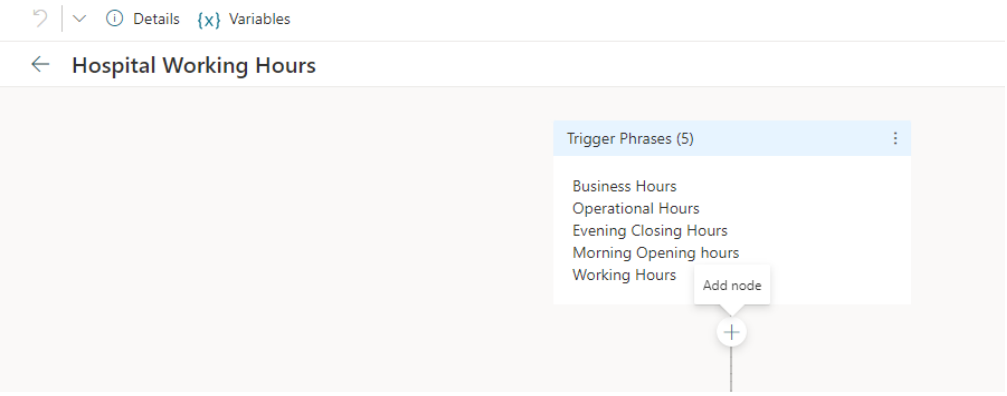
conversation is proceeding, we can navigate between multiple topics. Click on New Topic and Select From blank to create a Topic



We will create 4 topics each of them corresponding to an enquiry .To trigger the topic, so that the control flows to that specific topic, we will define few words that will transfer the control to these topics. First topic that we will define is Hospital working hours. The trigger phrases can be single keywords or a group of words and to cover a broad spectrum of possible trigger conditions, it is good to mention 5-10 different and still related phrases. We can specify the name, friendly name, and description for the topic in this section.



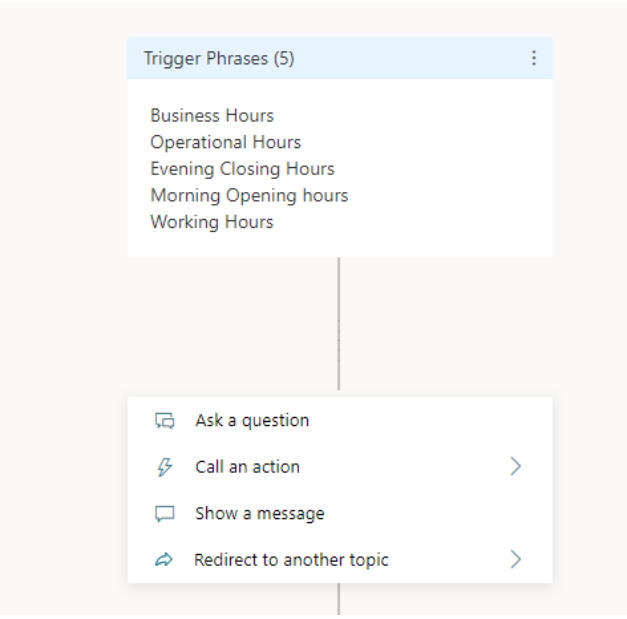
Once we are done with the updation of the trigger conditions, click on the “Go to authoring canvas” which will provide us with a screen where we can define the conversational nodes that will control the flow of the conversation and how the bot responds to the user.



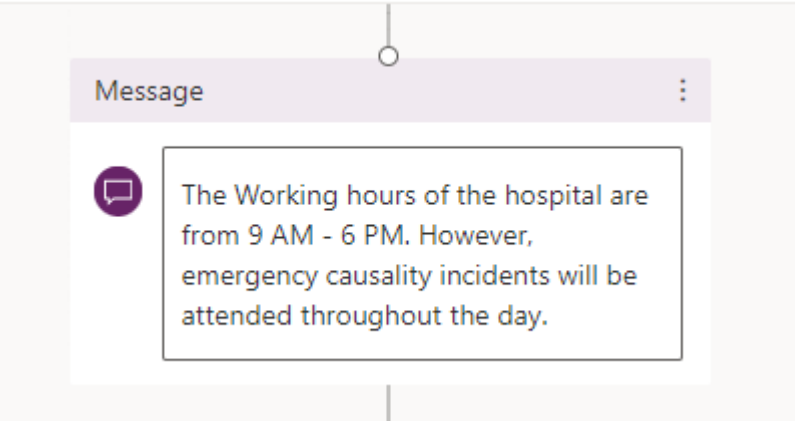
Clicking on Add node will provide us with the below options:

- Ask a question - We can Ask questions to the user so that we can take input from the end user
- Call action – Call Power Automate to start a flow
- Show a message – shows static message or information to the end users
- Redirect to another topic – Transfers control to another topic for execution

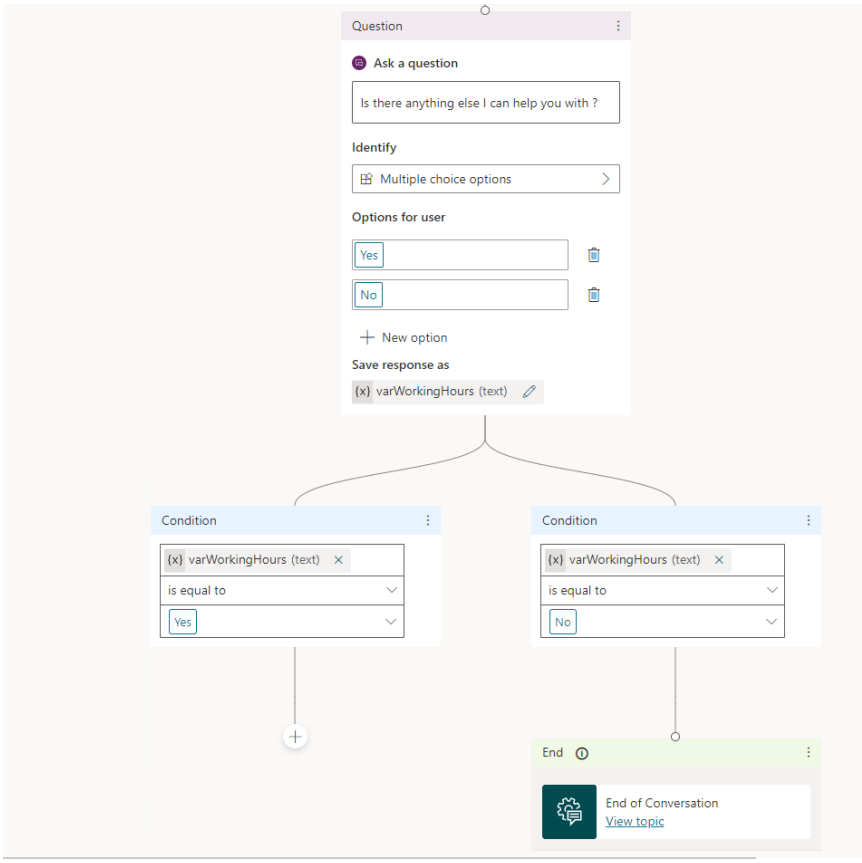
In the current sample, we will select show a message node.



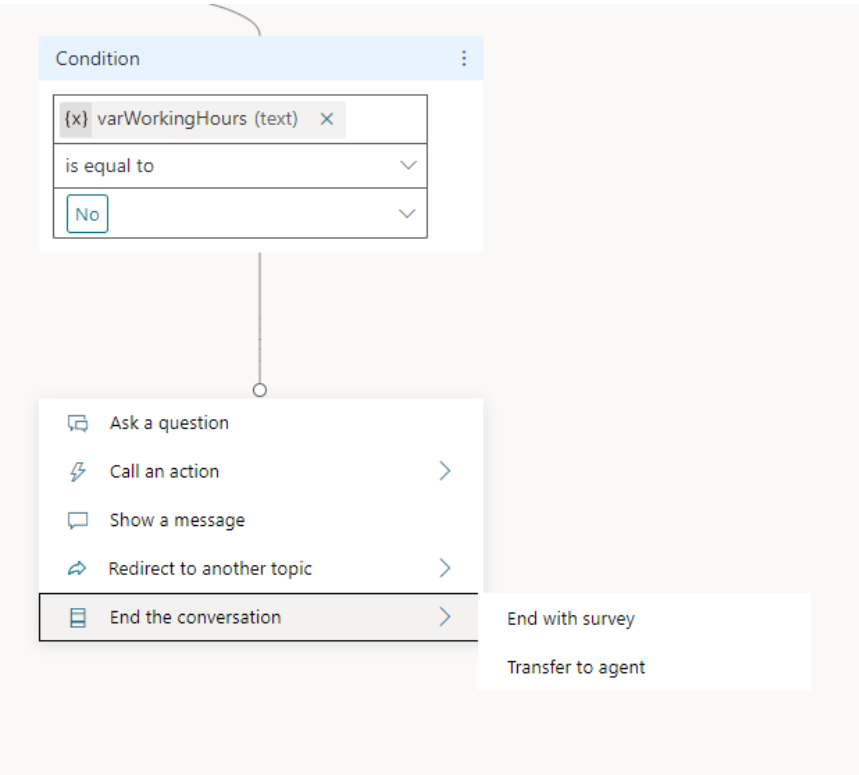
We will add the information about the hospital timings in the message box which will be displayed back to the user.



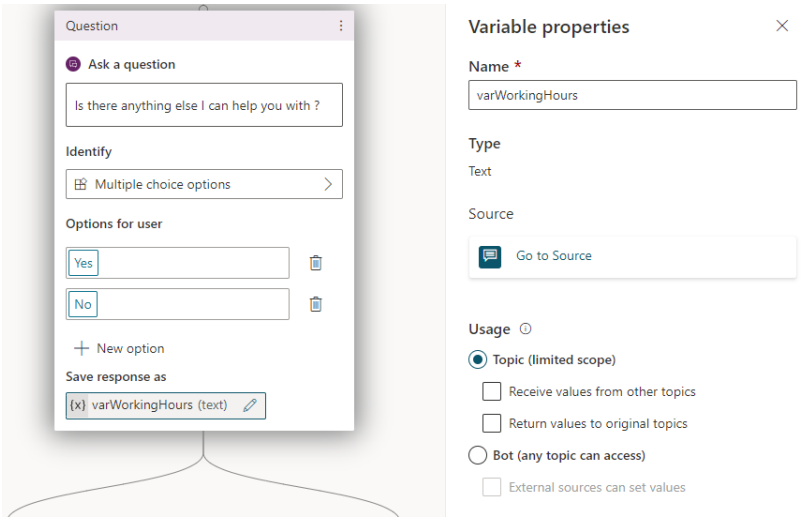
We will need to keep the conversation going with the user and to ensure that, we can ask a question, if they need any more help, by adding another node and provide a Yes/No option.



The response selected by the user will be saved in the variable name varWorkingHours. Based on the value, it will have branches and the control will flow based on the user input. In case the value is Yes, the chat will listen to next input from the user which will trigger another topic else if the value is no, we will end the conversation by selecting a survey so that it exits the chat session.

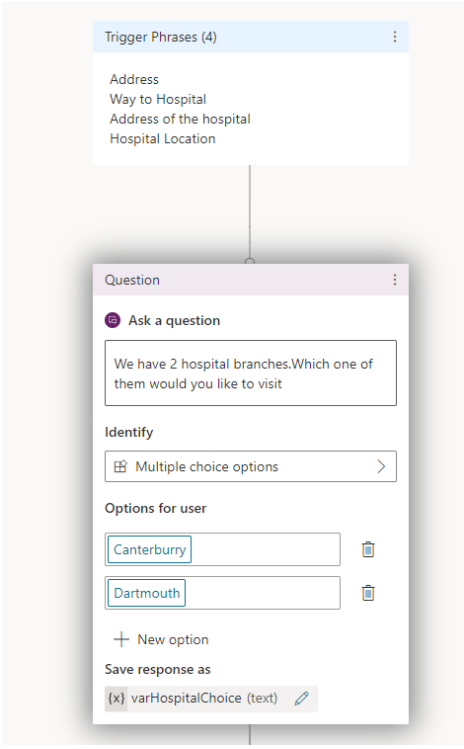


In case we need to rename the variable, we can click on it which will open the right pane where we can edit the variable name.

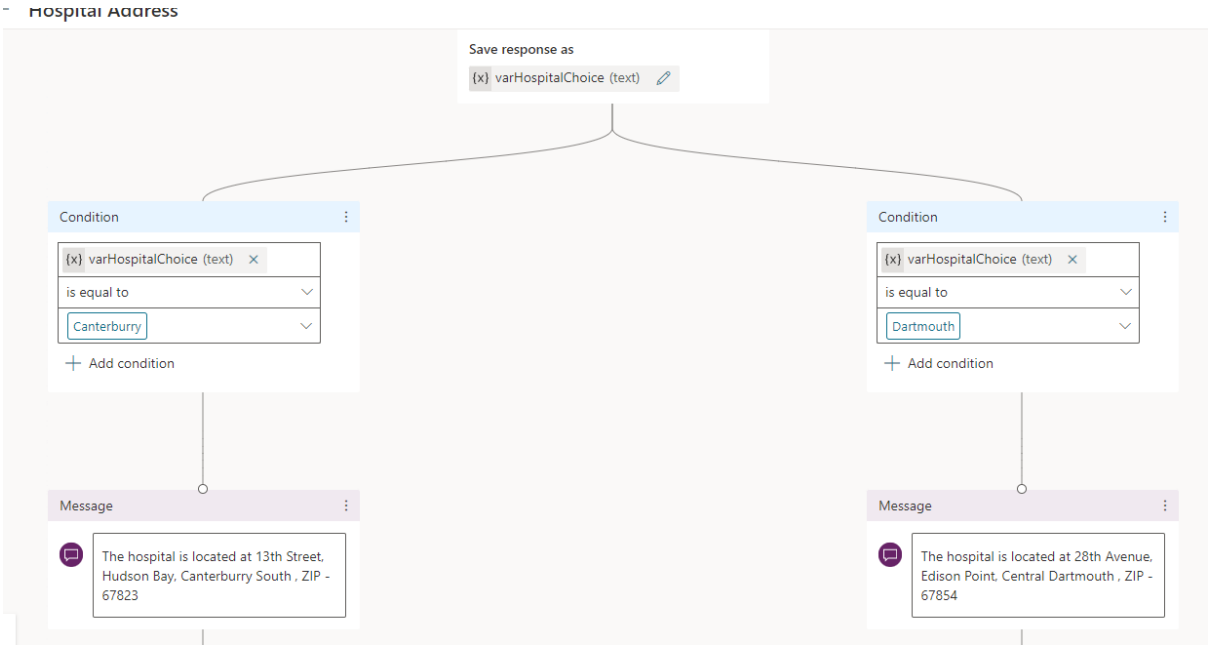


Add more topics

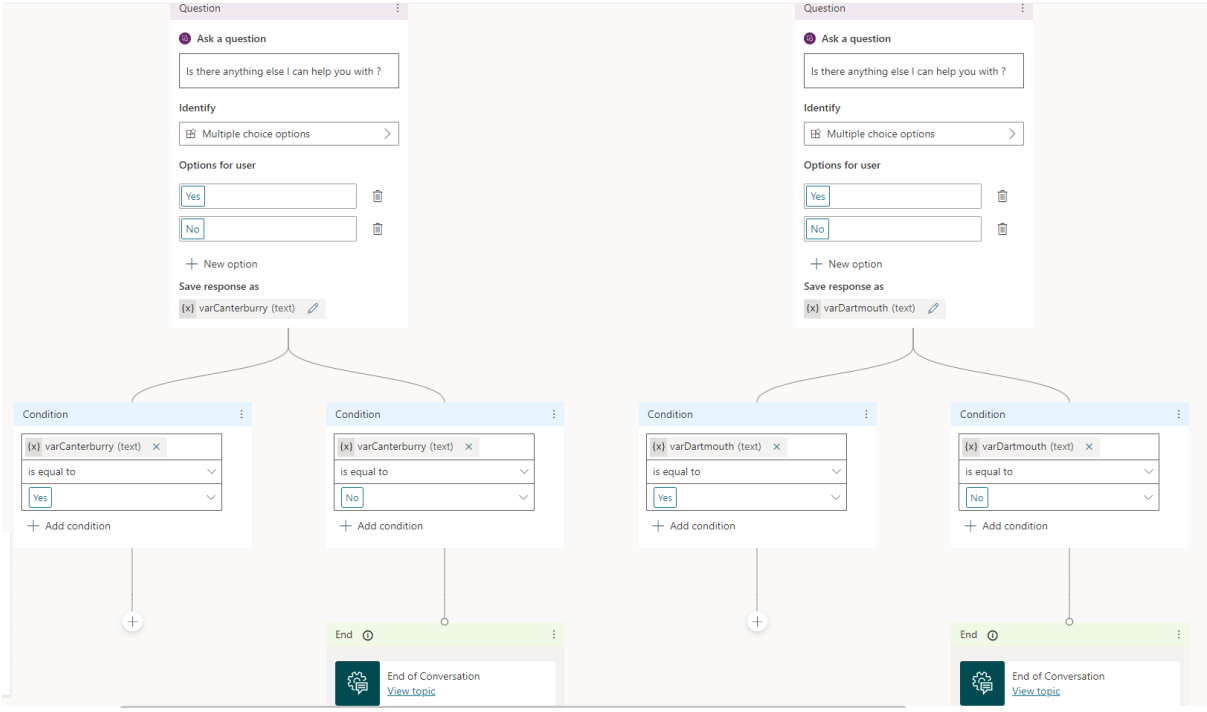
Same way, we can add more topics to broaden the Knowledge base so that the bot can take up more questions and provide valuable information to the user. We will go back to Home and click on New topic and this time create a topic – Hospital Address. We will add the trigger phrases and move on to the authoring canvas. As there are 2 branches to the hospital, We will add a question to get more clarity on what information is the user seeking by providing the location for both branches.



Based on the user input, the value will be saved in the variable varHospitalChoice and the control will pass on to the corresponding branch which will show the respective hospital address in the message box.

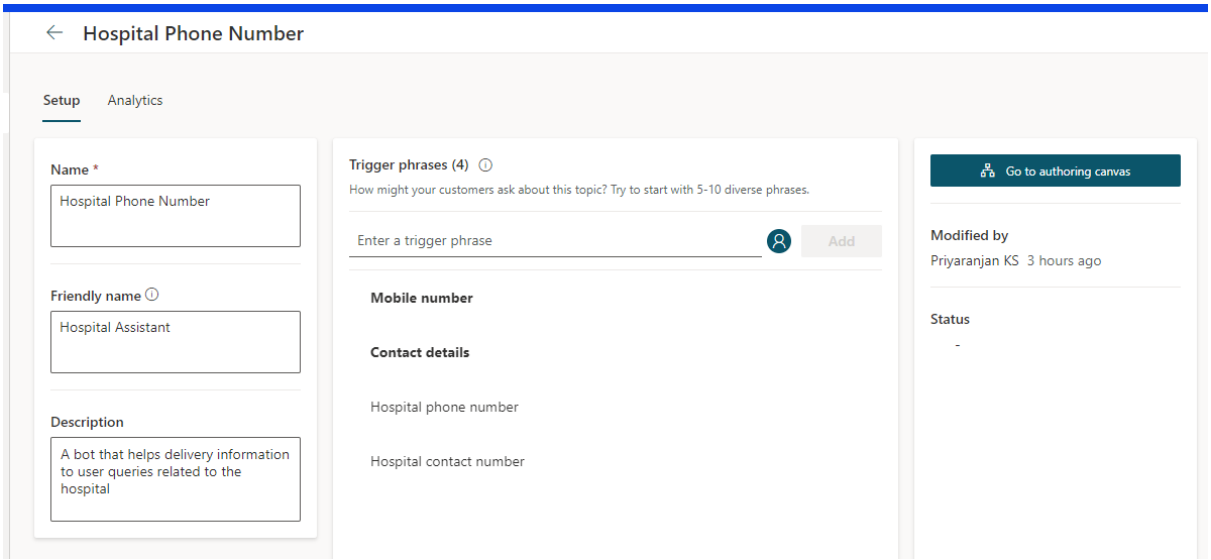


To continue the conversation, we will ask the same continuity question we asked in previous topic. If the user chose Yes, we will again keep the node open ended which will listen to the user input from the chat window to trigger another topic and if the user selects No, we will end the chat with a survey.

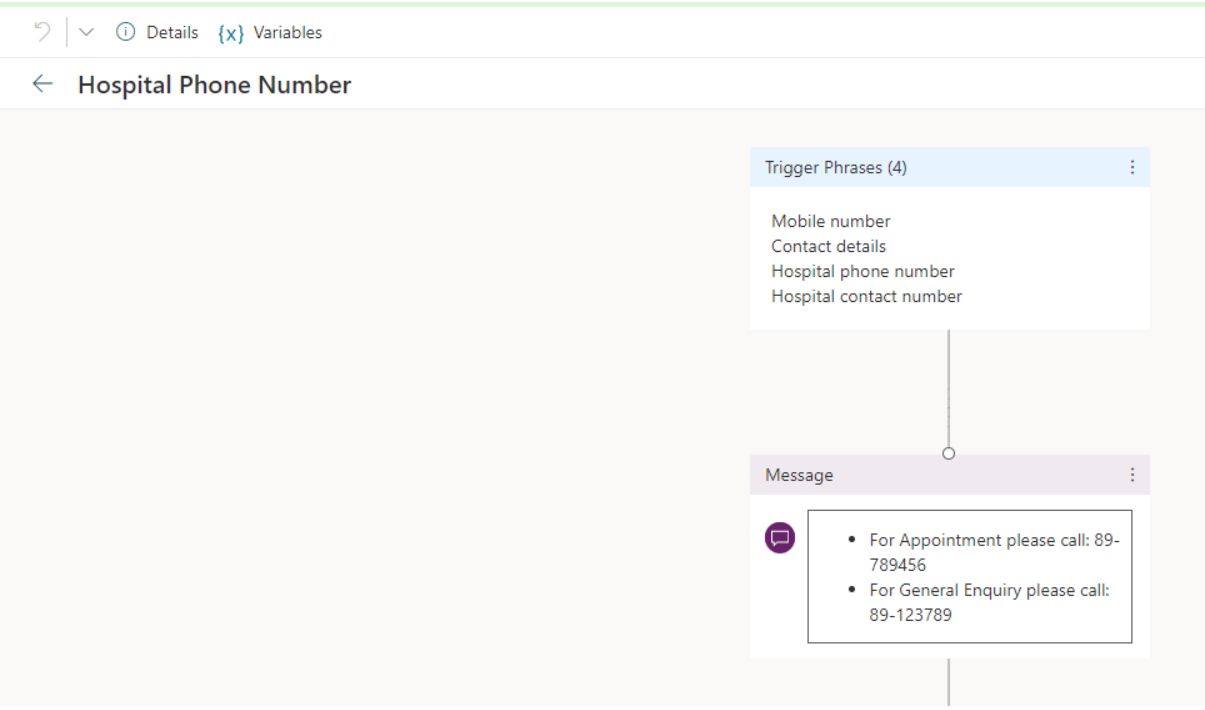


More Topics

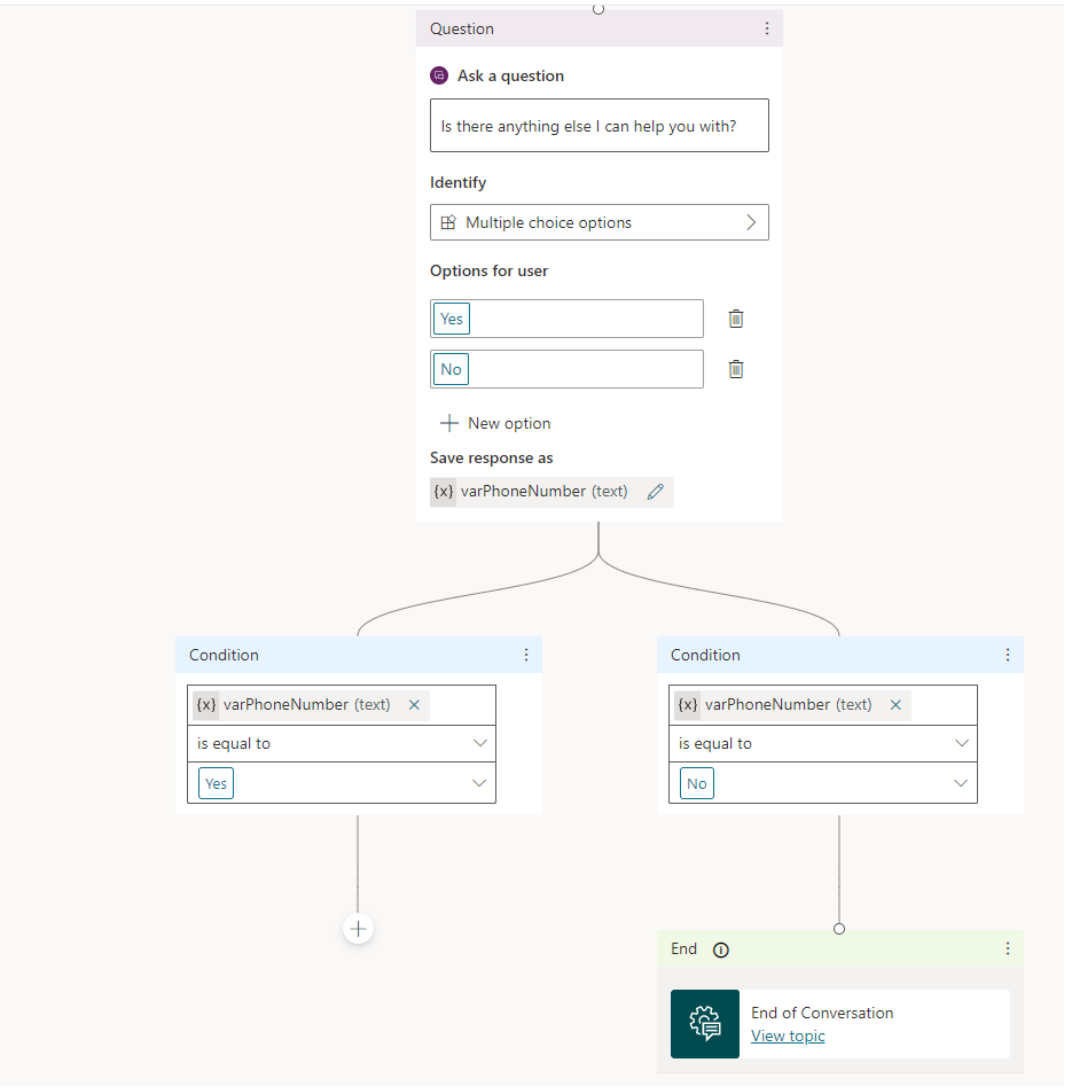
As a third topic, lets add the topic that will drive conversation around the Contact Number and name its Hospital Phone Number. We have added few trigger phrases and will move on to the authoring canvas.



Inside the authoring canvas, we will add the message box that gives the details about the contact numbers

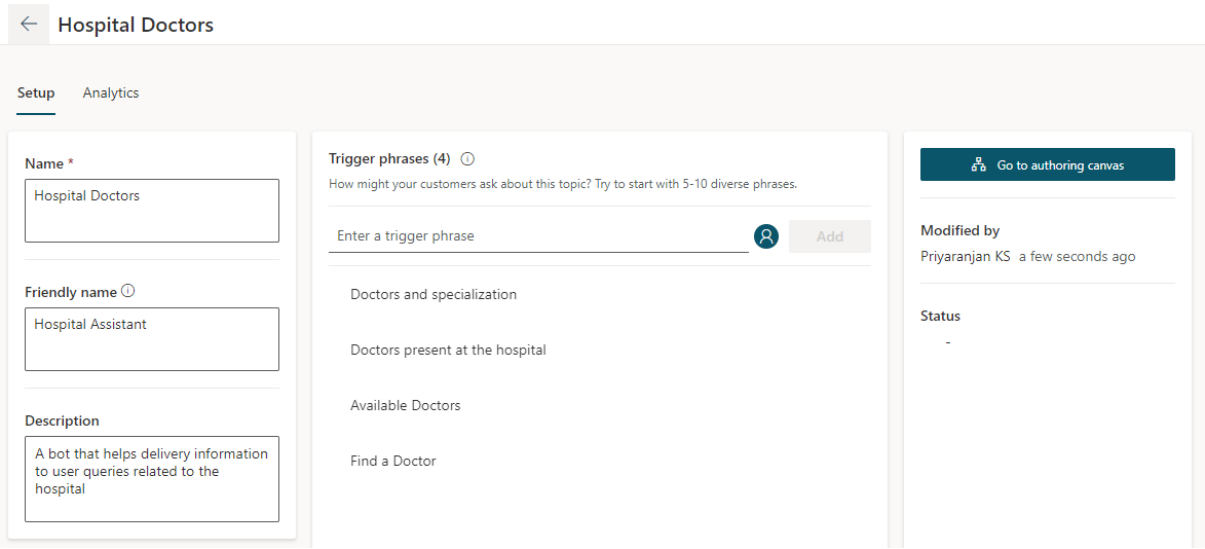


Finally, we will add the continuity question and based on the Yes/No input of the user we will either leave the node open ended in case of a Yes or end the chat with a survey in case of a No.

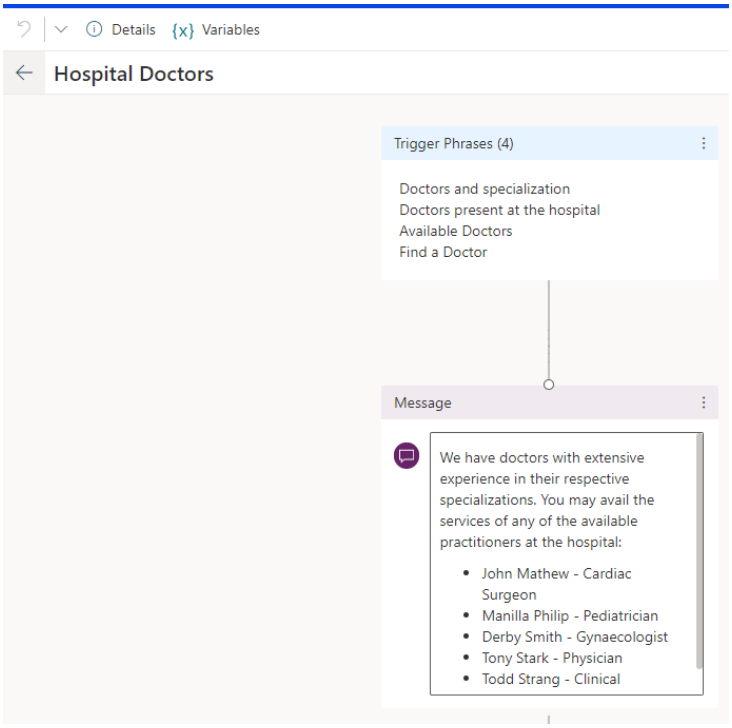


Final Topic

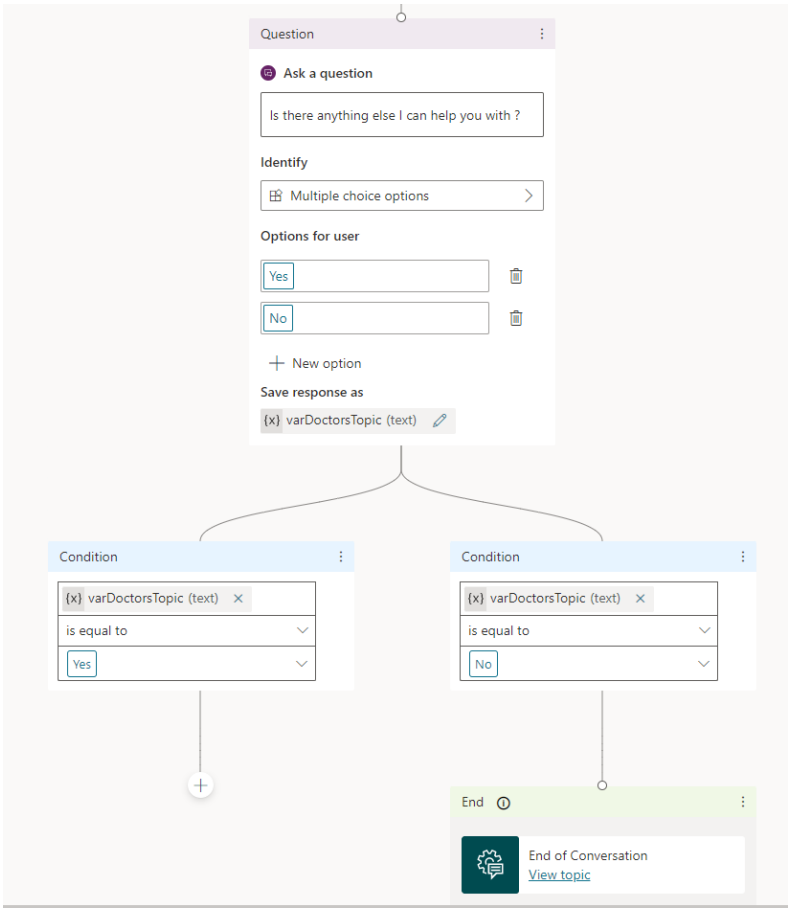
Let's add one more final topic that gets us the doctor details and name it Hospital Doctors . After adding few trigger phrases, we will move on to the authoring canvas.



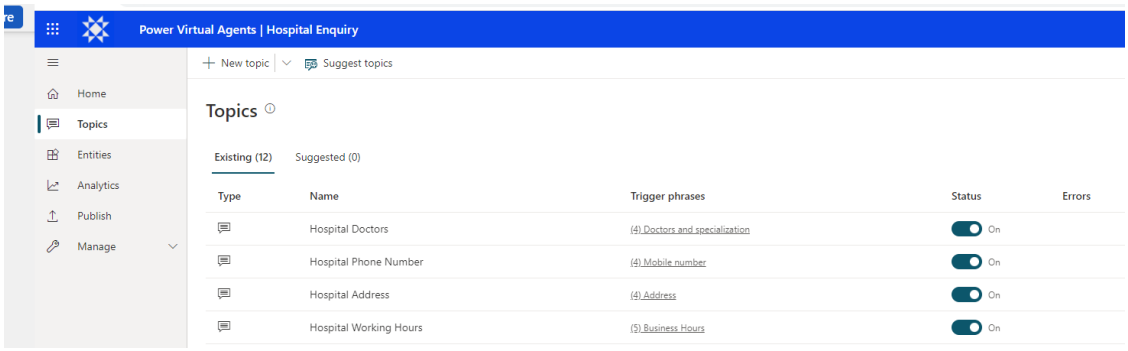
Here we have added the message node to display the doctor information.



We will finally add another node to ask the usual continuity question and if the input is a Yes, leave the chat open ended for further listening to trigger phrases that invoke any other topics or if the input is a No, end the chat with a survey

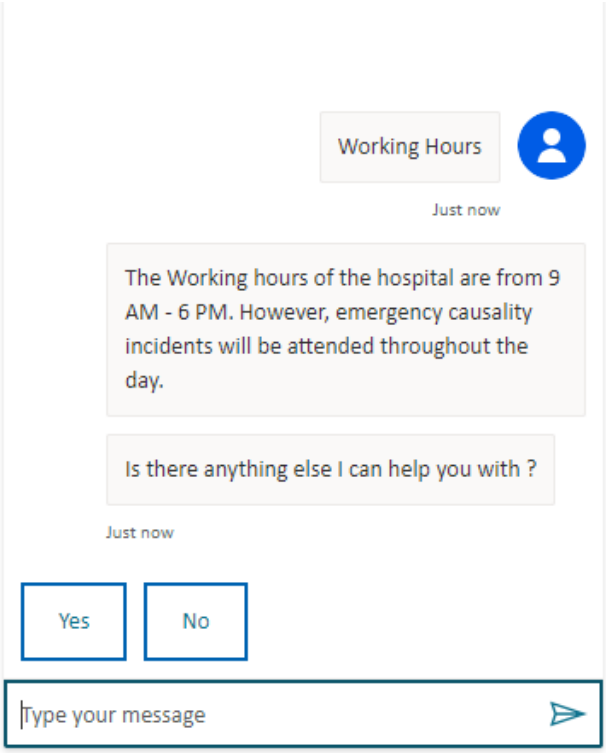


Thus, we have created 4 topics overall

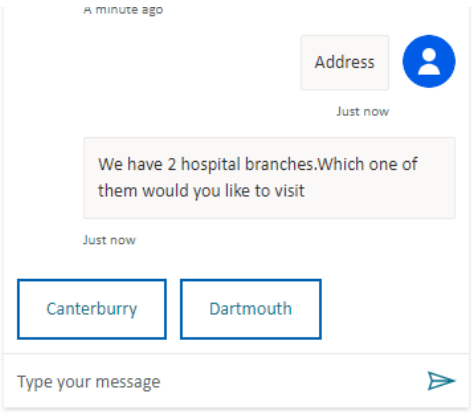


Test the Bot

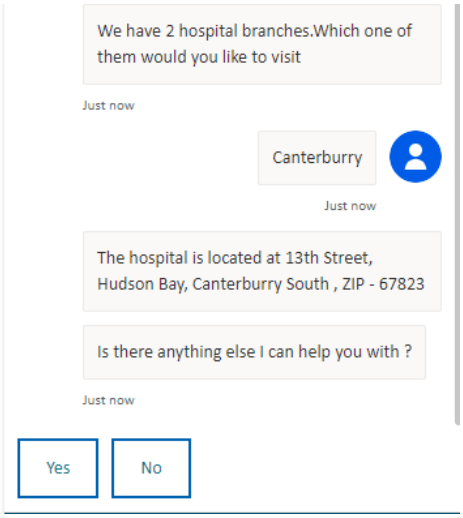
Now let's test the bot by making use of inbuild test bot test pane where we can type in a message which will trigger one of the topics. I have typed in the Working hours text which invoked the "Hospital Working Hours" topic and gave us the corresponding information as a message.



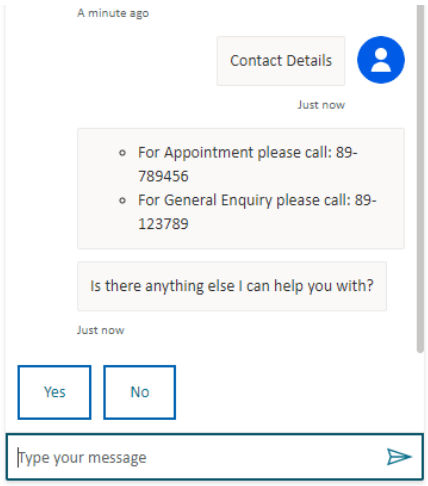
Let's add another query related to Hospital address



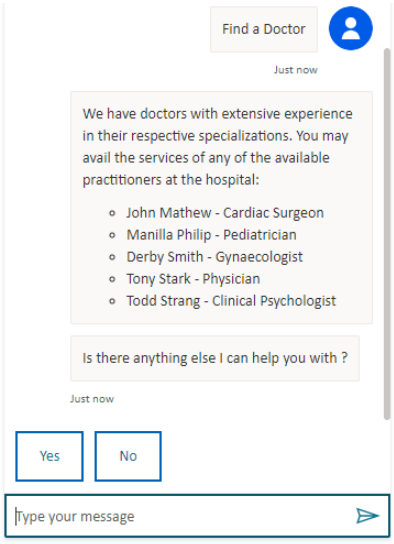
Selecting one of the location, will give us the exact address of the hospital :



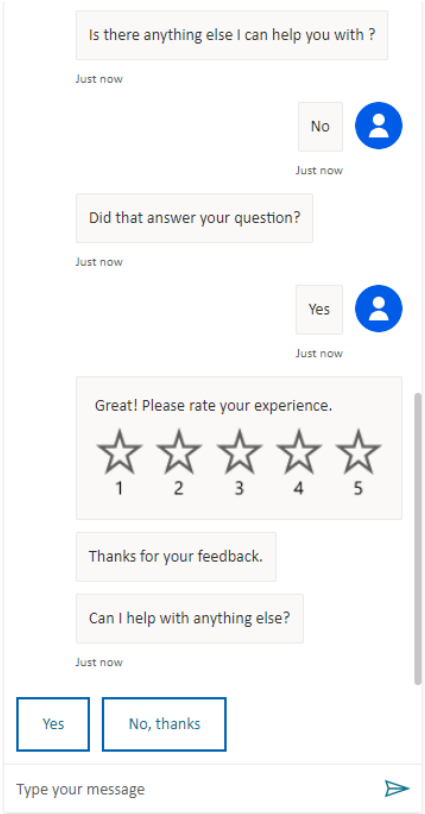
Trying out another query related to Contact Number, we get to trigger the Hospital Phone Number Topic and give us the required details



Let try one final query related to Doctor information which triggers the topic Hospital Doctors and gives us the details of the available doctors

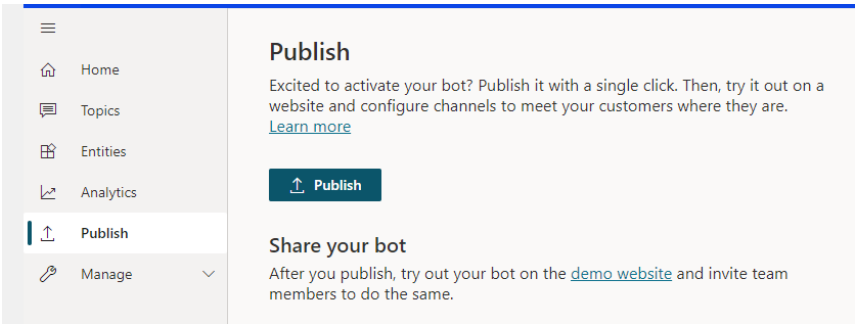


Selecting No, will end the chat with a survey

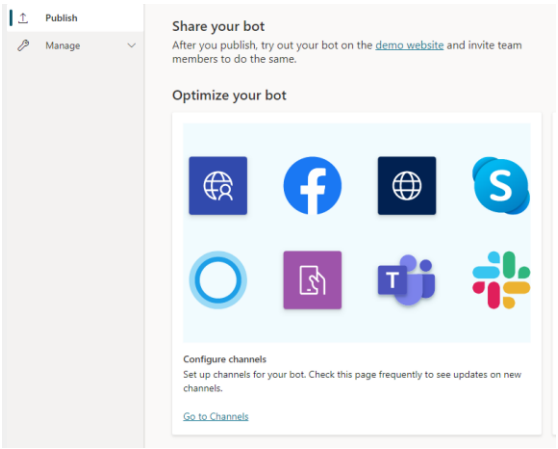


Publish the Bot

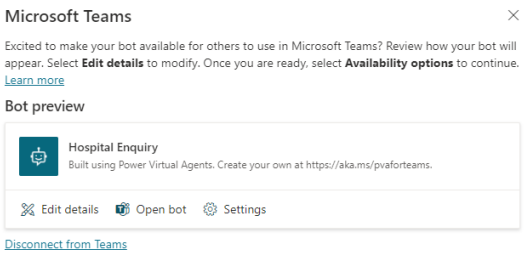
Let's finally publish the bot by going to the Publish Tab



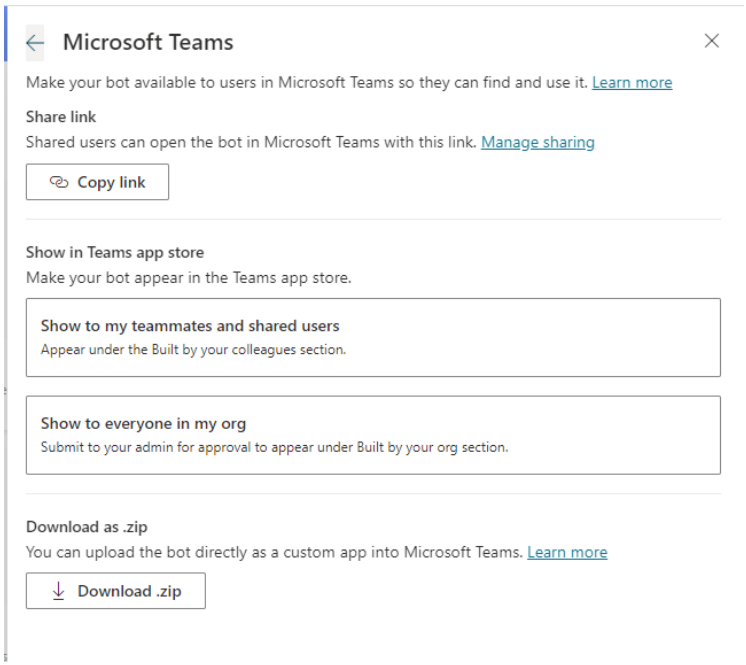
We can make the bot available in multiple channels by clicking on Go to Channels option at the bottom of the page.



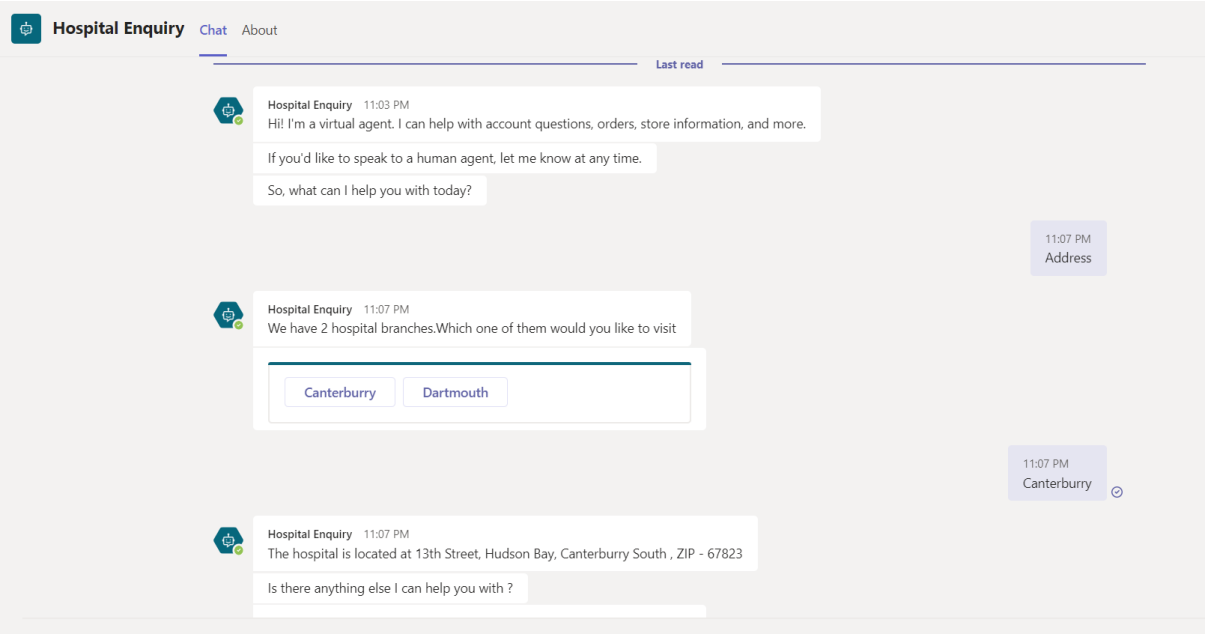
It will provide us the option to view the bot in Teams to try it out even before installing it in team by clicking on Open bot. Or else to view options to deploy it to teams, click on Availability Options



It will provide the option to share it with users or submit it to the admin for tenant wide availability. We can also download the bot as a zip and upload it as a custom app in Teams.



Once the bot is made available in teams, we can test it out the same way we did in the inbuilt bot tester.



Same way , we can deploy it to other channels like host it in a website, Facebook, slack, telegram etc. :

Summary

Thus, we saw how we can use a Power Virtual Agent to create a very basic Automated Enquiry system using Bots and multiple Topics

Call Power Automate from Power Virtual Agents

Introduction

Power Virtual Agents helps us create intelligent conversational chat bots without any code implementation. With these bots we can engage with customers and employees in multiple languages across websites, mobile apps, Facebook, Microsoft Teams, or any channel supported by the Azure Bot Framework

In this article we will see how to create a basic Appointment Booking System using Power Virtual Agent, Power Automate and SharePoint Lists

Business Use Case

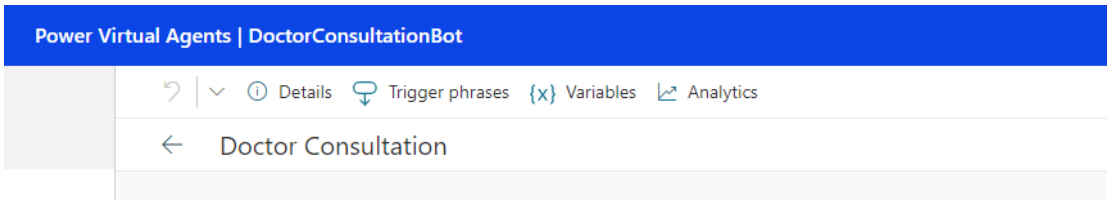
We have a requirement where we need to setup an automated appointment booking system which will let the users book a specific doctor of choice for consultation on a specific day and keep the appointment record in the hospital back end so that the hospital can pull daily appointment reports for operational activities.

To achieve this, we will create a SharePoint List in the back end to store the appointment details with the columns:

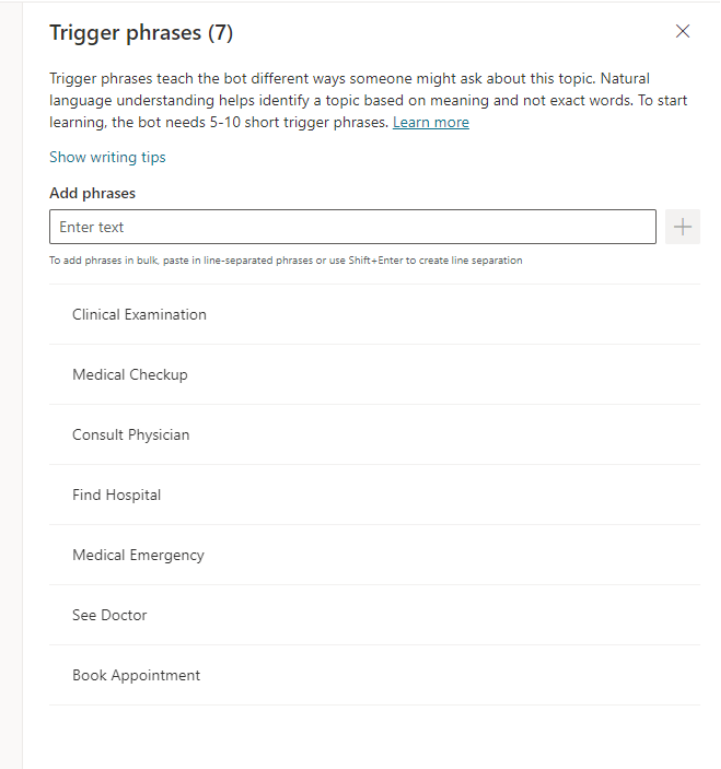
- Patient Name
- Appointment Date
- Doctor Name

Bot Creation

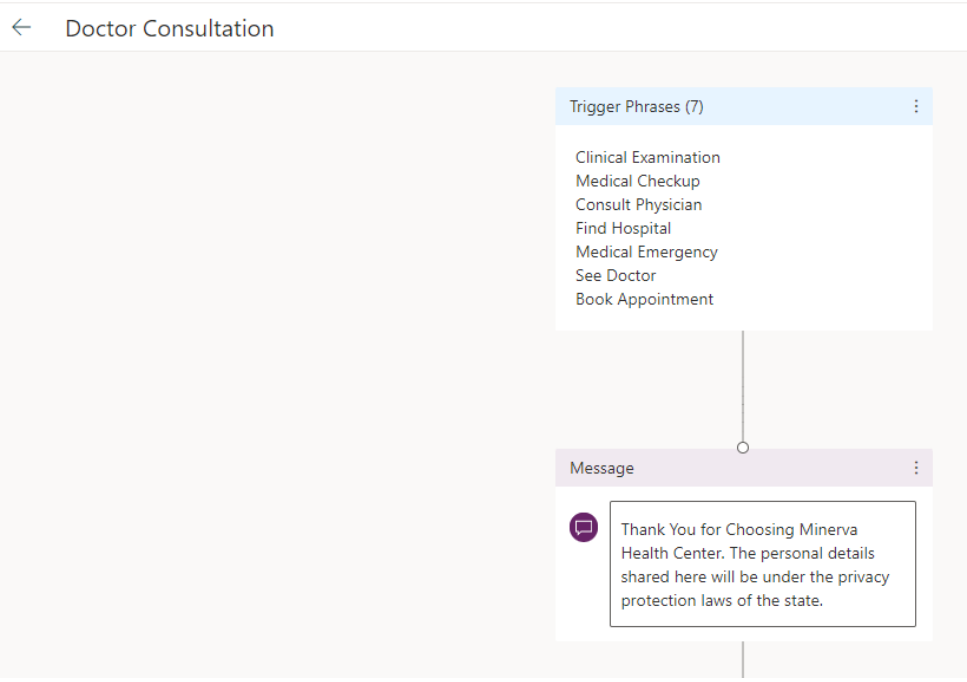
Let's create the bot by selecting the bot option from the top right corner and select "New bot". It will open up the window where we can specify the bot name and language. Once the bot is provisioned, we can create a new topic and select the trigger phrases to define the words that will invoke the conversation with the bot



The trigger phrases will help in defining the words and phrases that will start the conversation with the bot. We have defined 7 such phrases and the more different and related words we add to it, better the chances of bot understanding the trigger condition.

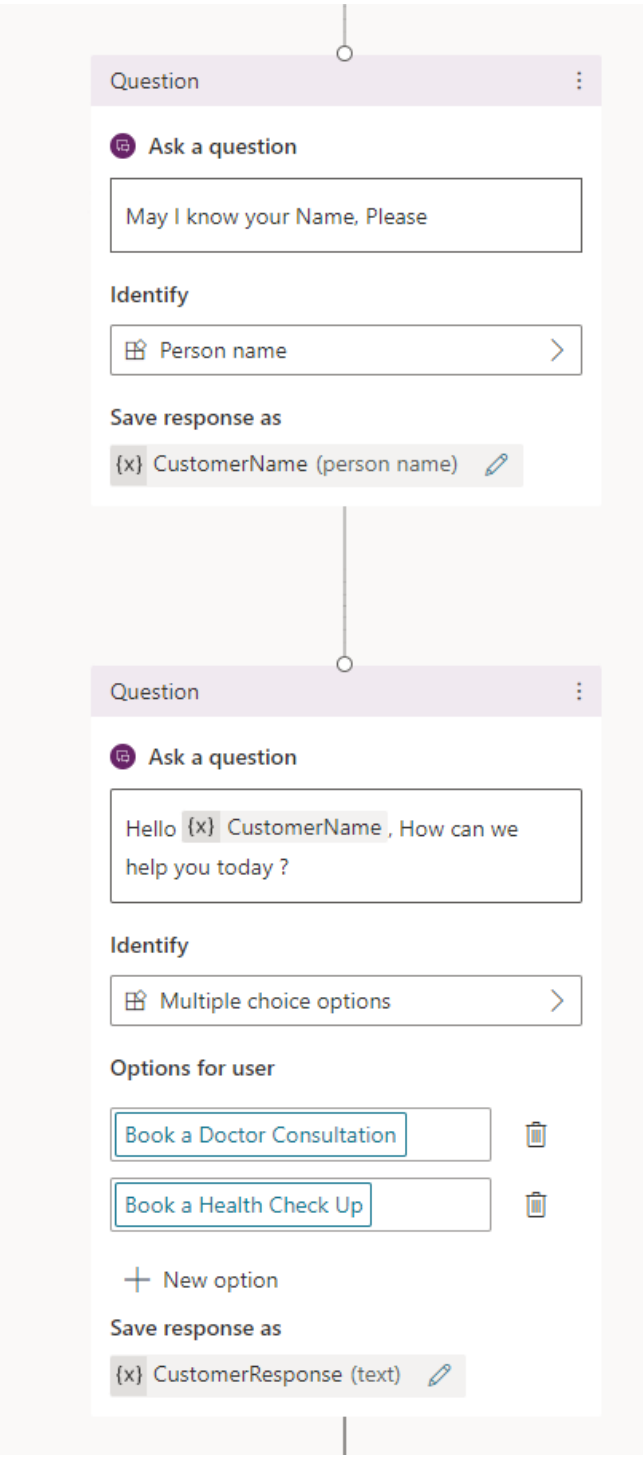


Let's add a welcome message to the Authoring Canvas

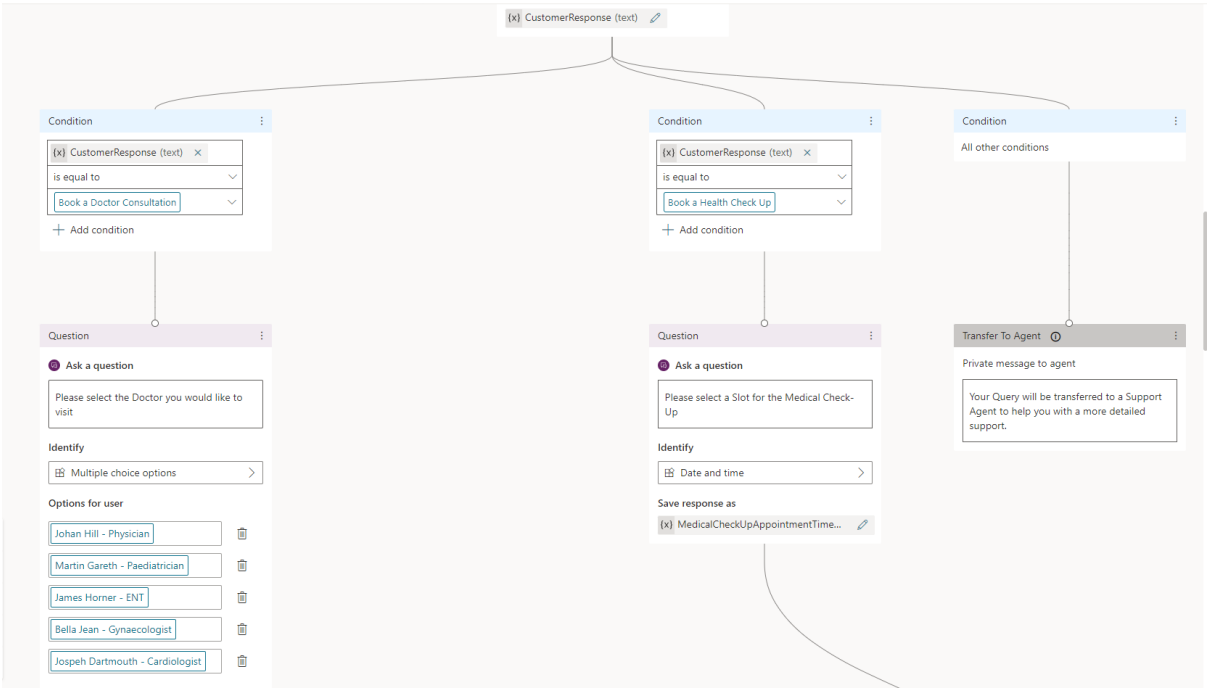


To get the customer's name, we will add the node – Ask a question to get the customer details. Followed by that, we will add another ask a question node to get the reason why the user has initiated the conversation by giving choice options :

- Book a Doctor Consultation
- Book a Health Check Up

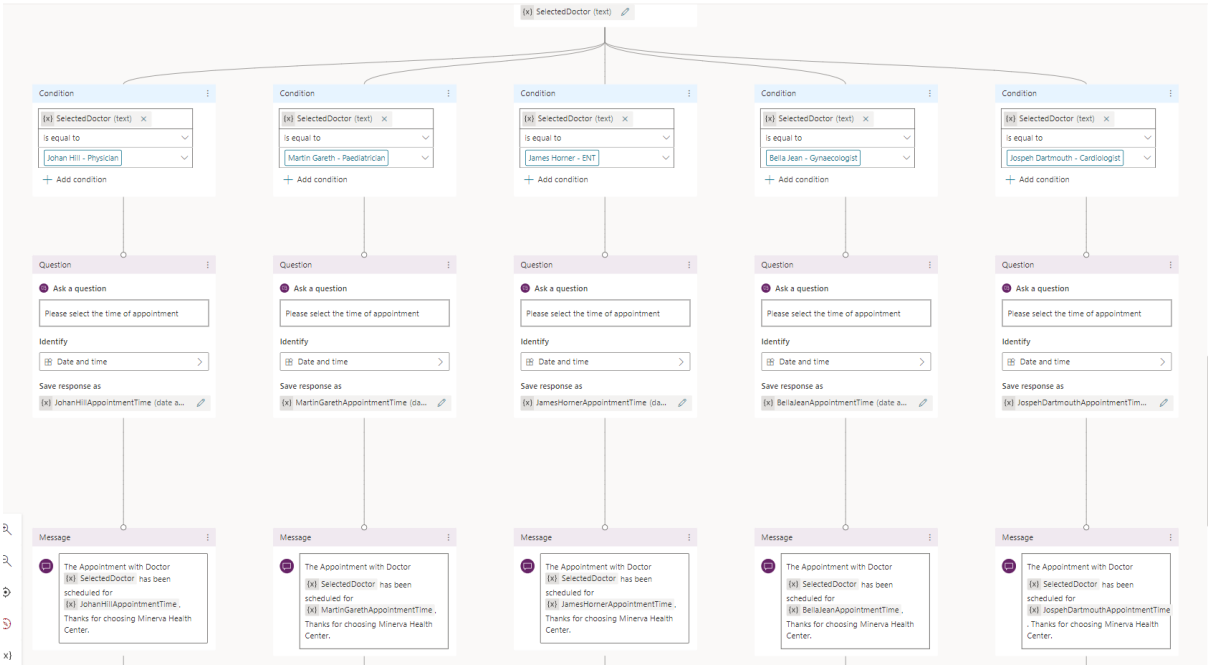


Based on the customer response, which is stored in the variable CustomerResponse, conditional branches will be automatically created in the authoring canvas. For the Book a Doctor Consultation branch, we will add another question, to give the user the option to select the doctor, while for the Book a Health Check up branch, we will give the user the option to type in the date and time for health check-up booking.

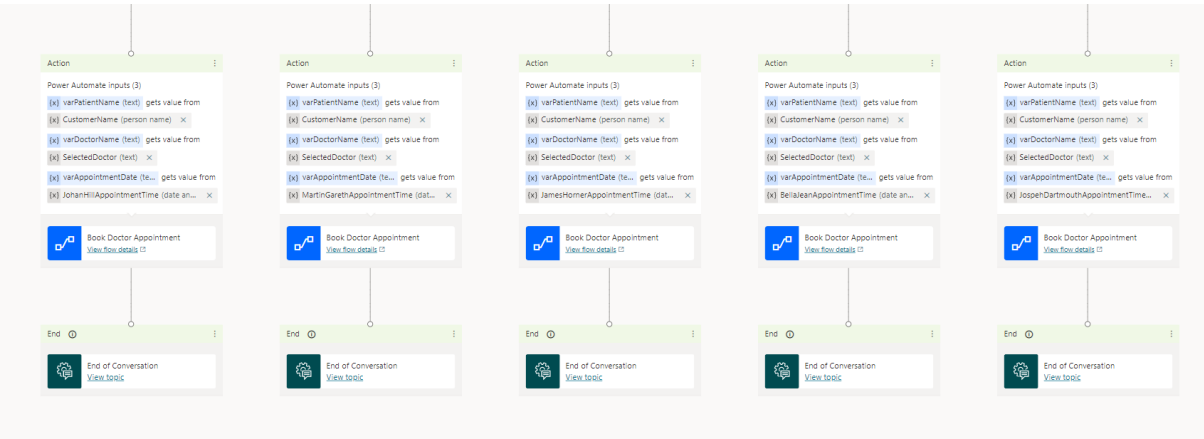


Book a Doctor Consultation

For each of the Doctor Selection, authoring canvas will again automatically create branches for each choice. For each branch, we will add the ask a question node to get the date and time of preferred booking and upon input from the user, we will show a message that the booking with the mentioned doctor has been booked.

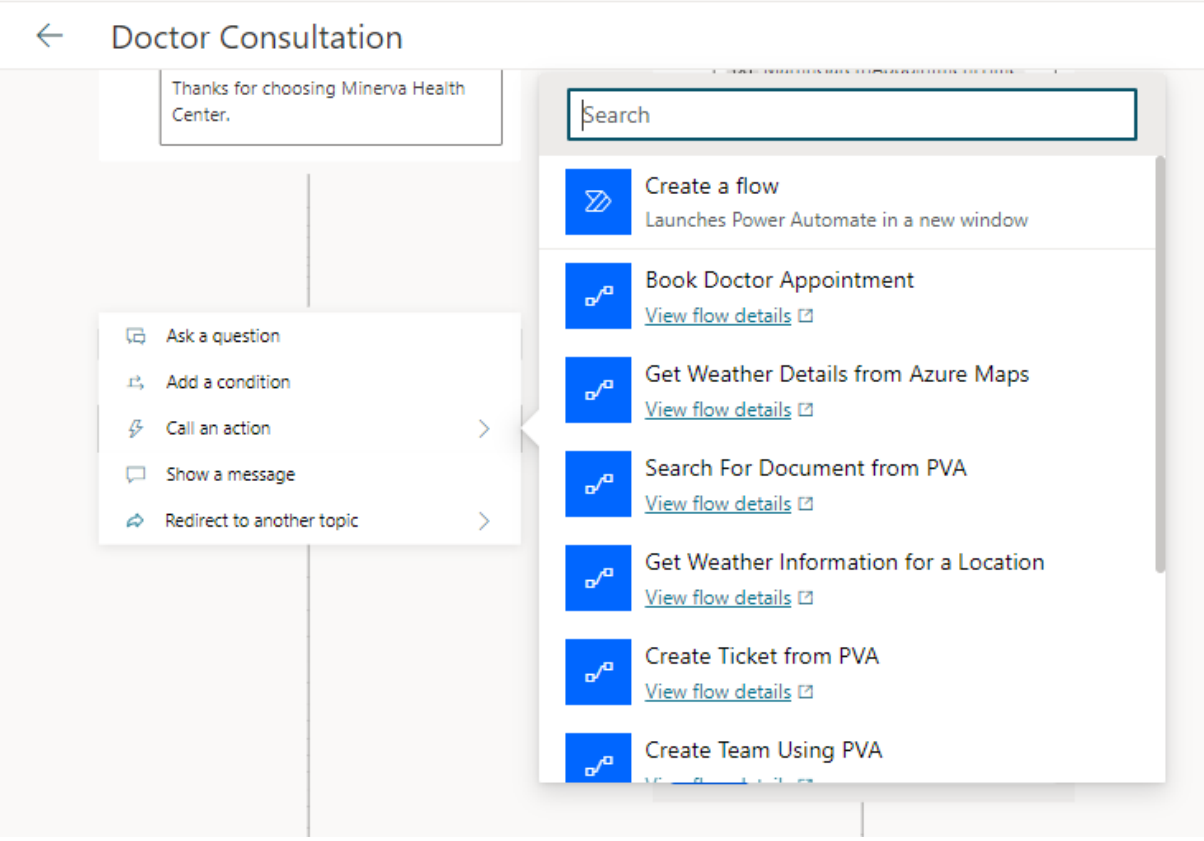


As a last step in this branch, we will add the call an action node and connect to the Power Automate which will save the details to the back end SharePoint list.

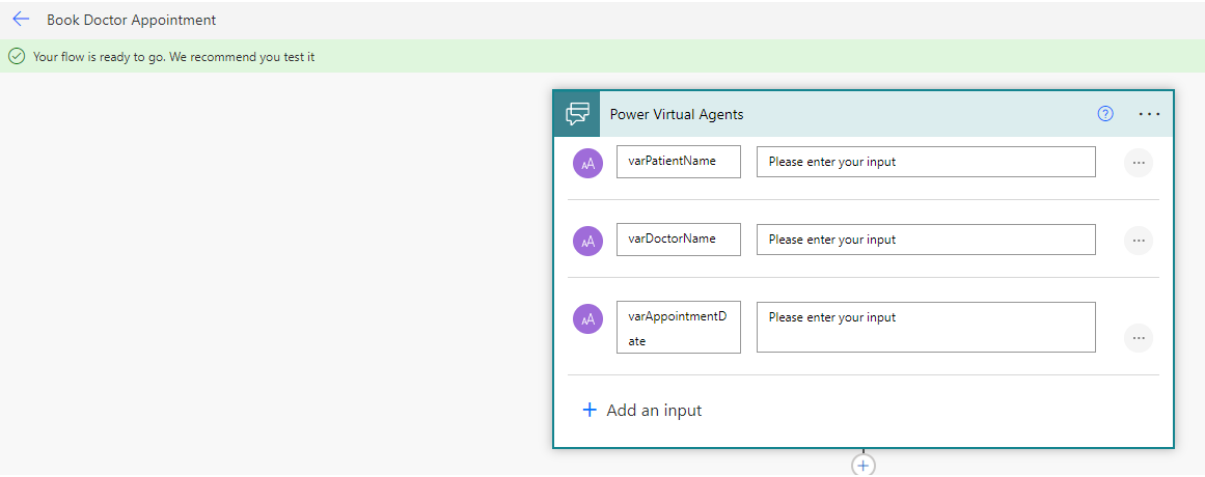


Building the Power Automate

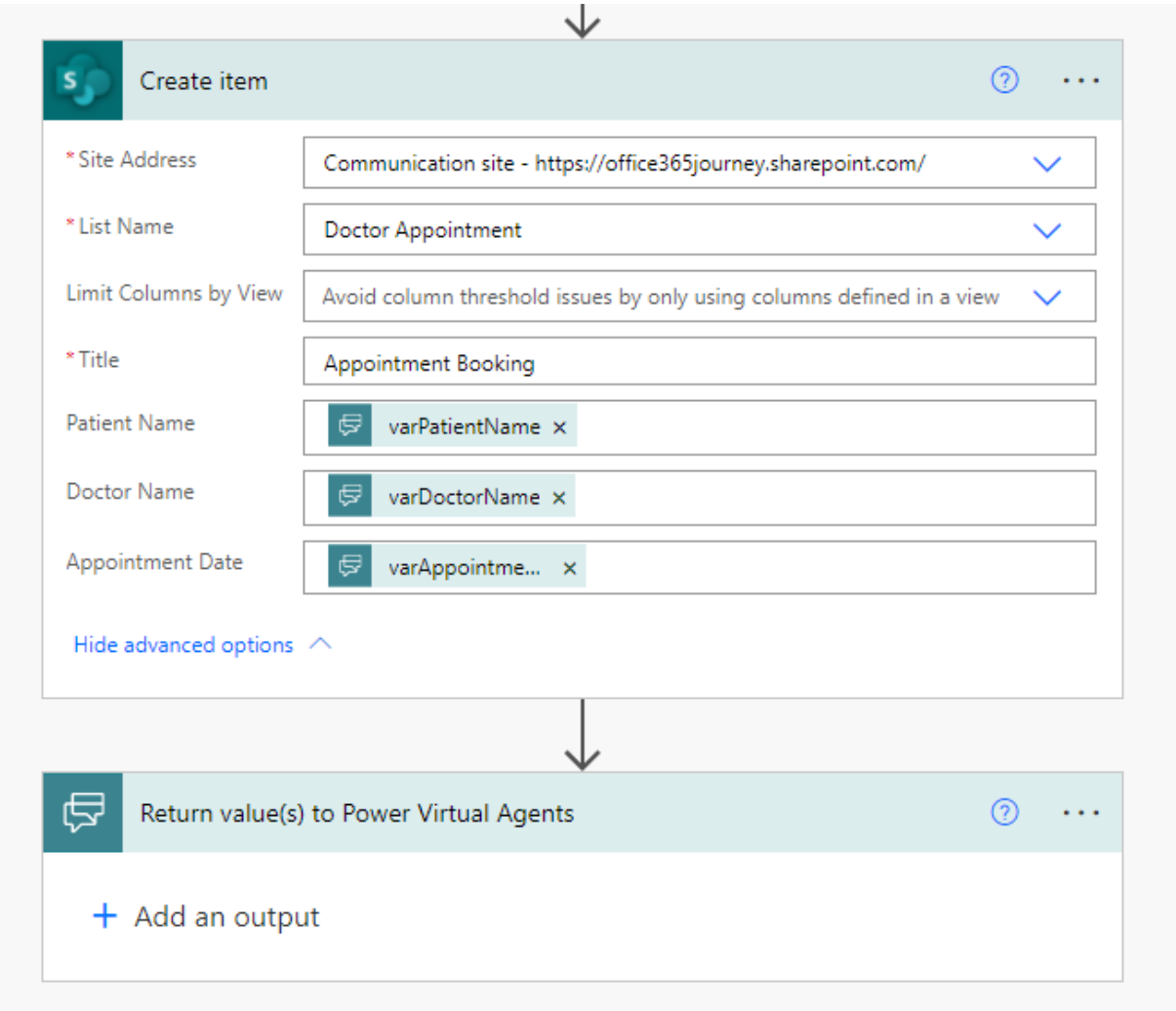
The Power Automate in the previous step is created by selecting the Call an action and clicking on Create a flow



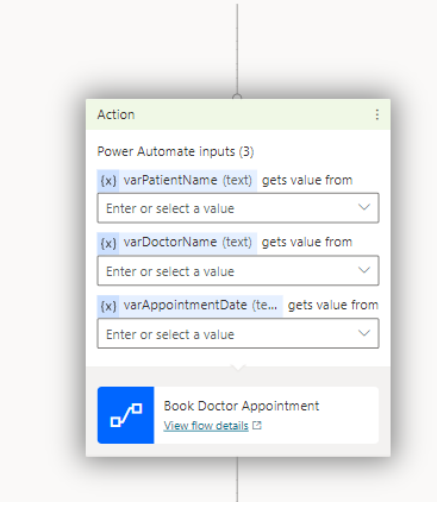
This will open the Power Automate window where we will define the 3 input variables which contains the Patient Name,Doctor Name and Appointment Time.



We will then add the create item action to save the details to the SharePoint list which will act as the appointment records which the hospital can pull information daily. As we don't have any specific data to be returned to the PVA, we will leave the return values empty. If we want to add additional appointment checking logic during item creation, we can return the success/failure message back to the PVA to make the system more robust.

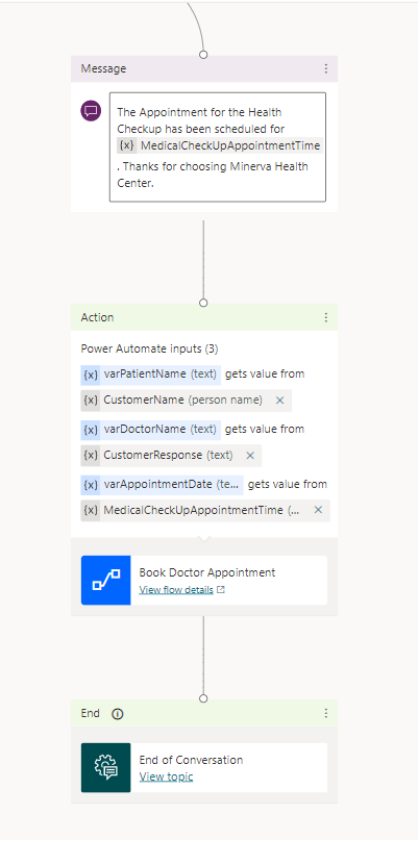


Once the Flow is created, we can add the call an action node and select the recently created power automate flow and specify the required variables in the input section. We have done this for all the 5 branches that we saw in the above section.



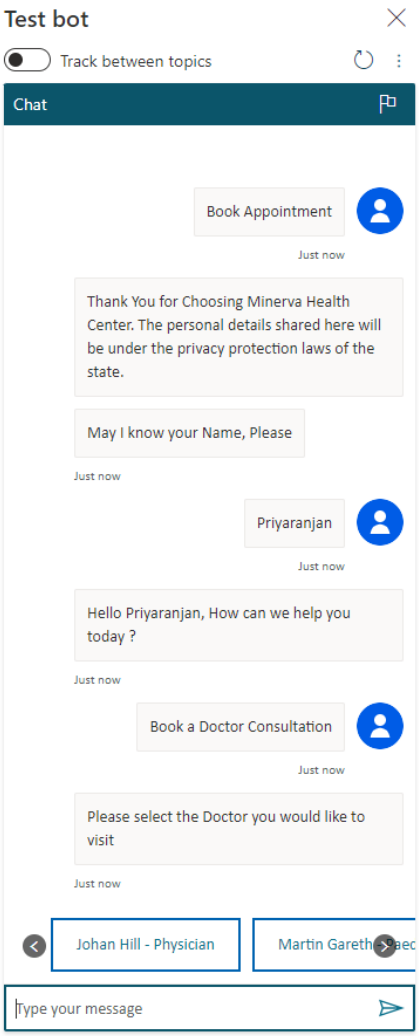
Book Health Check up

In case of health check-up, we will follow a similar pattern, where we will add the message to the end user that the booking is done and call the Power Automate and pass the required values which will be saved to the SharePoint list. Finally, we will end the conversation with a survey, if we intend to add more logic, we can extend the conversation with more question actions.

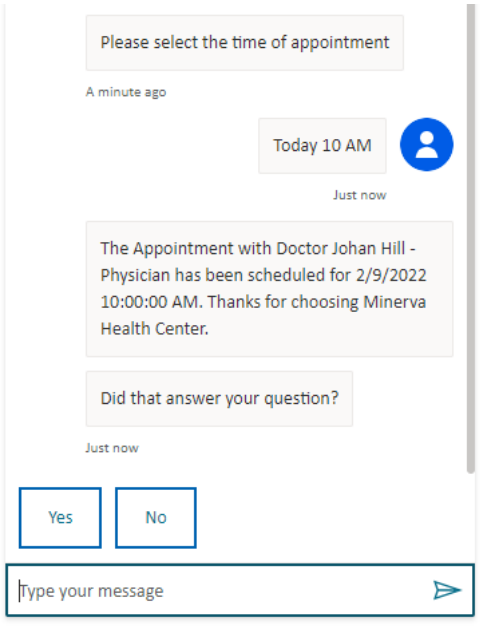


Test the bot

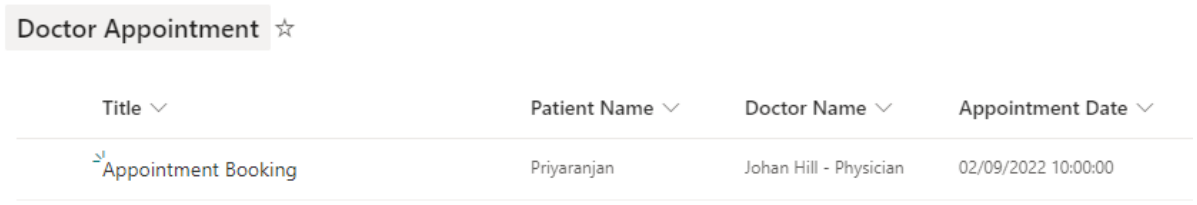
Now let's test the bot with a trigger word – Book Appointment



Upon giving the inputs and selecting the doctor, we can specify the date and time. We can enter a specific date or even enter free text like Today which will be converted into equivalent date and time.



The appointment record has been created in the back end as well:



Summary

Thus, we saw how we can use Power Virtual agents to create automate doctor appointment booking system that invokes power automate to save the appointment records to the back-end SharePoint list. This can be expanded to implement enterprise scale appointment bots and published to Public Facing website as well as other channels like Teams, slack, Facebook etc:

Getting Started with Markdown Language

Introduction

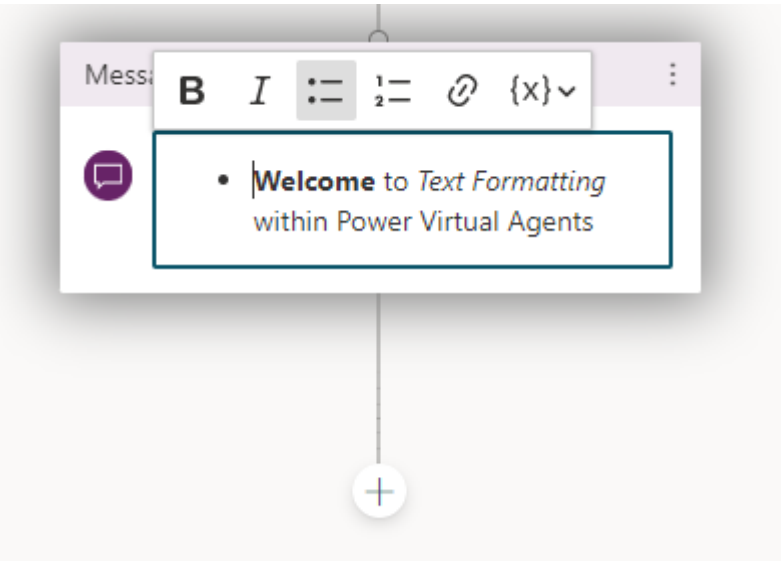
When we interact with Power Virtual Agents formatting the displayed text with bold/italics, showing ordered list of information, or displaying tabular data and images elevates the user experience. However, we cannot use the most reliable HTML tags to create the needed textual changes within Power Virtual Agents.

As of now only Markdown language is supported to display the textual formatting within Power Virtual Agents. Markdown is a lightweight markup language for creating formatted text which dates to 2004. Markdown is being widely used to make formatting changes and editing the documents in platforms like Github.

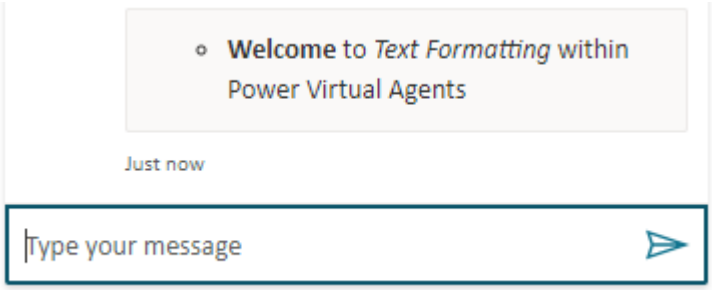
Let's see the basics of working with the Markdown language within Power Virtual Agents.

Basics

The markdown syntax does not work directly when applied with Power Virtual Agents. By default, the Send a message action in Power Virtual Agents have few formatting options that lets you beautify the text.

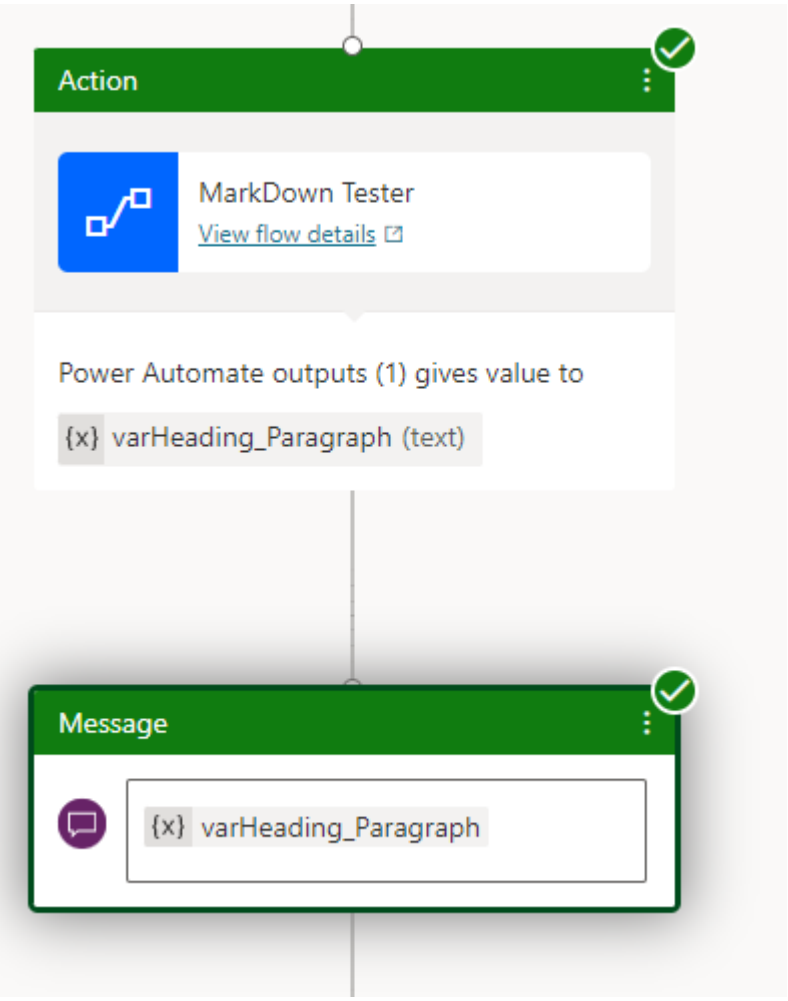


It will provide the output as :



However, beyond this, there is no inherent formatting options available as of now. So as to extent the formatting abilities, Power Virtual Agents however support displaying markdown language. But it comes with a caveat, if we use it directly within the Message node, the markdown will be ignored. So

the only way to add markdown is to use Power Automate where we can define the markdown text and share back the variable holding the mark down data which can be displayed using the message box. So we will call the Power Automate and accept the markdown variable and show it as :



Let's see some basic formatting samples that we can achieve within Power Virtual Agents using Power Automate.

Heading and Paragraph

We can define the heading levels by using the # symbol followed by text. The number of # that we use will determine the size of the heading.It is reqivalent to the H1-H6 heading tag in HTML.

For instance, we can use up to 6 # to define the heading levels with Single Hash indicating the largest heading size.

This is heading 1

This is heading 2

This is heading 3

This is heading 4

Heading level 5

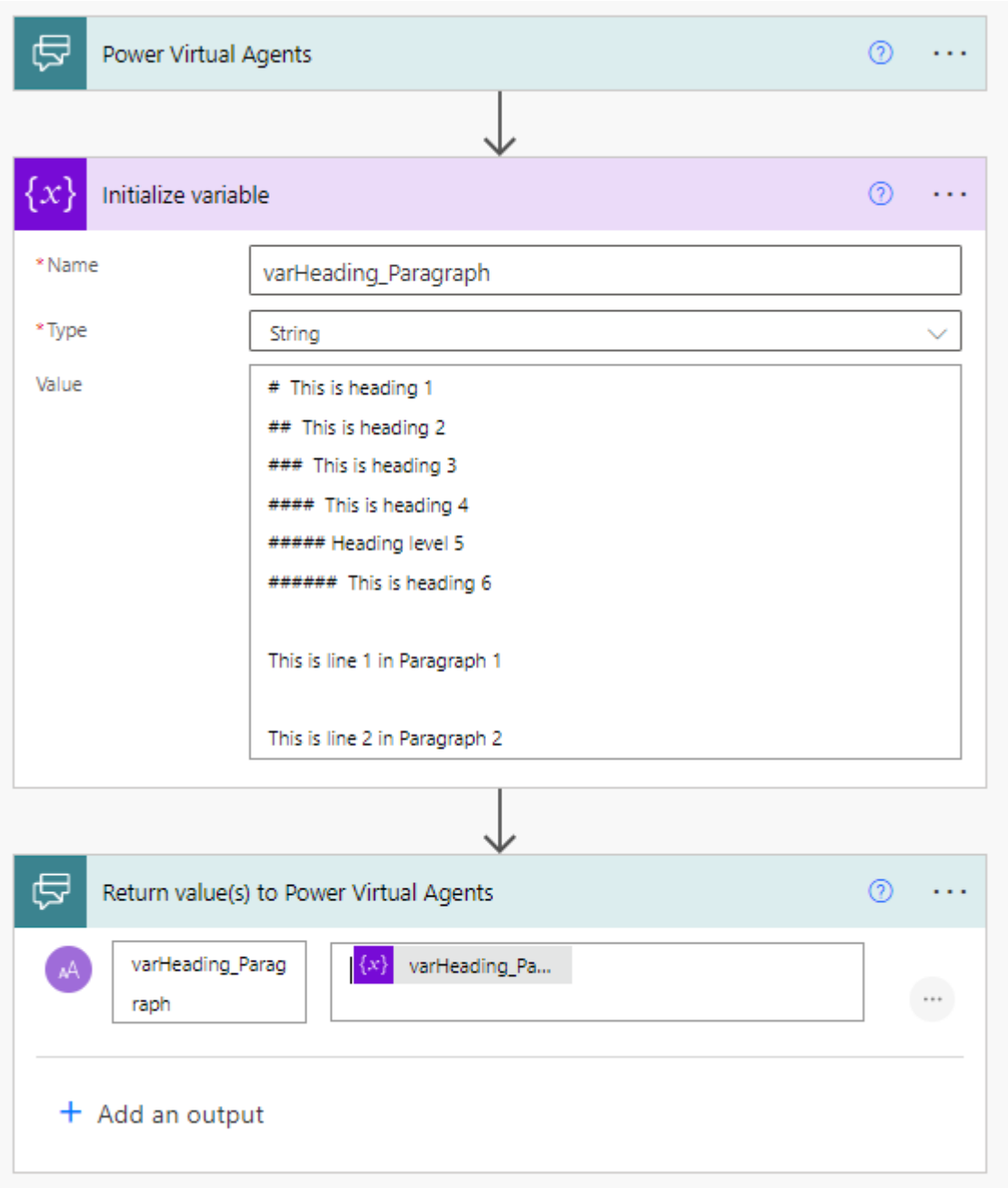
This is heading 6

To create paragraphs, we will use a blank line to separate one or more lines of text as below :

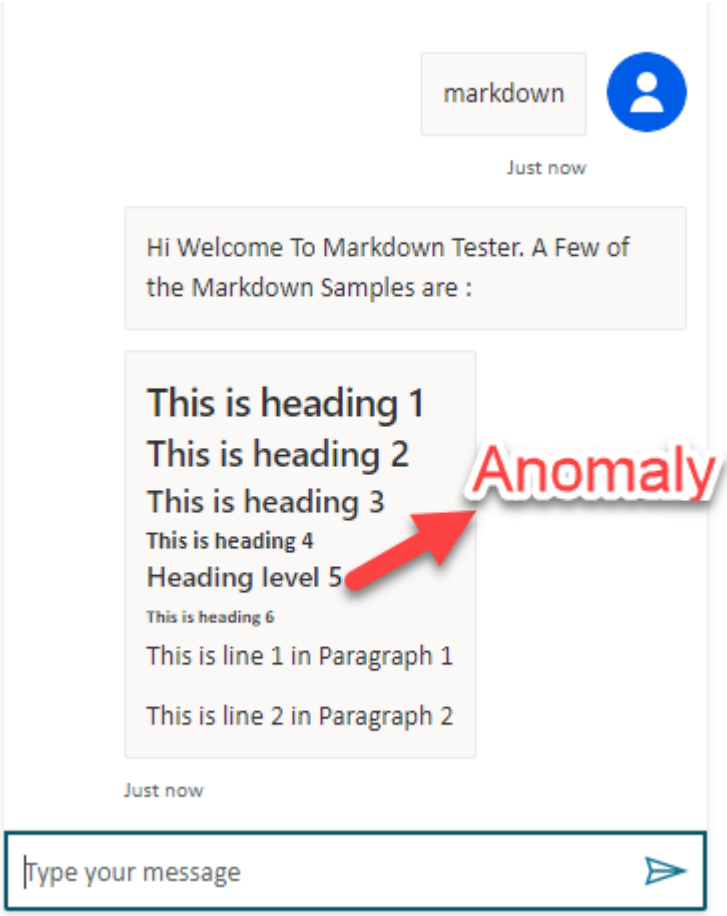
This is line 1 in Paragraph 1

This is line 2 in Paragraph 2

We will be calling the Power Automate from within Power Virtual Agents to define the markdown in a variable and will send it back to the Power Virtual Agent as below:



It outputs the markdown in chat window as:



As we can see the Heading and Paragraph markdowns have rendered but the Heading with 5 Hashes(H5) doesn't work as expected which is by design.

Bold and Italics

We can now try to add Bold and Italic formatting to the text. A text is made Bold by adding 2 asterisks before and after the words as :

I am a Bold Text

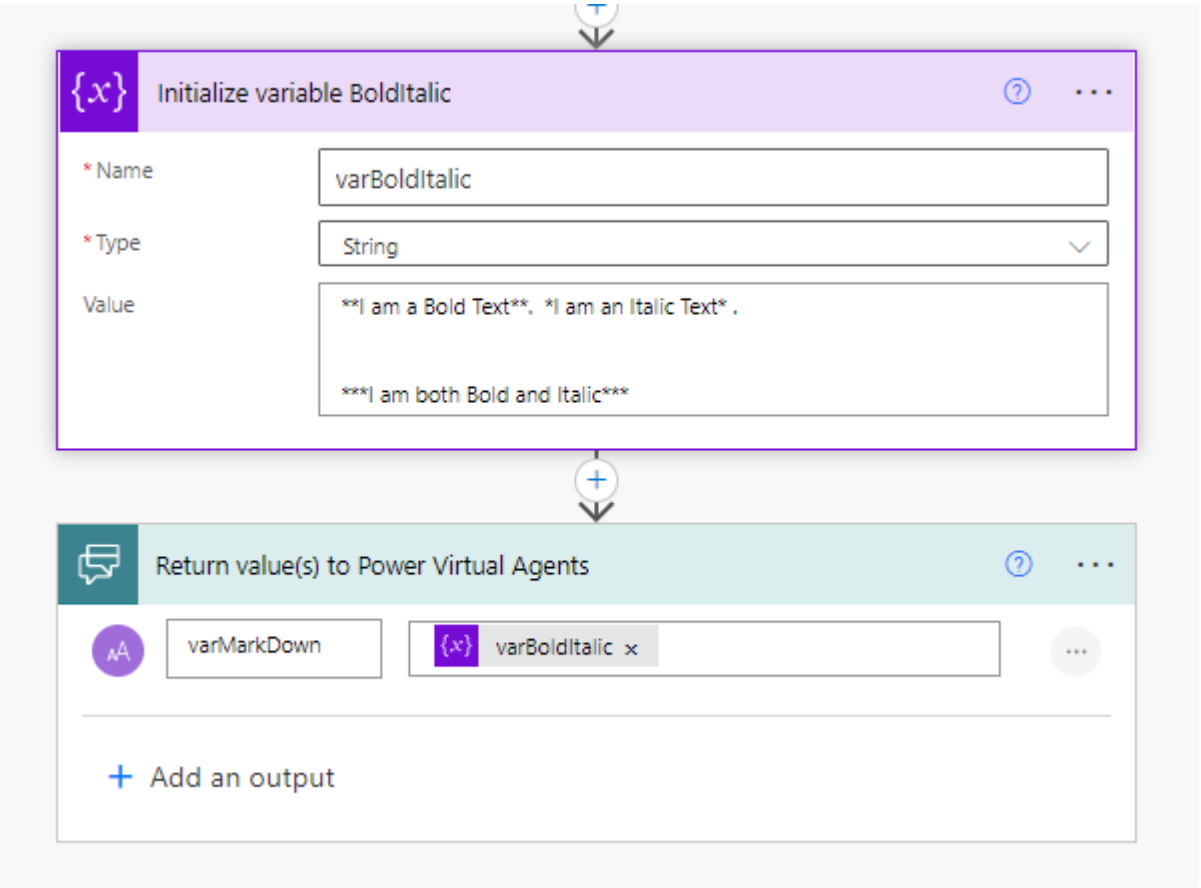
A text is made italics by surrounding it with single asterisks

I am an Italic Text

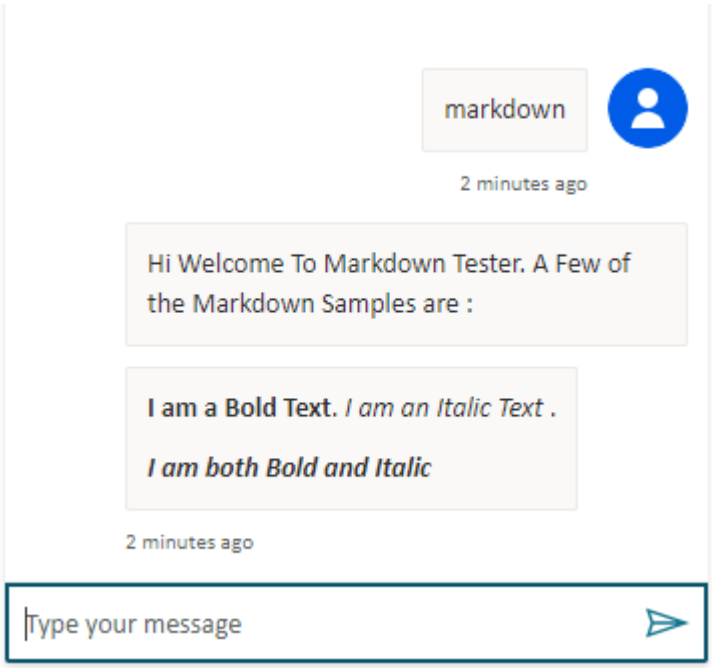
While the text can be made both bold and italic by surrounding it with 3 asterisks

I am both Bold and Italic

In Power Automate , it would look like :



The output in Power Virtual Agent will look like :



Block quote

Ideally in the markdown world, we can create a block quote by appending the text with the > symbol. But in Power Automate, it is just ignored. So not all markdowns are respected AS IS in Power Virtual Agents for now.

Ordered and Unordered Lists

A very common use case is to list down the data as bulleted or numbered lists. We can create the ordered list by starting the text with a number followed by a period and the relevant data as below. Same way we can indent the items as well.

1. First List item

2. Second List item

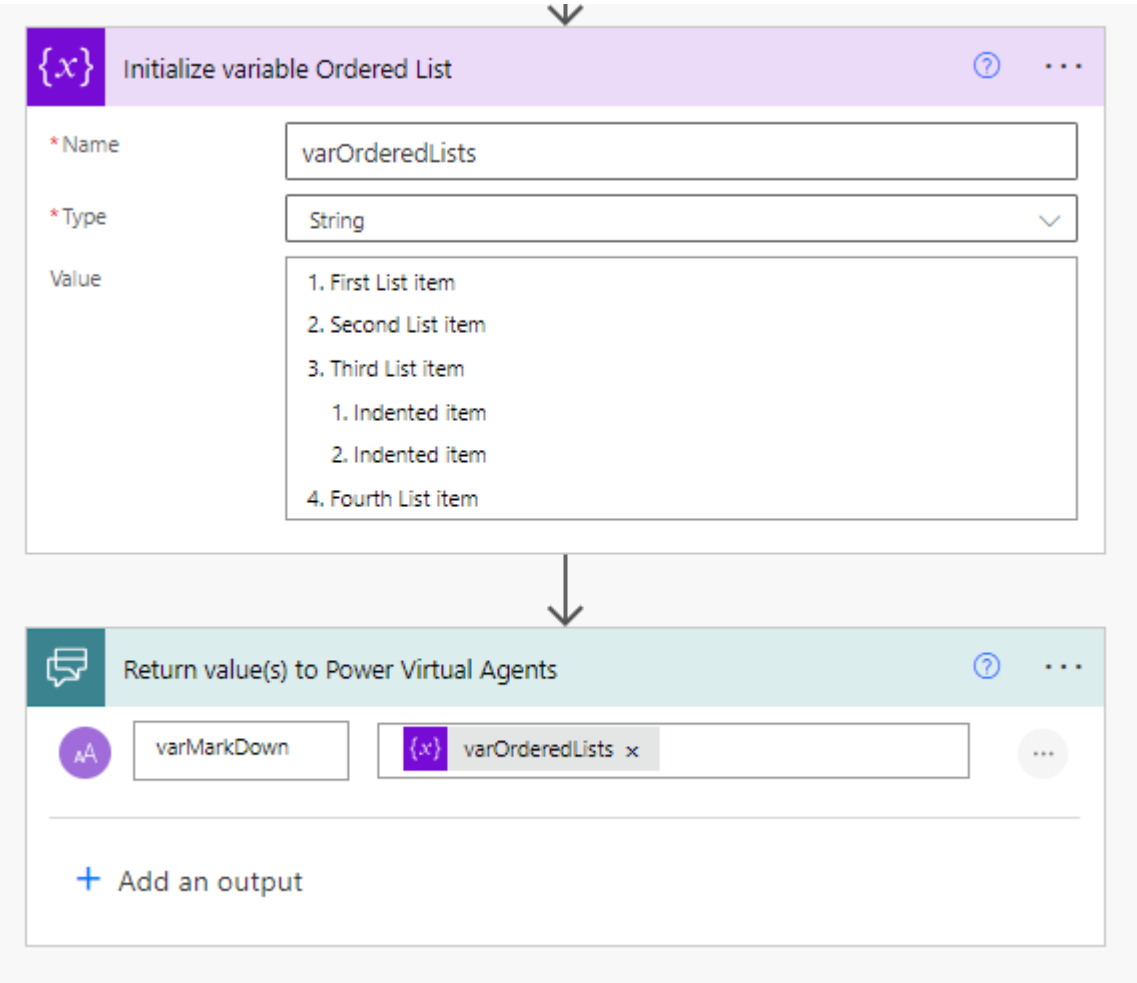
3. Third List item

    1. Indented item

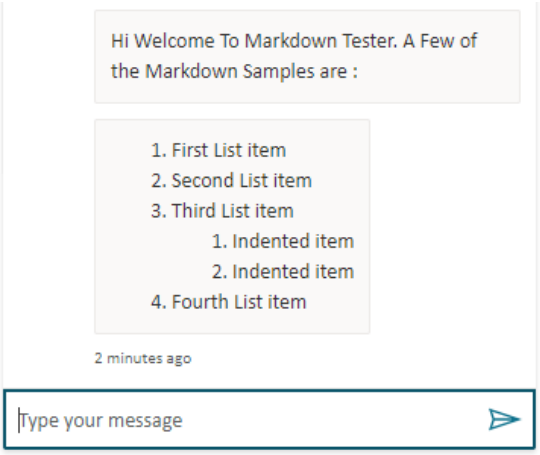
    2. Indented item

4. Fourth List item

In Power Automate it would look like :



The PVA output will the markdown as :



To create an unordered list we can add dashes (-), asterisks (*), or plus signs (+) in front of the text and indentation can be done by adding dashes/ asterisks/ plus signs as

* First List item

* Second List item

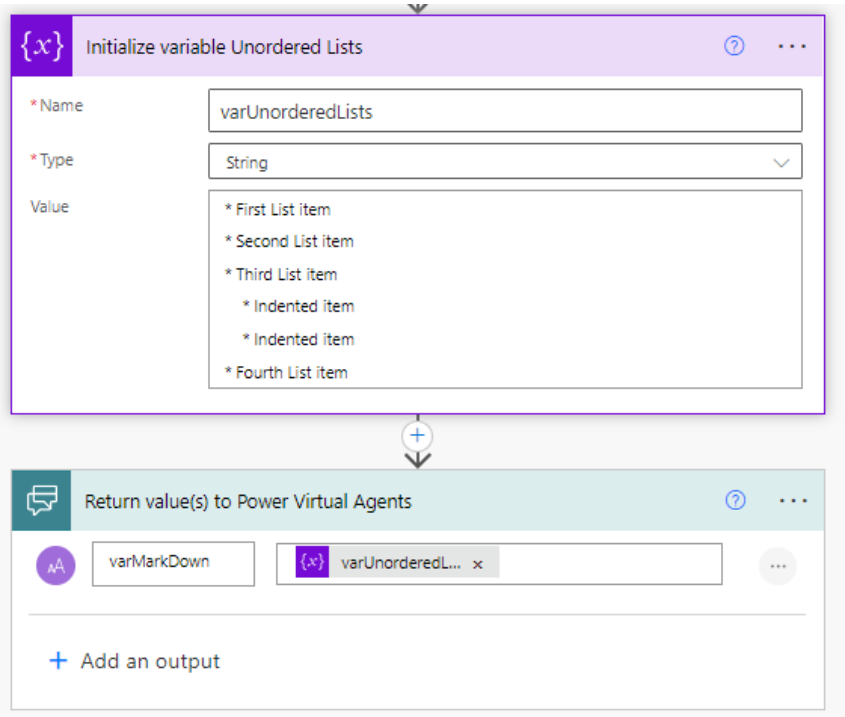
* Third List item

  ** Indented item

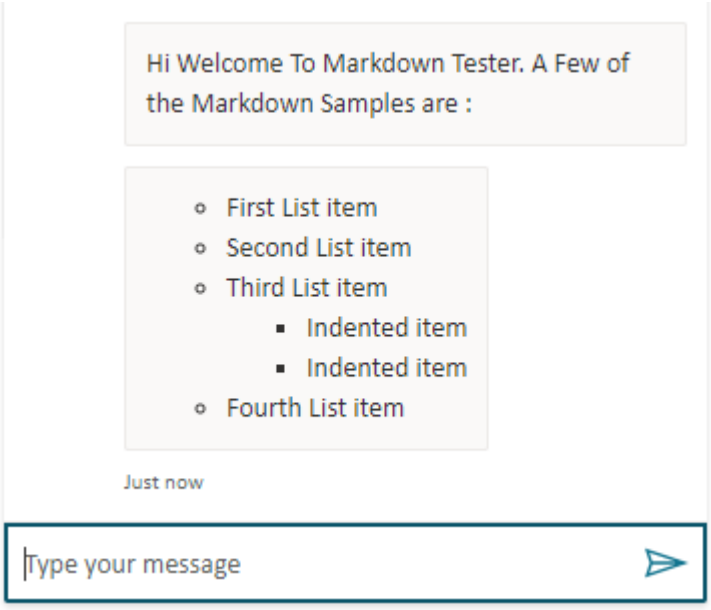
  ** Indented item

* Fourth List item

In Power Automate it would look like :



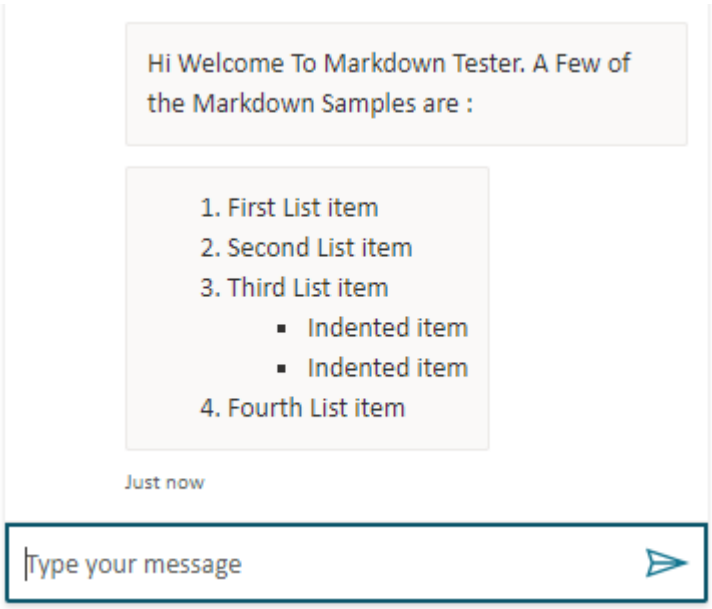
The markdown output in PVA will be as below :



We can also nest the Ordered and Unordered lists as :

1. First List item
2. Second List item
3. Third List item
    * Indented item
    * Indented item
4. Fourth List item

Which would result in the below output



Table

To add a table, we use three or more hyphens (---) to create each column's header and use pipes (|) to separate each column.To add the rows we will again use pipe as separator or each cell as below :

|Name|Department|

|----------|----------|

|Priyaranjan|Modern Work Place|

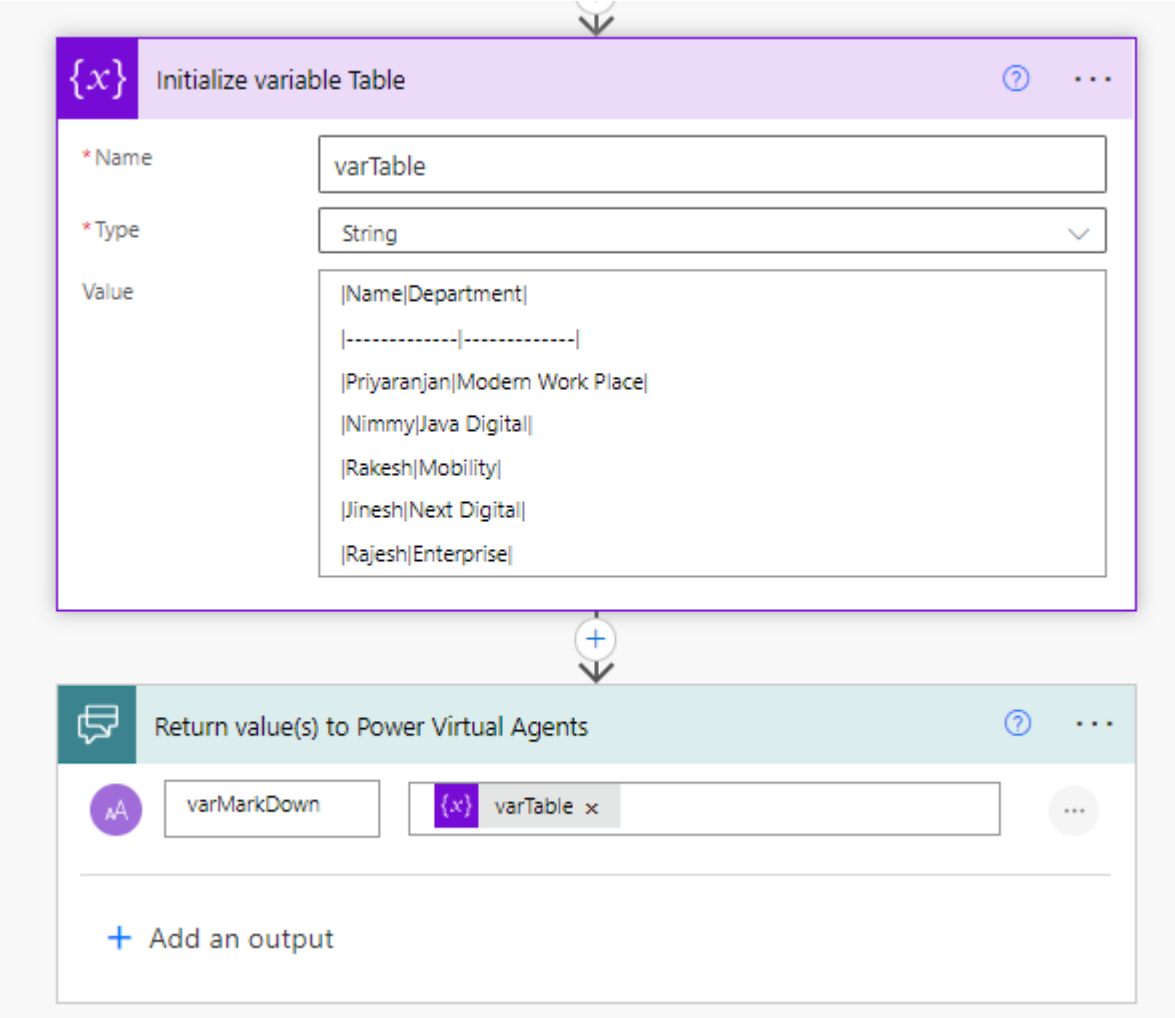
|Nimmy|Java Digital|

|Rakesh|Mobility|

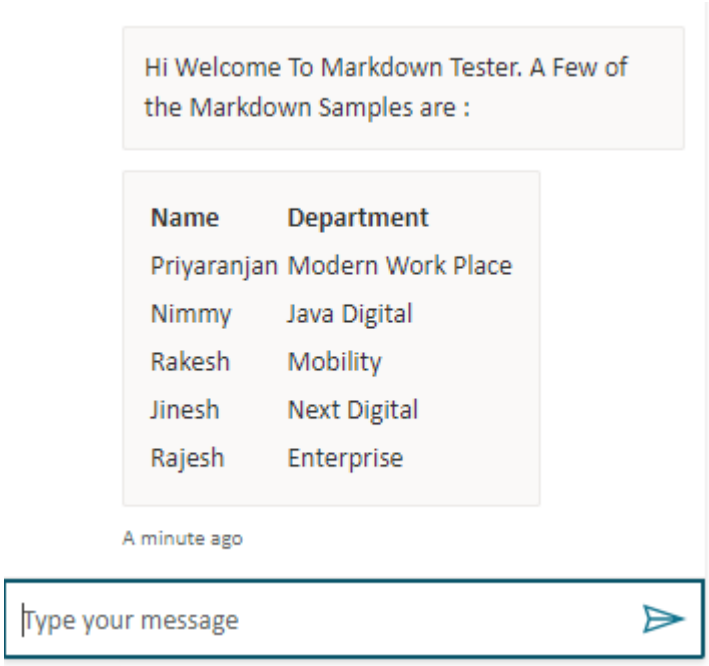
|Jinesh|Next Digital|

|Rajesh|Enterprise|

We will add it to Power Automate as :



Which will give the markdown output in PVA

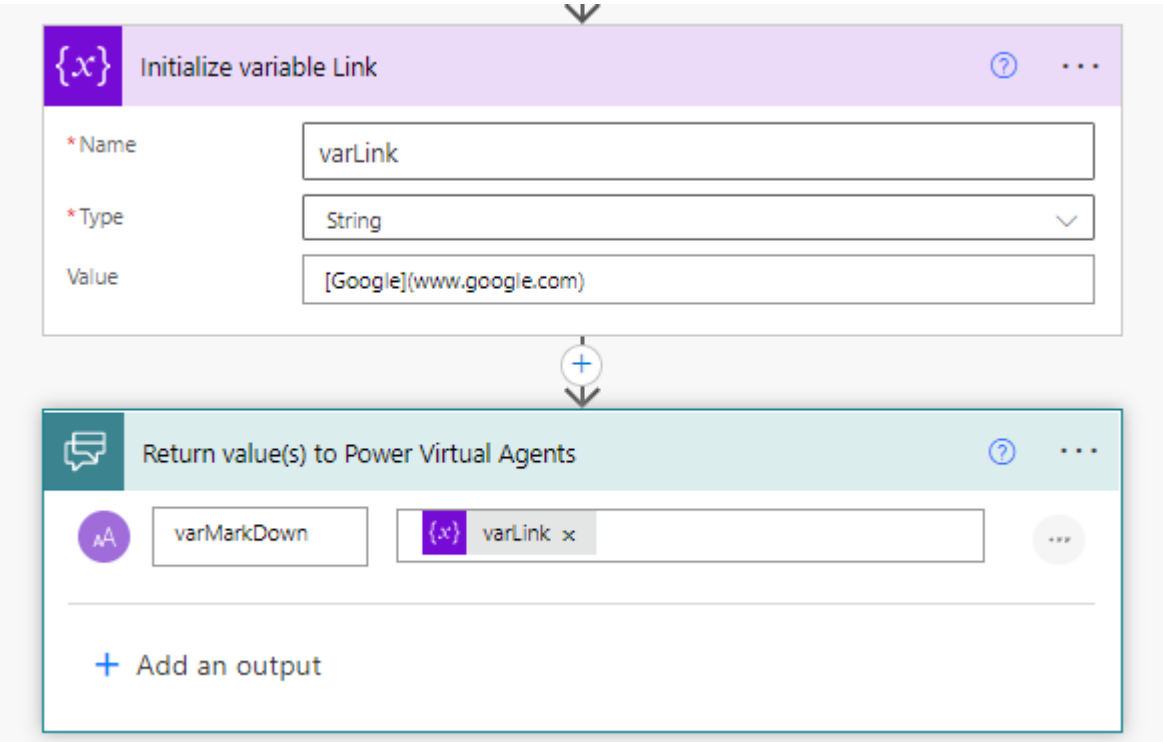


Links

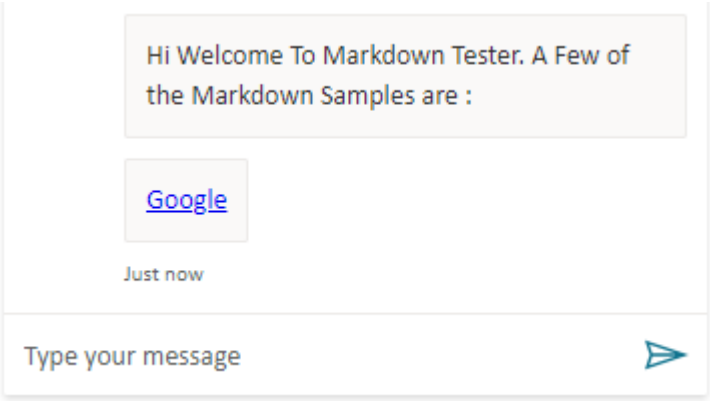
We can also format the text as links using the below syntax to display a clickable link in PVA

[Google](www.google.com)

We will add it to Power Automate as:



Which will give the output in PVA

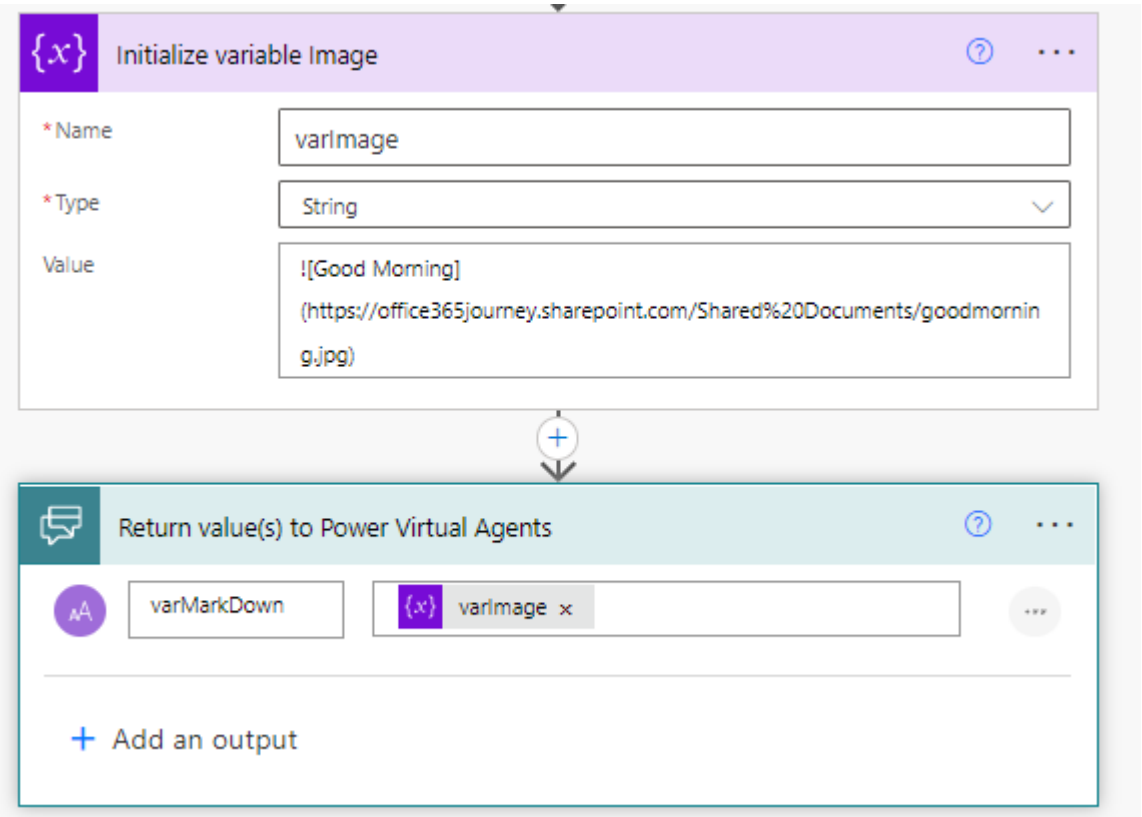


Images

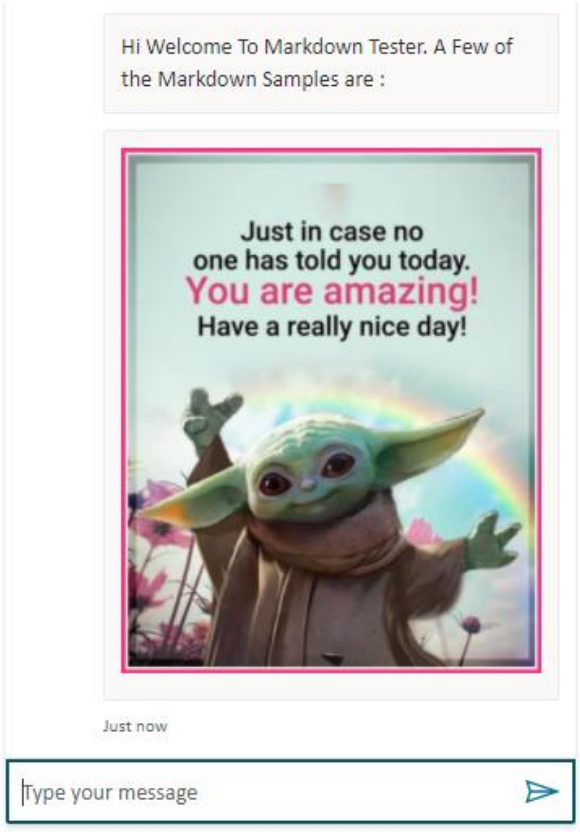
To render an image using markdown, we will follow the syntax

 ![alternative text](link to image)

We will add it to Power Automate as :



And the PVA output will come up as:



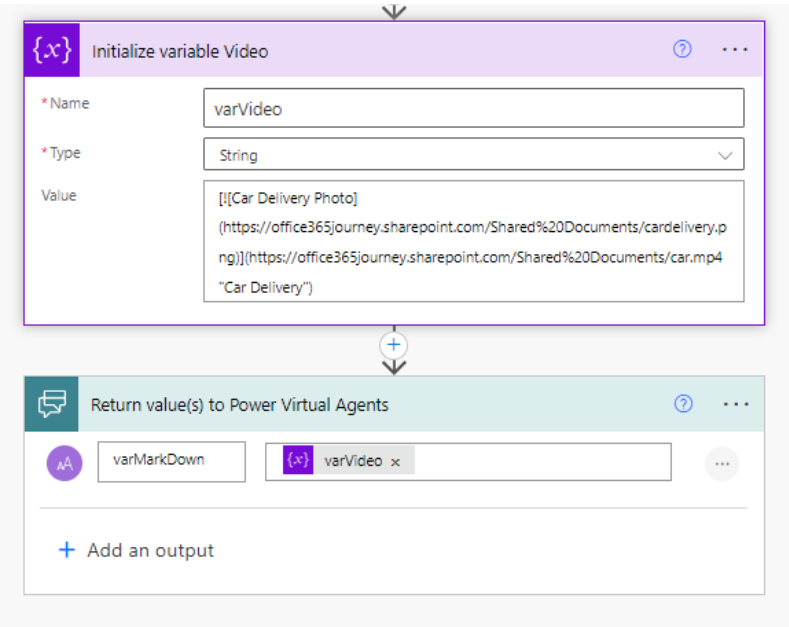
Video

Same way as adding the image, extending the markdown to the below syntax will let us add the Video to the PVA

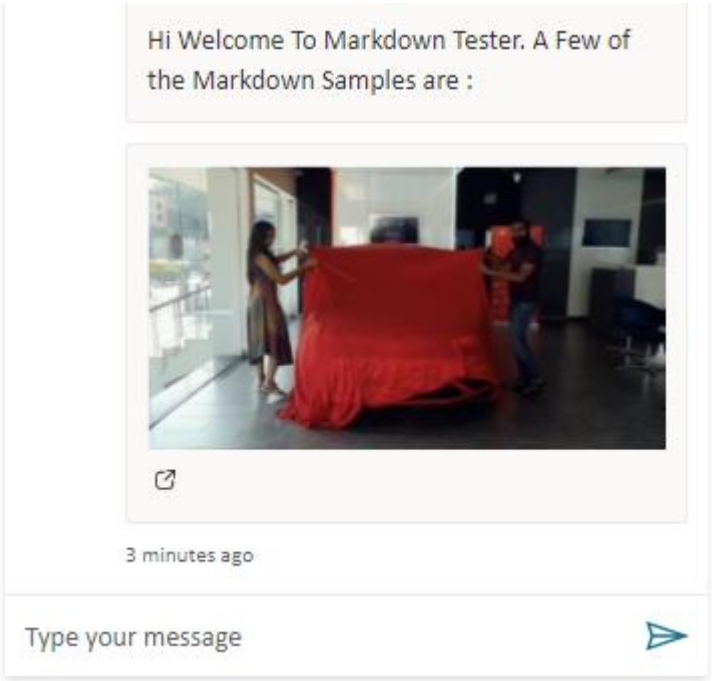
[![Alternate Text]({image-url})]({video-url} "Link Title")

Here we can add the Video Thumbnail URL, its alternate text, Video URL and the hover over video title

We can add it to Power Automate as :



Which will output the Video link in PVA as



It doesn’t have the flexibility to playback within PVA, clicking on it will take you to the respective web-based media players.

Summary

Thus, we have seen how to work with markdown language to format and beautify the conversations within Power Virtual Agents

Practical use case of using Markdown with Power Automate and Power Virtual Agents

Introduction

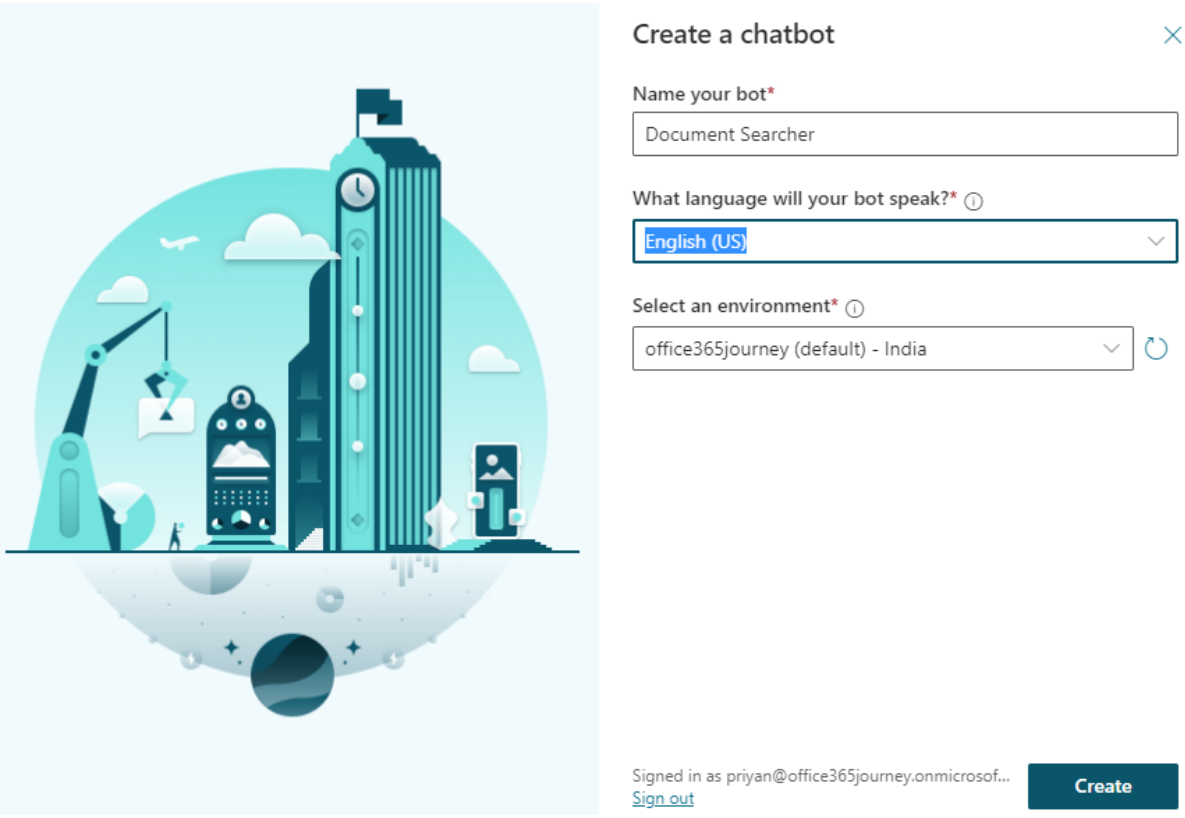
We did see the basics of markdown language. In this section we will see how to create a basic bot that can be used to search for documents in SharePoint and show them as a table using markdown language with the help of Power Automate.

Business Use Case

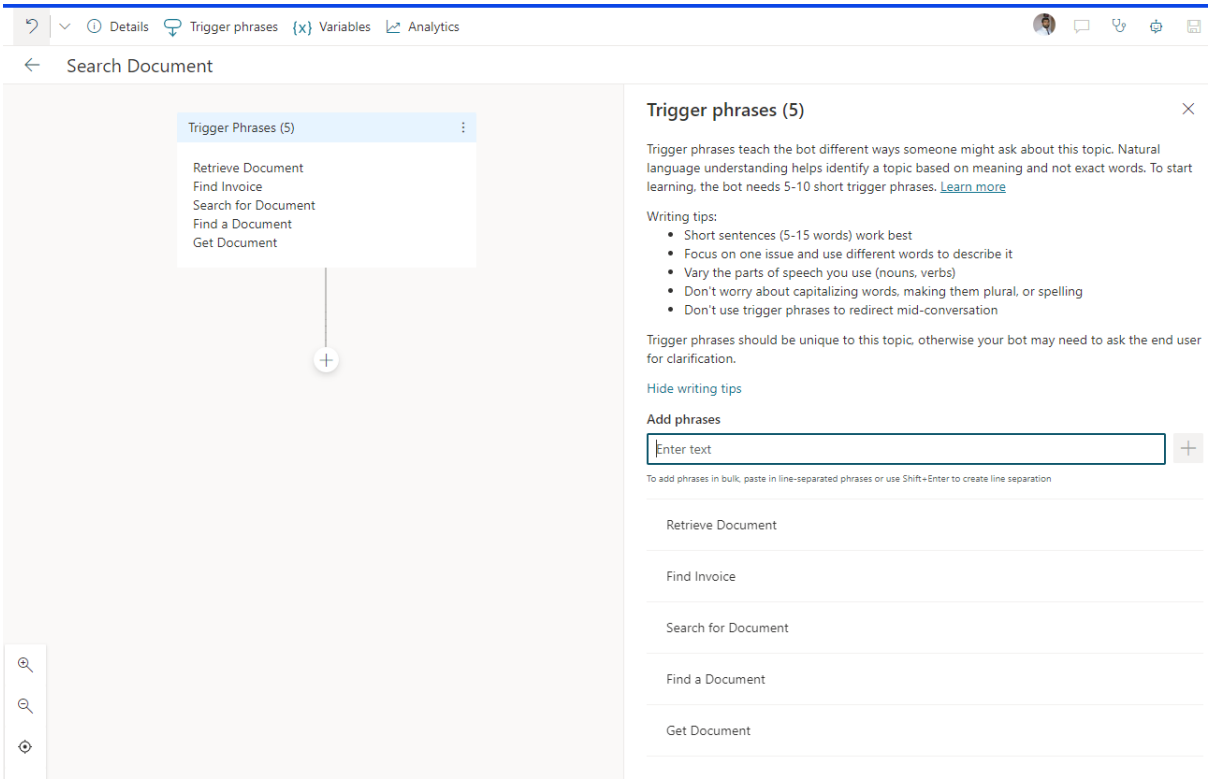
We have a requirement where SharePoint is used as a content repository and a single document library at times can have 1000s of documents. Rather than navigating to SharePoint and use the search option if we can setup a basic bot that can pick a document by name, it will be a productivity gain for the users.

Implementation

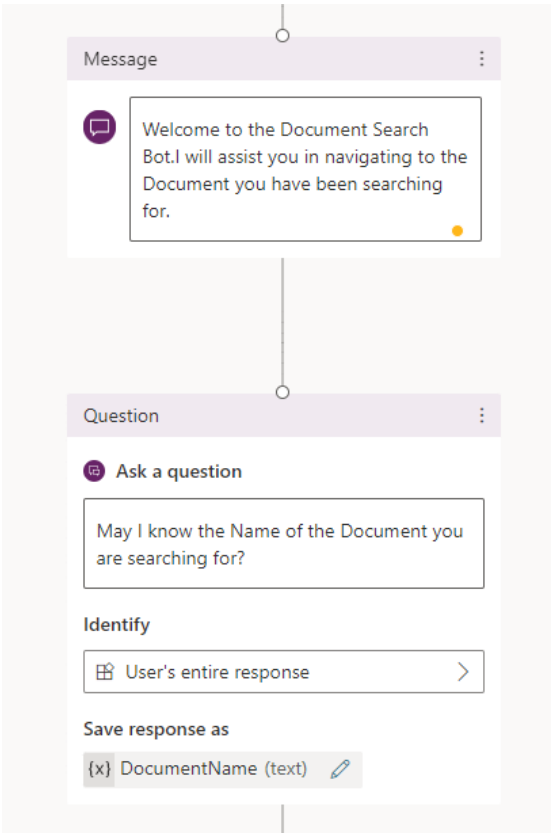
We will use the SharePoint Document library to store the documents and will build a bot that takes the input of the document name to be searched for. Let's create the bot by selecting the bot option from the top right corner and select "New bot". It will open up the window where we can specify the bot name and language. Once the bot is provisioned, we can create a new topic and select the trigger phrases to define the words that will invoke the conversation with the bot.



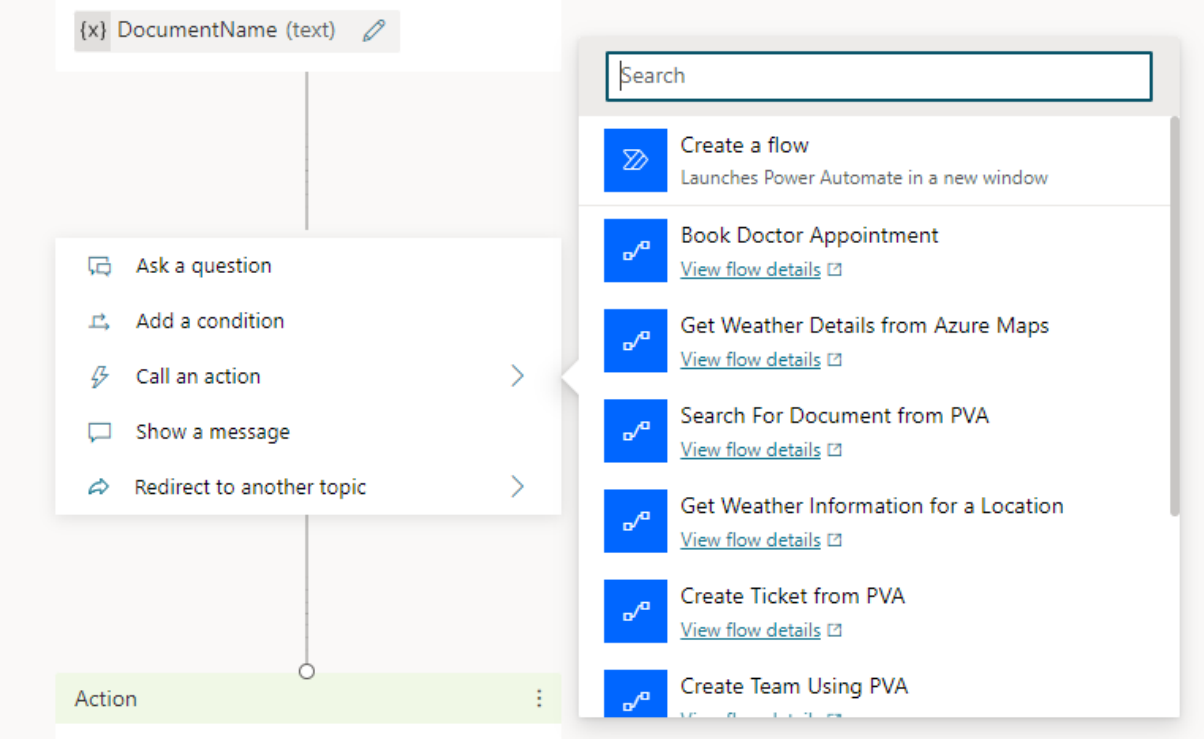
The trigger phrases will help in defining the words and phrases that will start the conversation with the bot. We have defined 5 such phrases and the more different and related words we add to it, better the chances of bot understanding the trigger condition.



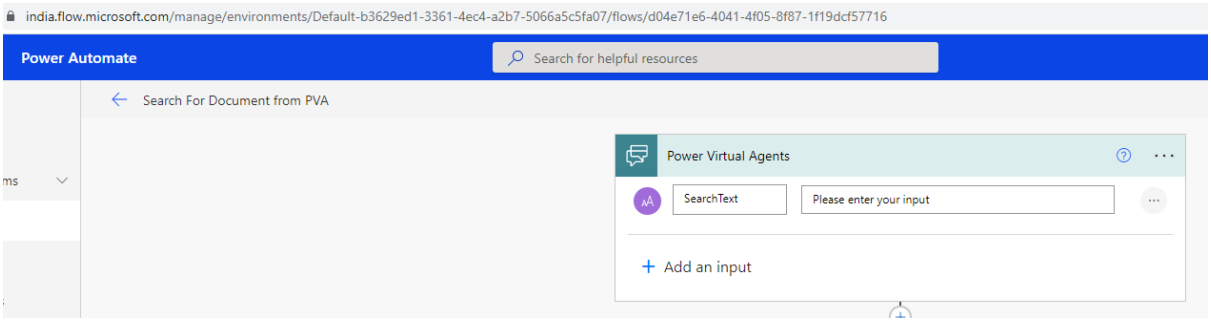
Let's add a welcome message to the Authoring Canvas and followed by that we will add the node ask a question which will request for the document name the user want to search for. The input from the user will be saved in the variable DocumentName.



Next, we will create a Power Automate to which we will pass the Document Name to do a query in the SharePoint list. To do this click on the Plus Sign that show the nodes available to be added to the canvas and select, Call an action which will give us the option to create the flow.



Clicking on Create a flow will open the Power Automate window where we will define a text input by the name SearchText which will accept the input from PVA.

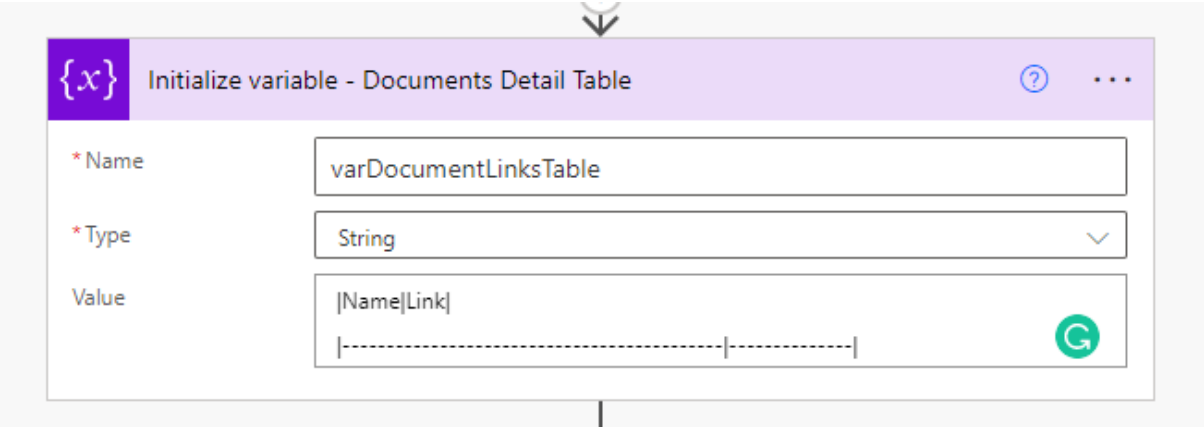


So as to display the weather details in the Power Virtual Agent as a table, we will be using mark down language and we will initialize a variable that will hold the header which is defined using the syntax:

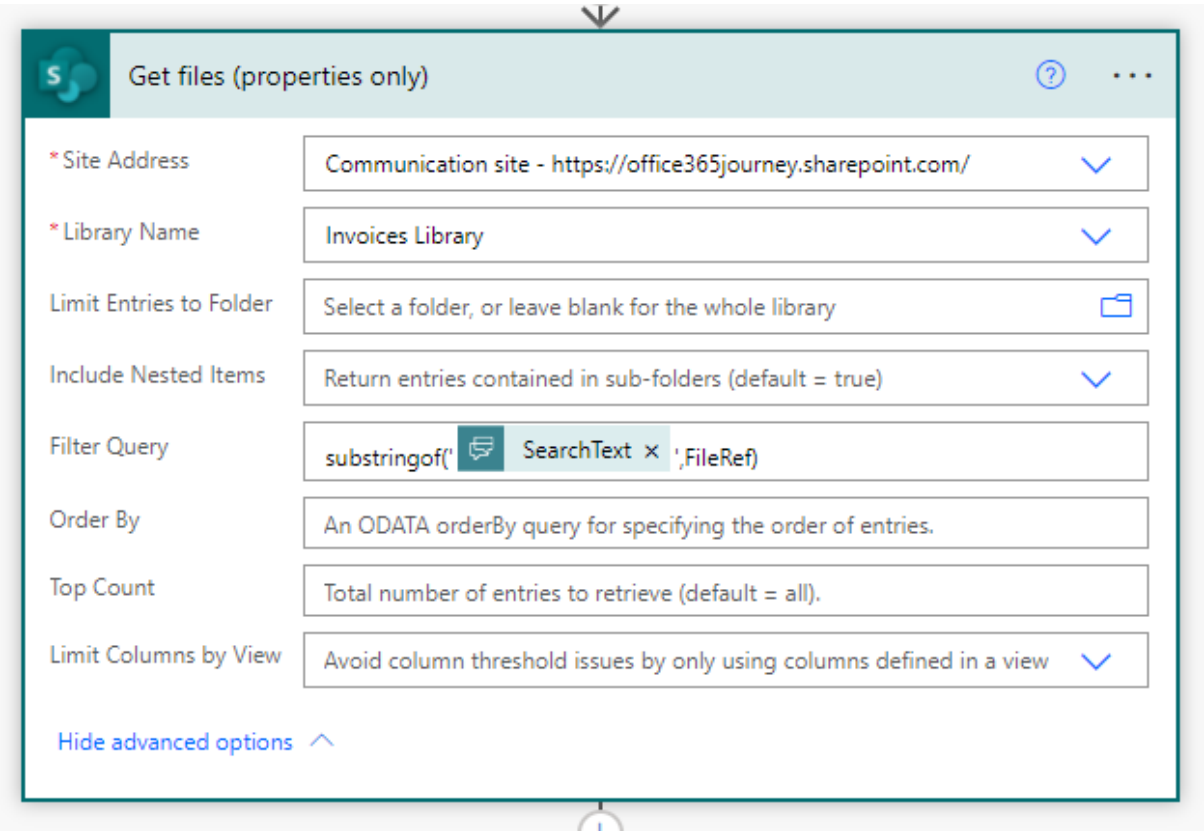
|Name|Link|

|----------|----------|

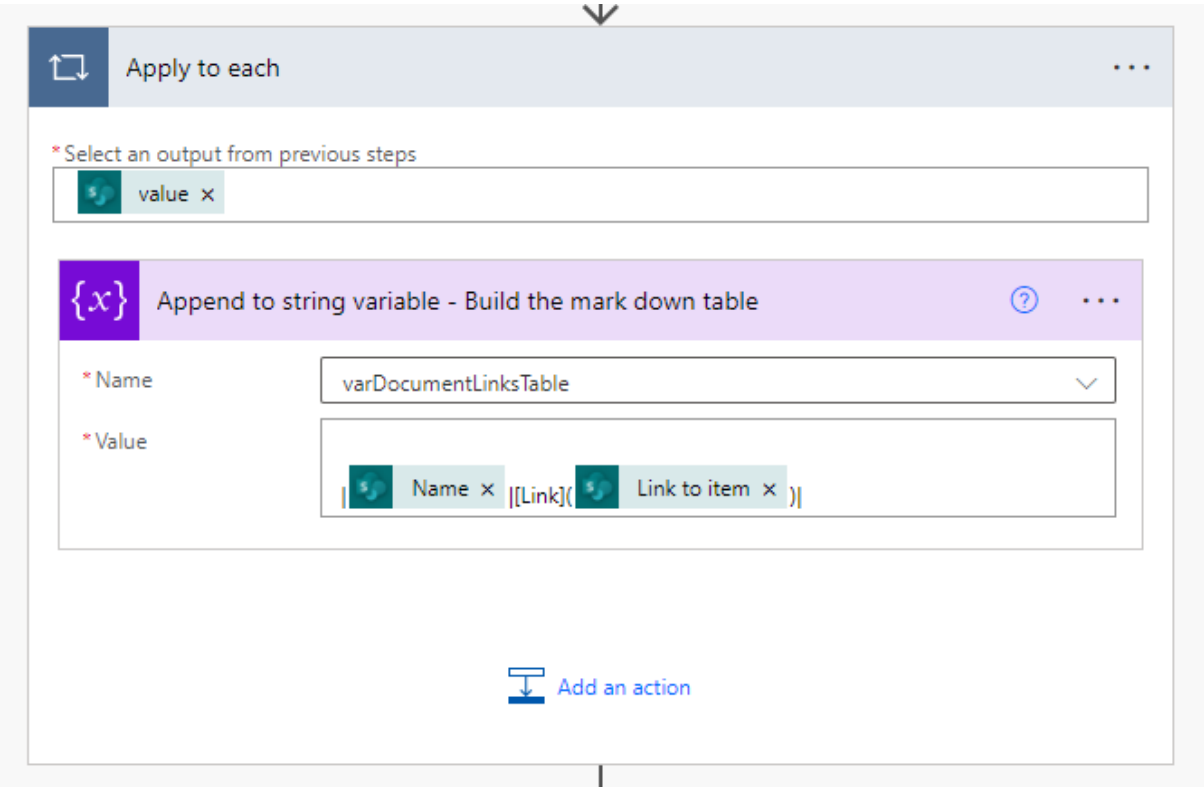
To add a table, we use three or more hyphens (---) to create each column's header and use pipes (|) to separate each column. A detailed overview of mark down language can be seen here



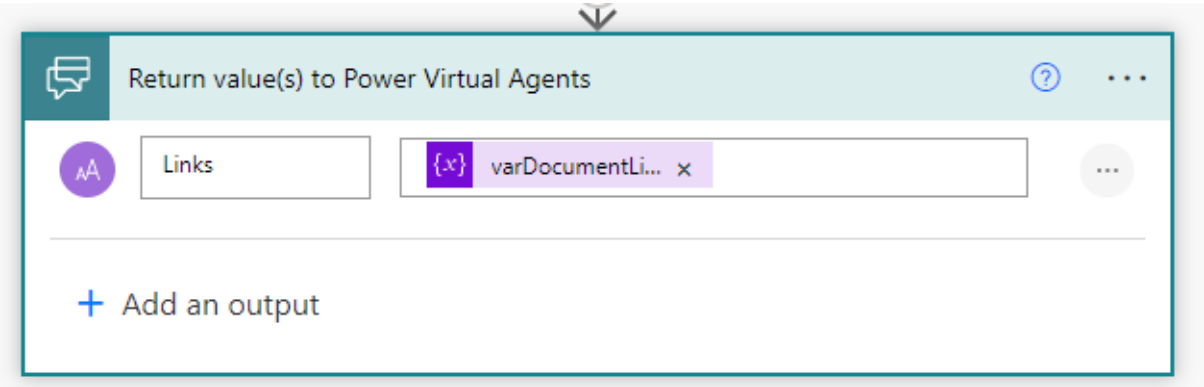
Next, we will add the Get files action which will be used to spot the files which contains the user inputted name in the file name. We will use the filter query that accepts OData syntax and we will use the syntax substringof('@{triggerBody()['text']}',FileRef) to see if the search keyword is present in the file name



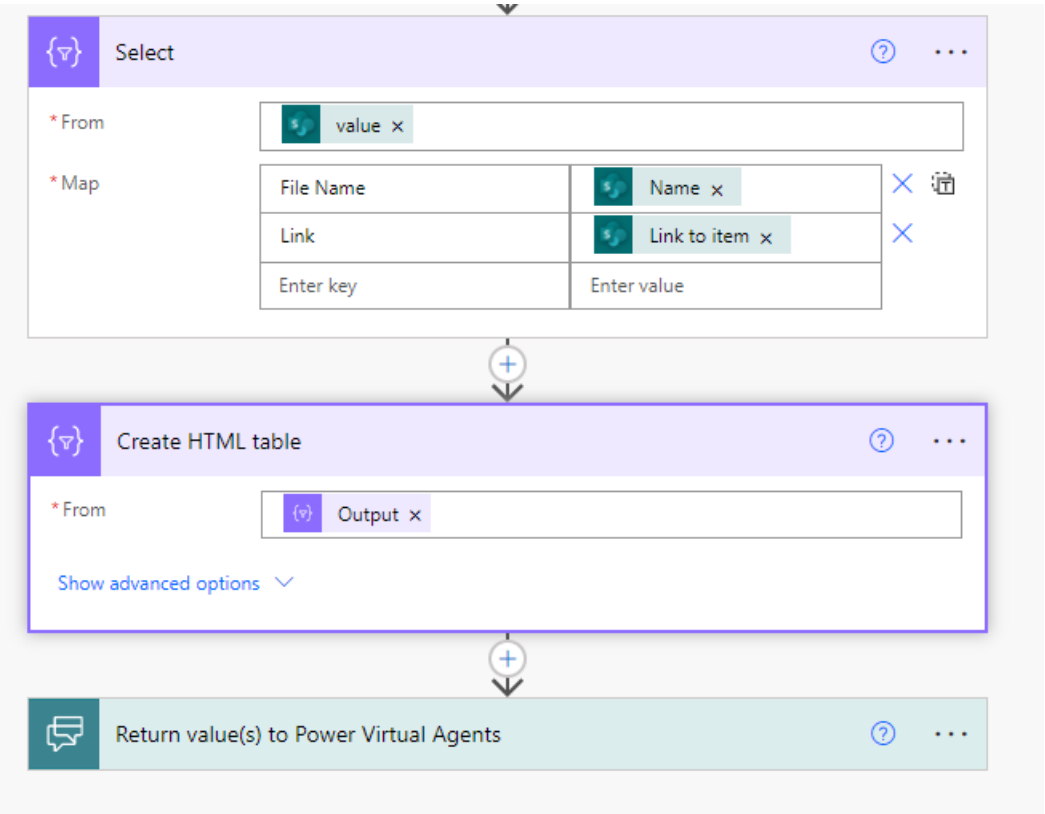
As there may be more than 1 file results returned, we will be using the Apply to each action to loop through the results to get the name and the link to file and build a mark down table by appending to the previously declared string variable. We will append the values within Pipe symbols so that they will show up as columns in the table. To display the link as a hyperlink we can use the syntax [Link](address to the location) to format the hyperlinks.



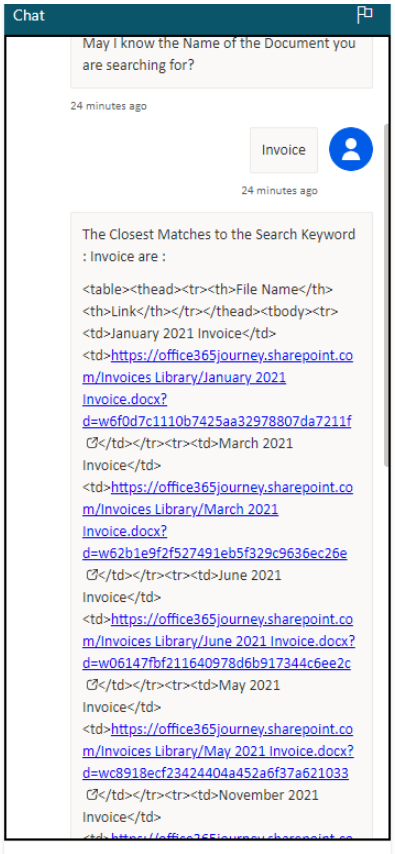
Eventually we will return the variable to PVA for displaying to the user.



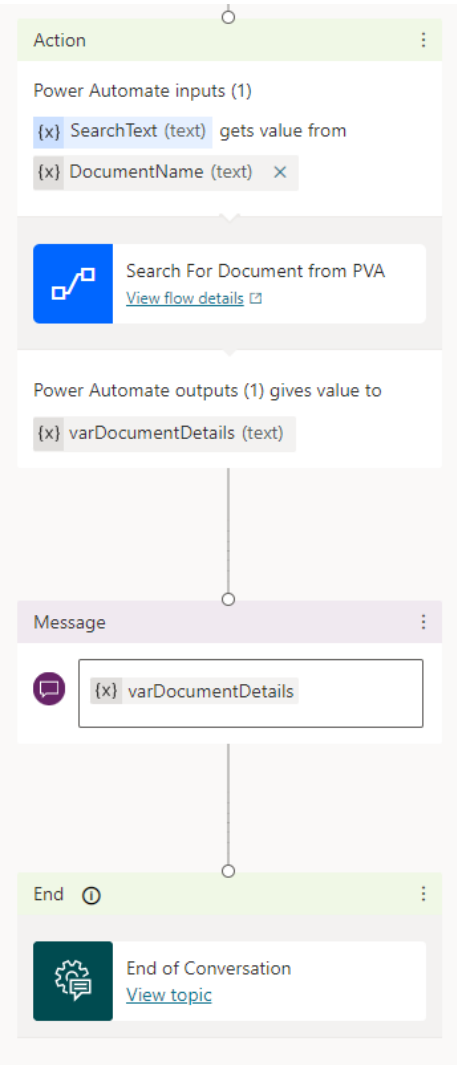
Note: PVA do not support HTML table display, hence if we decide to use the action Select followed by Create HTML table to format the output, the content will be displayed as plain text with html tags without rendering as HTML



The output would look distorted as below:

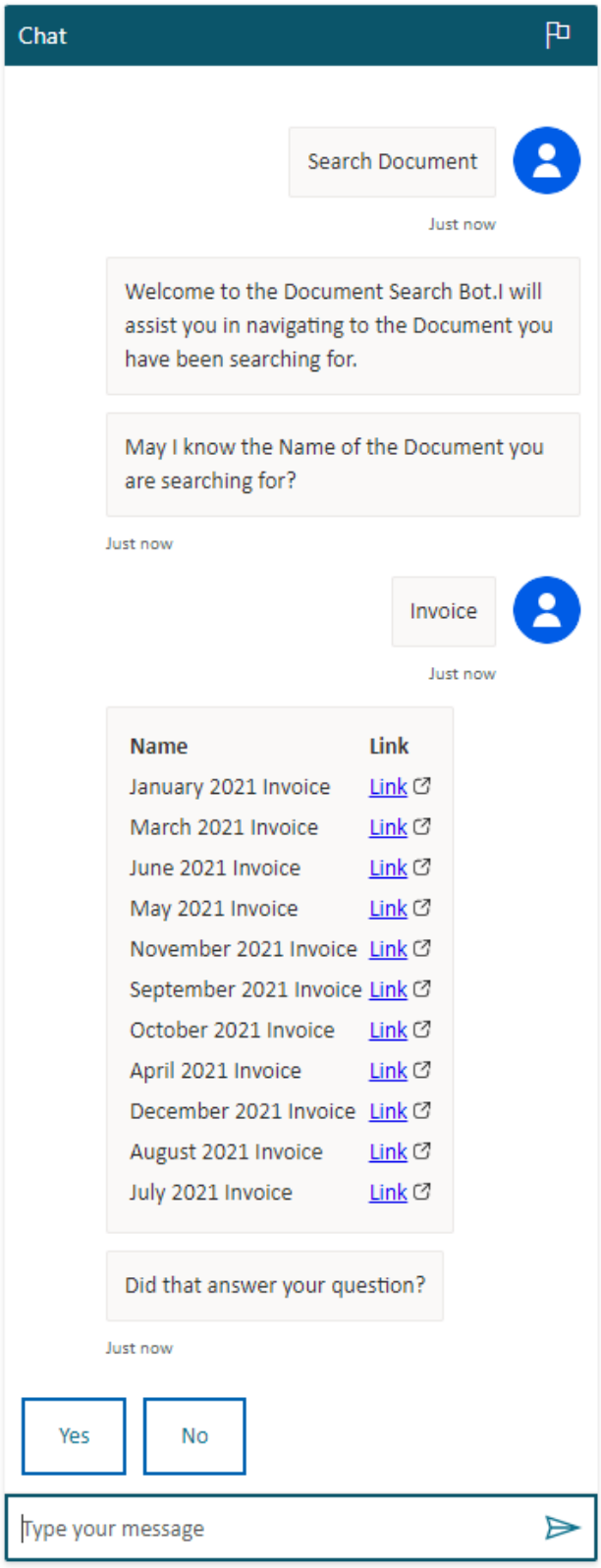


So, coming back to PVA, we will call the newly created Power Automate by passing the document name and the return output will be stored in the variable varDocumentDetails which we will show to the user using the message node.



Test the bot

Now let's test the bot using a trigger phrase – Search Document. It will trigger the conversation and ask for the document name key word. Using this input, the power automate will be called and SharePoint list is queries and the data is returned as a table using Markdown language and displayed back to the user.



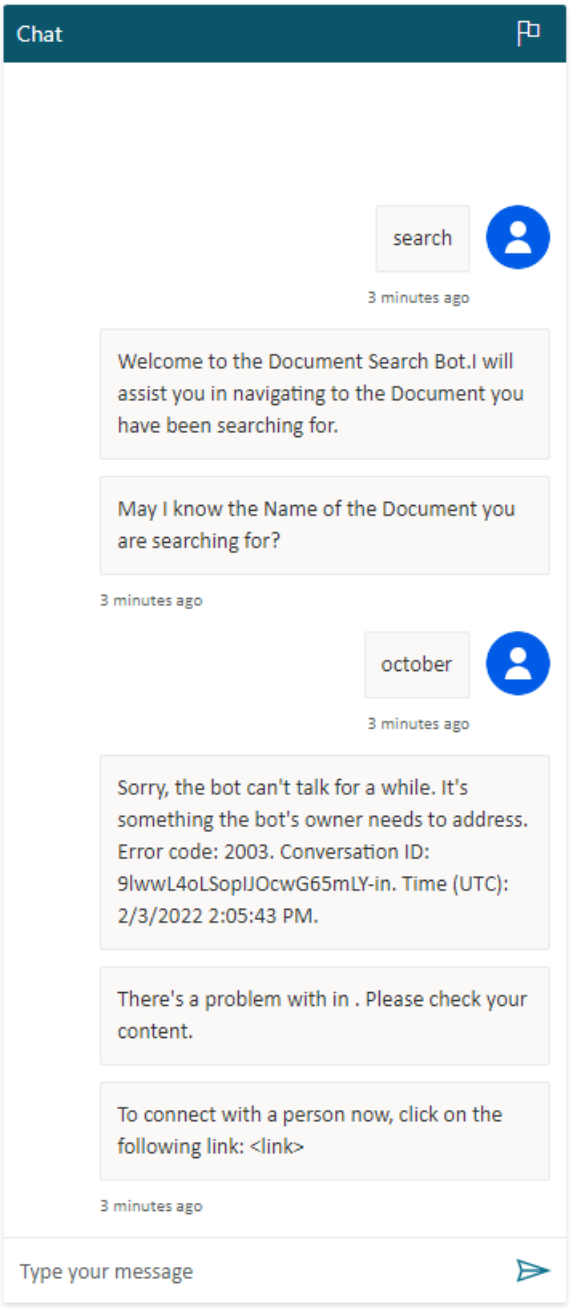
Summary

Thus, we saw how we can use Power Virtual Agents to automate document search functionality in SharePoint and we can publish this across channels like Teams which will improve the productivity of users

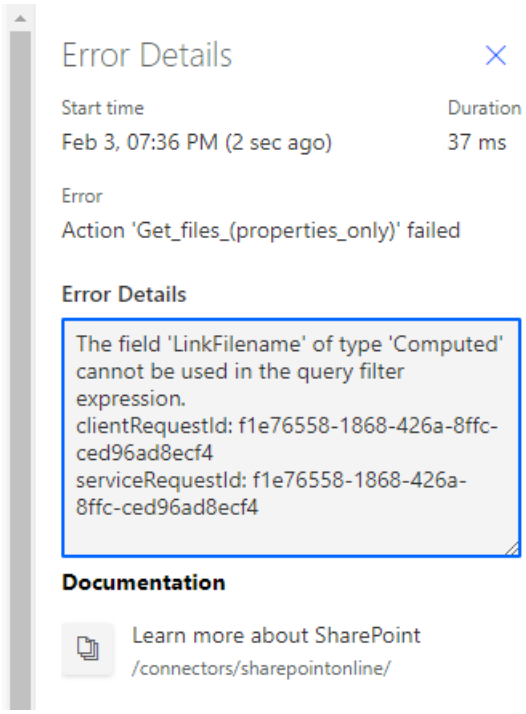
Troubleshooting Errors in Power Virtual Agents

Introduction

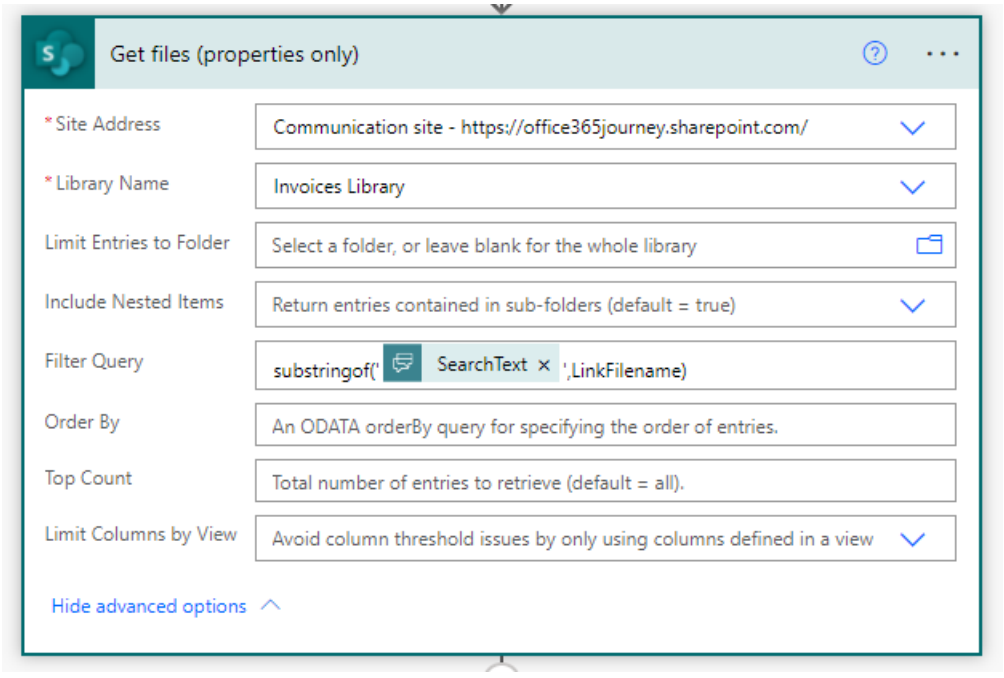
While creating a Power Virtual agent that sends a search keyword as input to a Power Automate while invoking it, the flow was erroring out resulting in an error being shown back in PVA as:



To check the error, we jumped into the power automate and the error listed there was “The LinkFileName” of type computed cannot be used in the query filter expression.

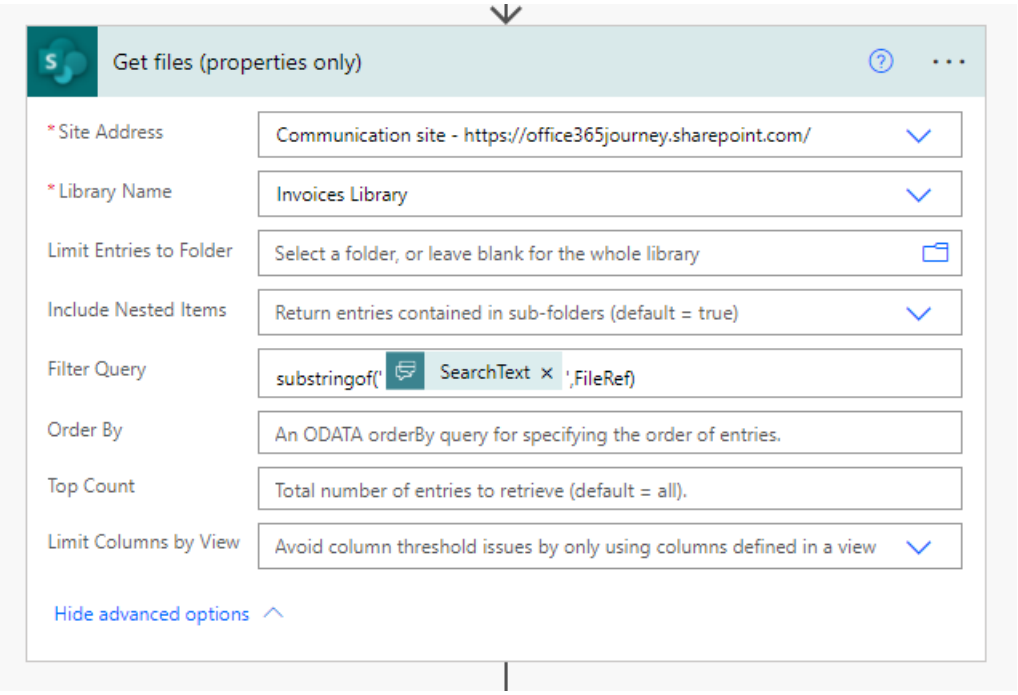


So basically, we were using the Search Keyword input from the PVA chat window and using it as a filter query to see a document existed by that name in SharePoint using the Get Files action. However, the error thrown clearly states that we cannot use LinkFileName in the filter query as it's a non-Filterable metadata field.

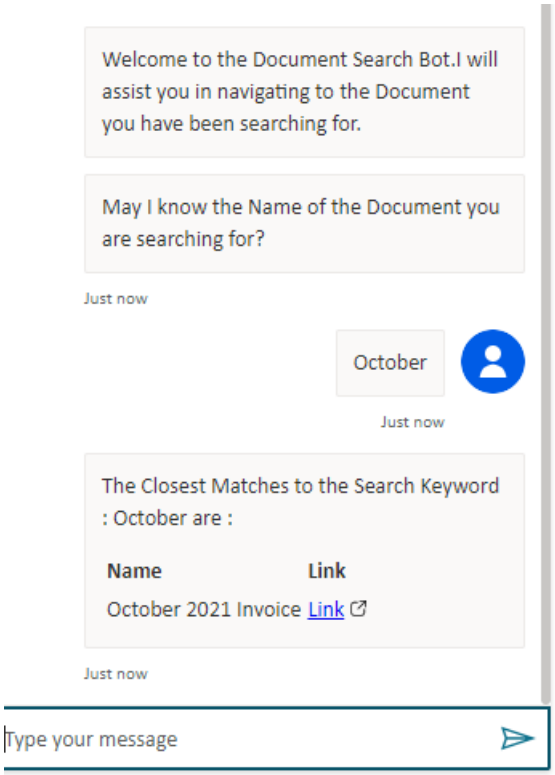


Resolution

After a bit of research, I was able to find that we still have an option to use another filterable field which hold the document name metadata which is **FileRef .** So instead of LinkFileName we replaced it with FileRef in the filter query and the error went away.



We did a search using a keyword October and the PVA has transferred this input to the Power automate which has used the Get Files action to filter the specific file name using the revised filter query and we have the result shown back in the PVA chat window



Summary

Thus, we saw how we can use FileRef field instead of LinkFileName to filter the document library by the document file name and this comes in handy when you need to search for a specific document in the library

Fetch Videos and Display them as a Response to User Query

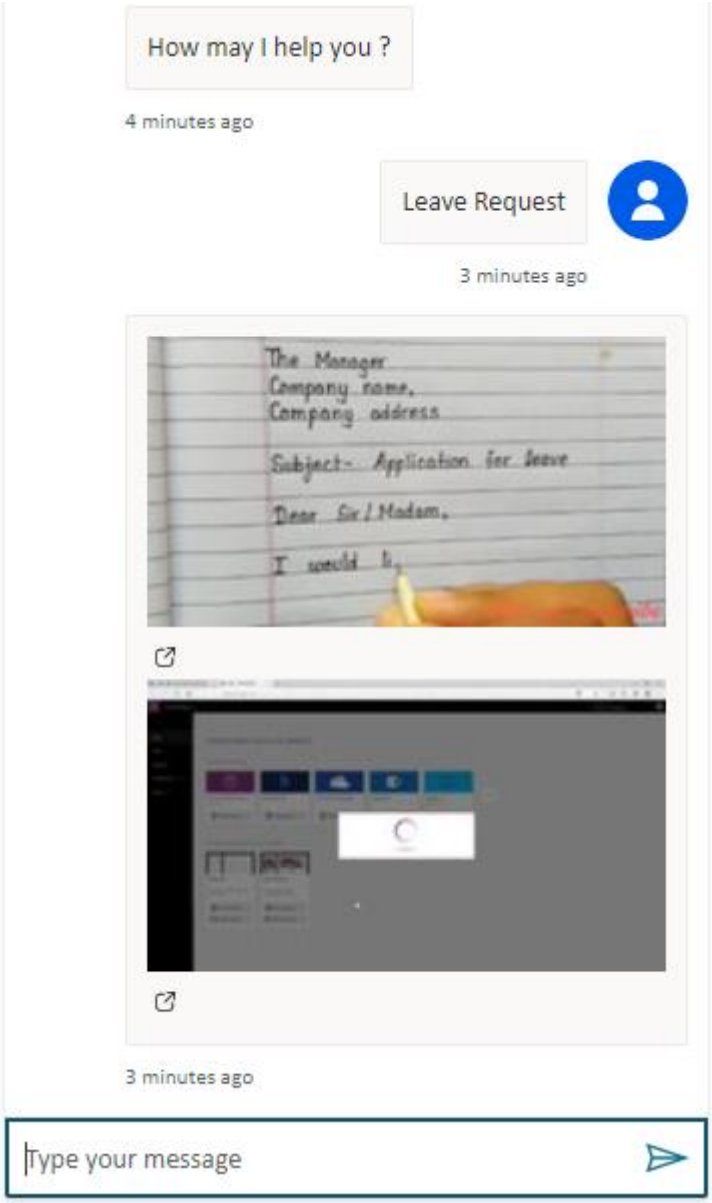
Introduction

In the previous sections, we did see how to use markdown language to work with the formatting and beautification of text that is shown in the Power Virtual Agents. We also saw the syntax for displaying Video file.

In this article, we will see a real time use case of creating a bot that displays image using markdown in the bot window.

Scenario

We have a requirement to create a Guide Me bot that will pick videos related to the user requests that shows them how to do a particular task and show it in the bot window. Clicking on it will walk the user through the detailed video.

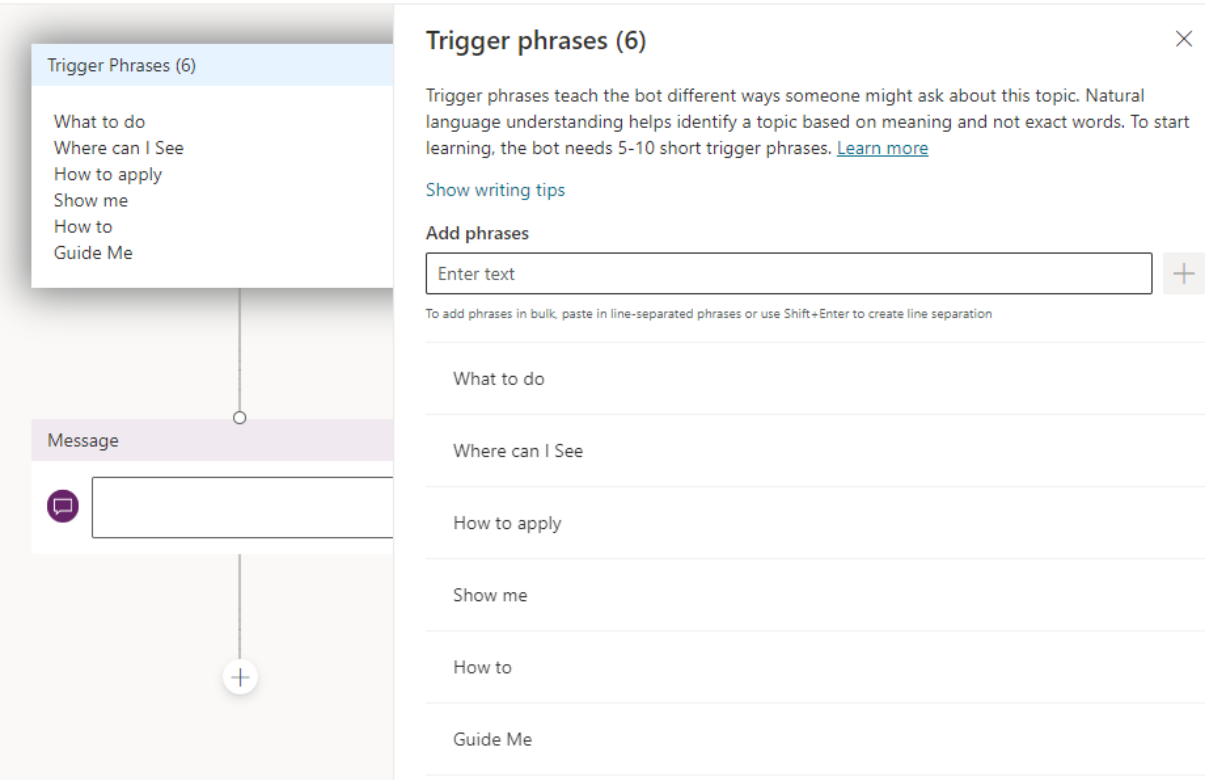


Implementation

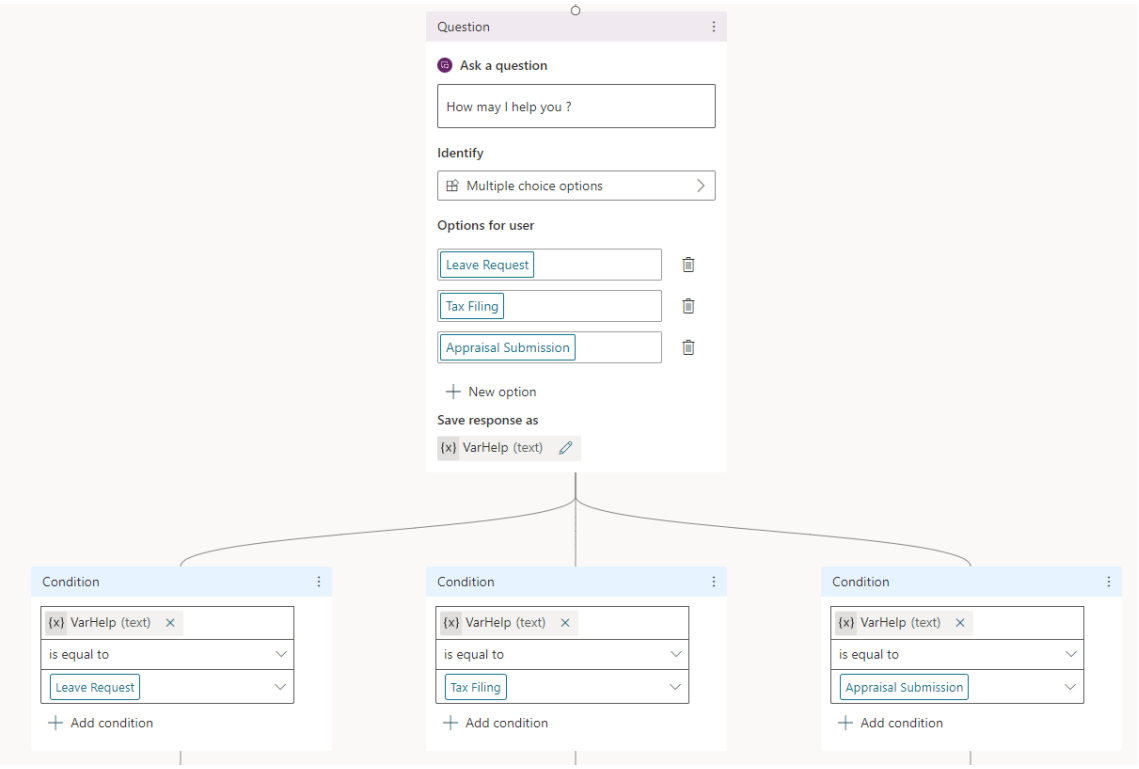
To implement this, we will create a back-end SharePoint library with the below columns that will be needed to create the markdown for the video. The Out of the box Name field will be used to store the video title

Thumbnail Alternate Image Text	Single line of text
Video Link Title	Single line of text

Let's head over to Power Virtual Agent and create a new bot. We will add few trigger words that will invoke the bot



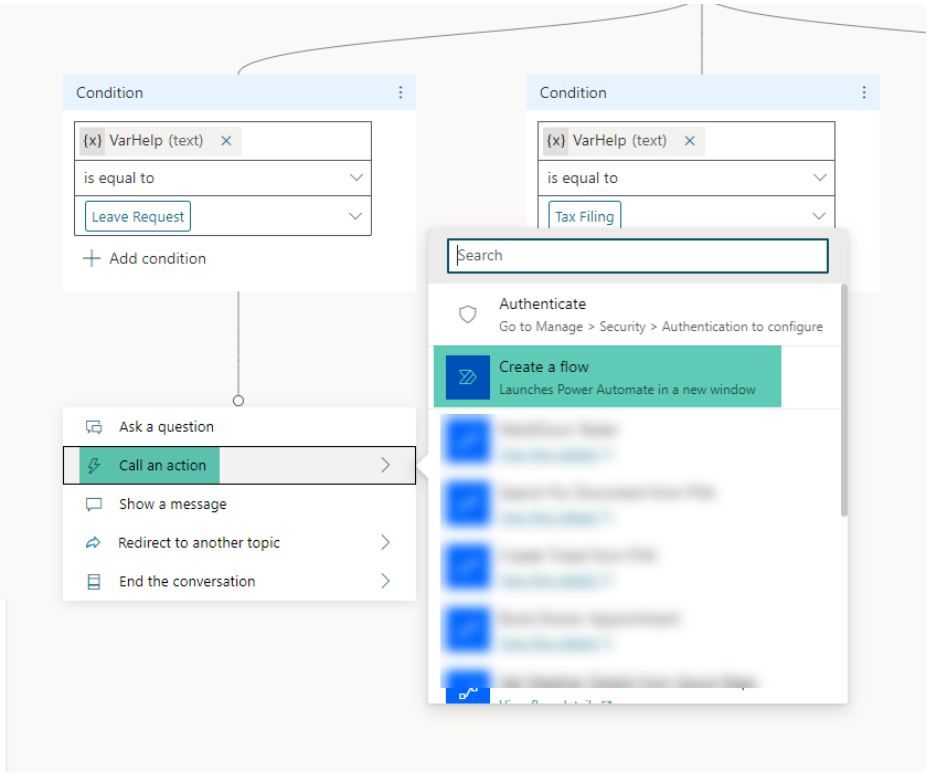
Followed by that lets ask the question on what kind of Help Videos the user would like to see.



Based on the user input, we will assign a branch to each option.

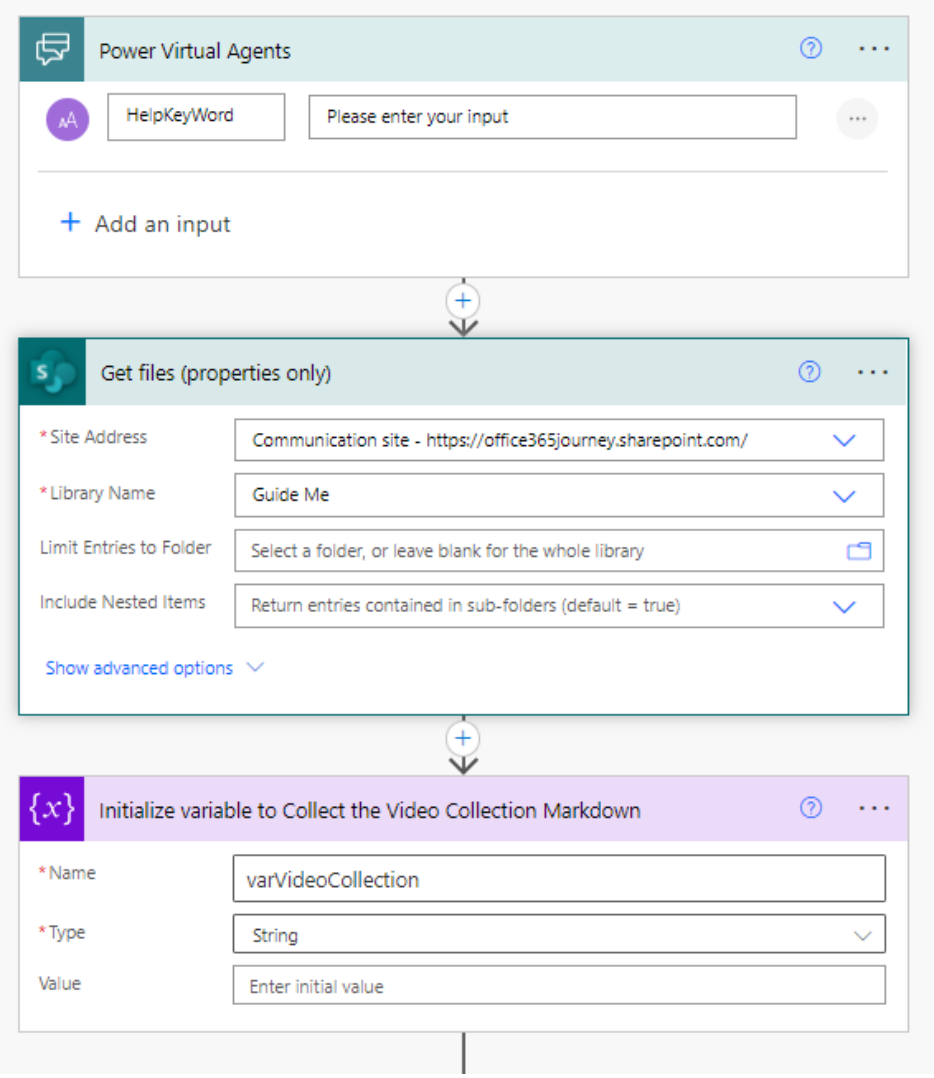
Create Power Automate

Now let's create a Power Automate by clicking on Call an action



This will open Power Automate where we can define the logic of fetching the related video and building the mark up that has to be shown back in the chat bot.

We will define an input variable that can accept the Query Search Keyword from the user. Followed by that we will fetch the Video Library Contents and define a variable which we will use to hold the markdown for the video collection.

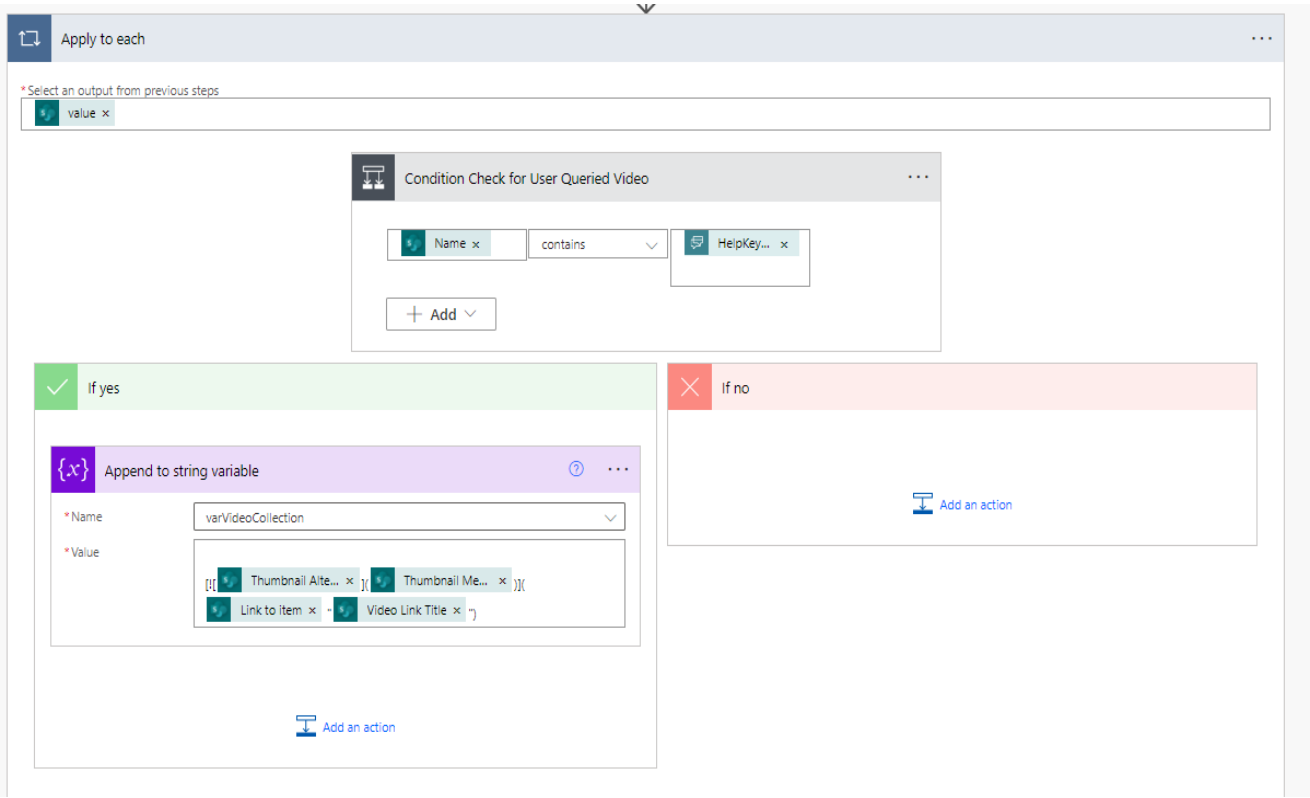


In the next steps, we will loop through the item collection and check if the Video Name contains the keyword searched by the user. And for matching records, in the Yes branch, we will append to the variable and create a mark down to show the video Link in PVA. The video link has the markdown syntax :

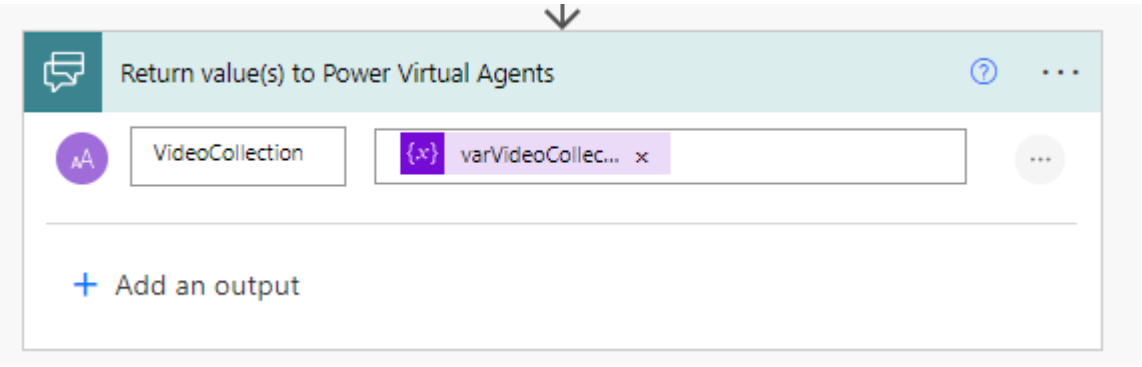
[![Alternate Text]({image-url})]({video-url} "Link Title")

We will be fetching the Alternate Image text from the SharePoint list column by the same name, the thumbnail image url is automatically fetched from the Out of the box list column – Thumbnail . Video URL is fetched from the Link To Item dynamic content and the Link Title is fetched from the custom SharePoint List Column :

[![@{items('Apply_to_each')?['ThumbnailAlternateImageText']}](@{items('Apply_to_each')?['{Thumbnail}/Medium']})](@{items('Apply_to_each')?['{Link}']} "@{items('Apply_to_each')?['Video_x0020_Link_x0020_Title']}")

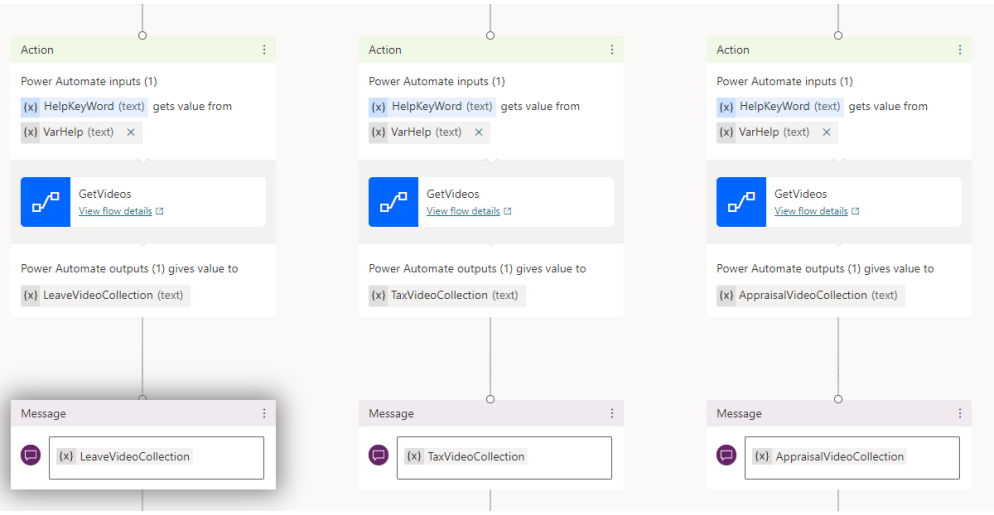


Finally let's send back the video collection markdown to PVA



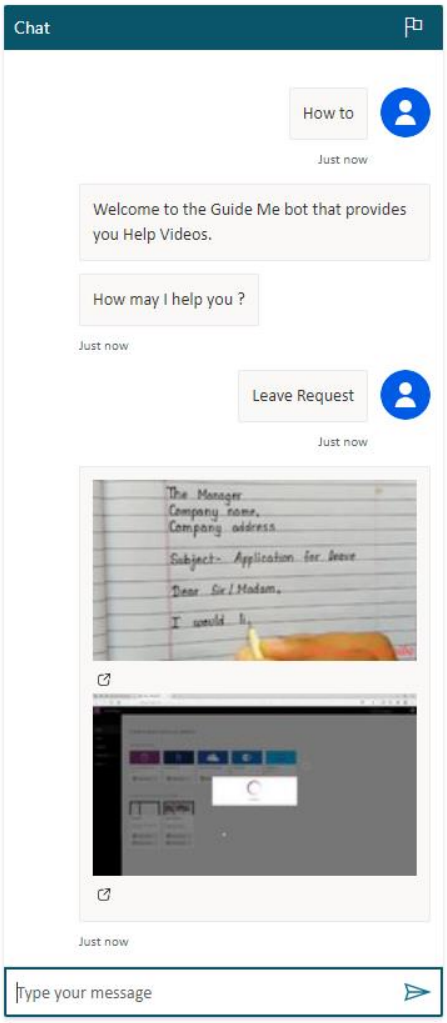
Display the Video Collection

Heading back to Power Virtual Agent, we will add the recently created flow and show the returned markdown video collection using the message node.



Test the Bot

Let's test the Bot by triggering the chat. It has fetched the leave application videos and displayed it in the chat window. Clicking on it will play the video using the web player in a new tab.



Summary

Thus, we saw how to work with markdown language and to fetch the video based on user enquiry and show the collection of videos in the chat for user viewing.

Search SharePoint List based on User Query to fetch Ticket Status Details

Introduction

With SharePoint as back end, Power Virtual Agents get a technical upgrade to store and query information with the least bit of effort. Let's see how we can use Power Automate to query SharePoint and display the information back to the user

In this section we will explore how to create a basic bot that fetches the ticket status details from the back-end SharePoint List.

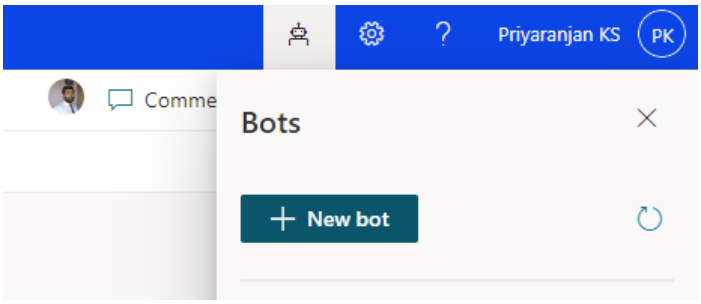
Business Use Case

As an end user or as an issue resolver, getting to know the open and closed tickets is a common requirement. Rather than contacting the person servicing it or without searching for the ticket details in the specific backend, having a Power Virtual agent that answers your ticket related query is a productivity booster.

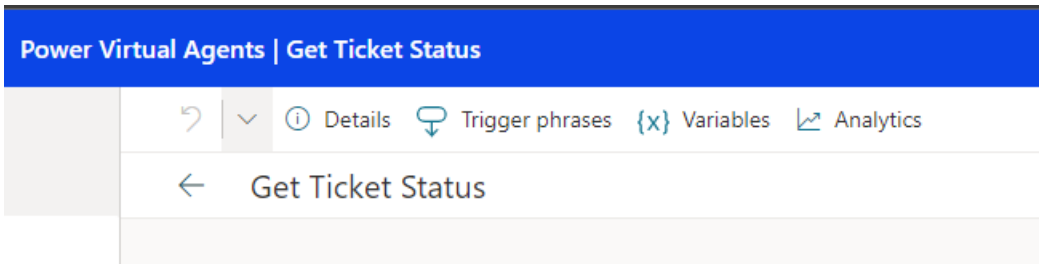
Let's try to create a bot that will listen to user questions related to Issues logged as tickets and respond back with details.

Implementation

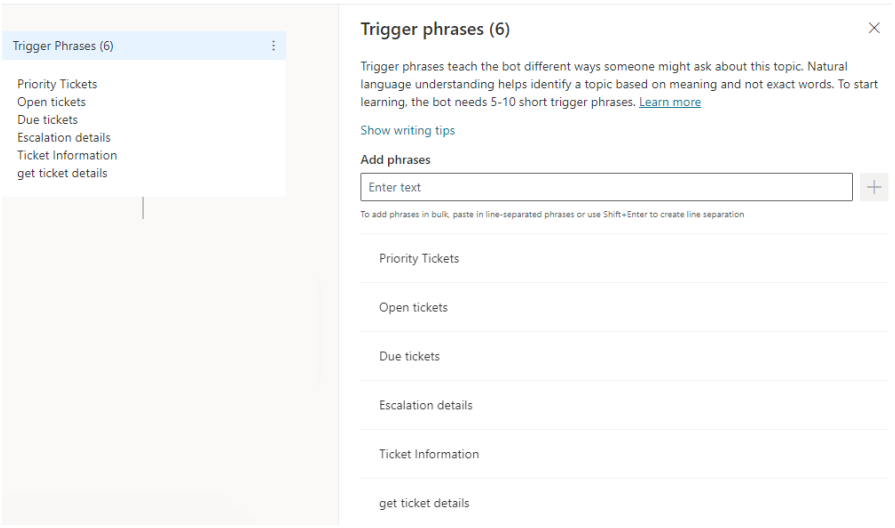
To implement the Automated Enquiry system, we will use Power Virtual Agents where we can create a bot to attend to the queries and give automated responses based on the information we have in the system. To start with we will head over to https://web.powerva.microsoft.com/ and click on the Bot symbol on the top right corner which lets you create a new bot



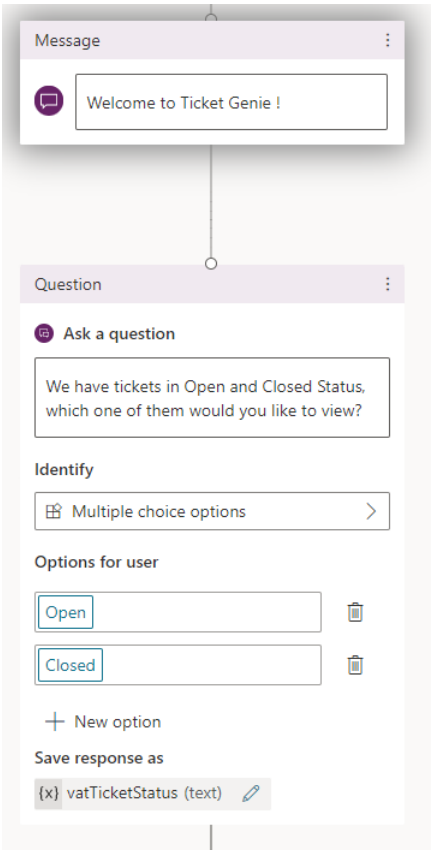
Specify the name and language for use in the bot as well as the environment where it should be provisioned. It will create a basic bot where we can add the customizations needed for our requirement. Once the bot has been provisioned, We can see the section Topics which by default lists multiple conversation topic listed out. A topic defines how a bot conversation will be initiated and how the bot would respond to user interactions. Click on New Topic and Select From blank to create a Topic. We will name the topic as Get Ticket Status.



To trigger the topic, so that the control flows to that specific topic, we will define few words called Trigger Phrases that will transfer the control to these topics. The trigger phrases can be single keywords or a group of words and to cover a broad spectrum of possible trigger conditions, It is good to mention 5-10 different and still related phrases. Click on Trigger phrases that will open a right pane where we can add the trigger words.

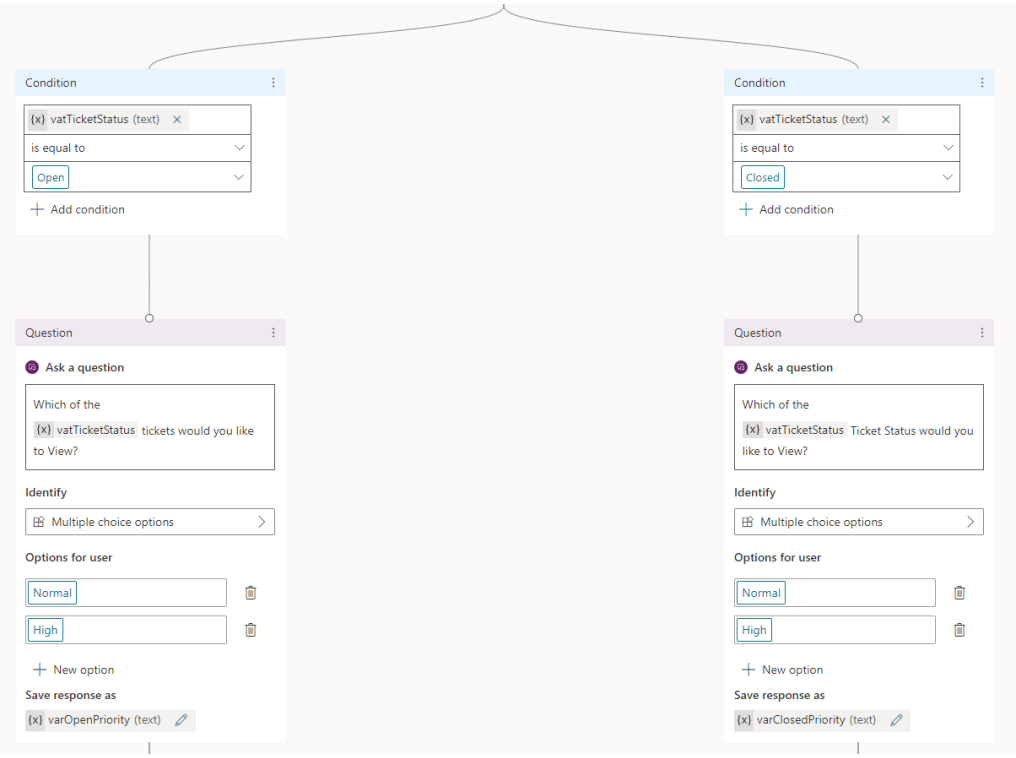


Let's add a message box with the welcome message and add the node add a question to ask for more details from the user related to which tickets he would like to view and give the options Open and Closed. The response will be stored in the variable varTicketStatus

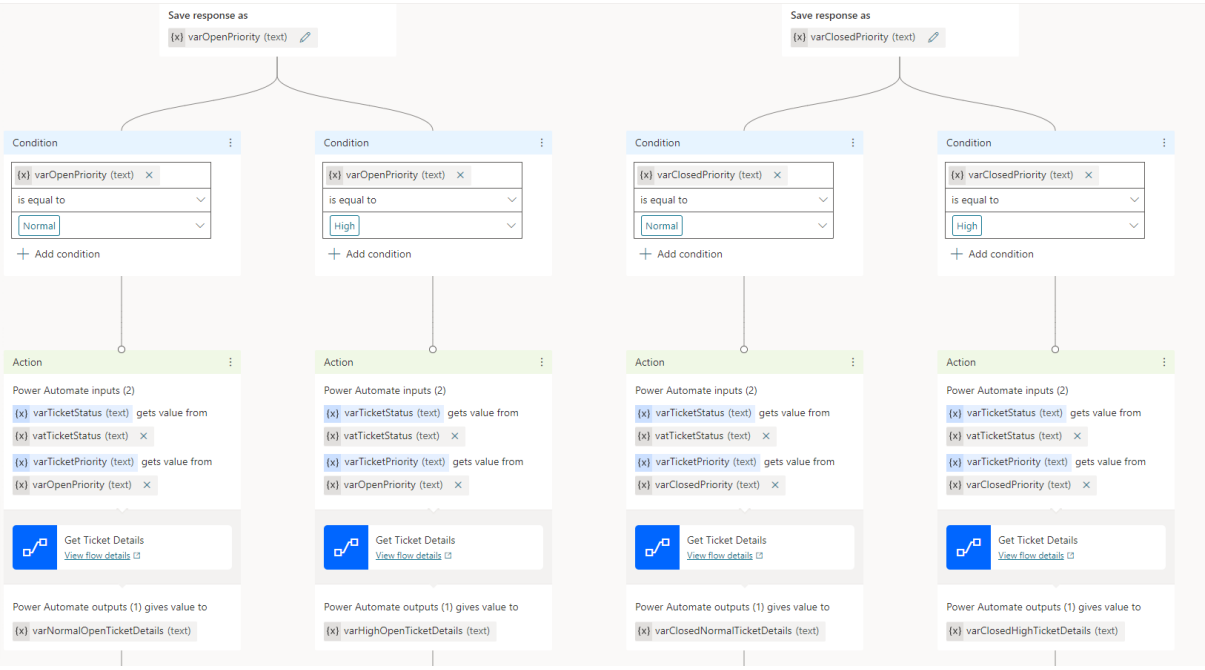


Based on the value chosen by the user, PVA will automatically generate 2 branches for both Open and Closed ticket status. We will further add another question node in both the branches to

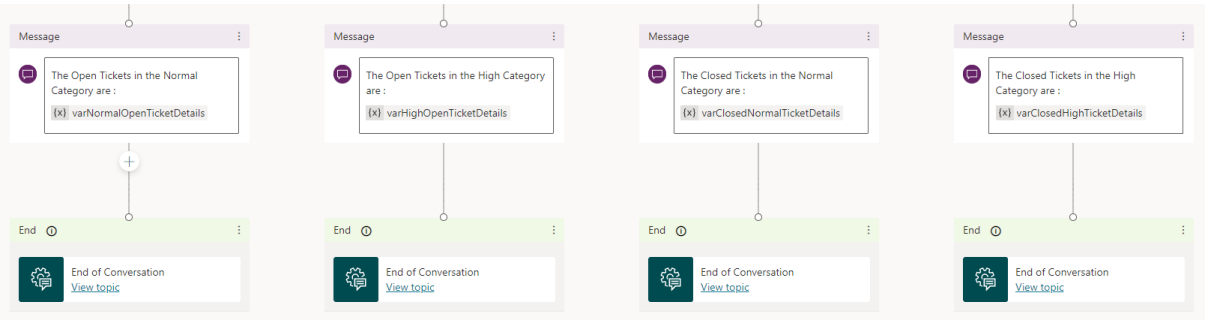
understand whether we need to fetch High/Normal Priority tickets. The value of the selection will be stored in varOpenPriority and varClosedPriority variables in respective branches.



Based on the value of the varOpenPriorty and varClosedPriority variables, PVA will created 4 branches. In each of these branches we will add the call an action node and select the “Create a flow” option to create the Power Automate in the new window to which we will pass the ticket status and the ticket priority values. Once the Power automate is created, we will again add the call an action node to select the recently created Power Automate in each of the branches as below.

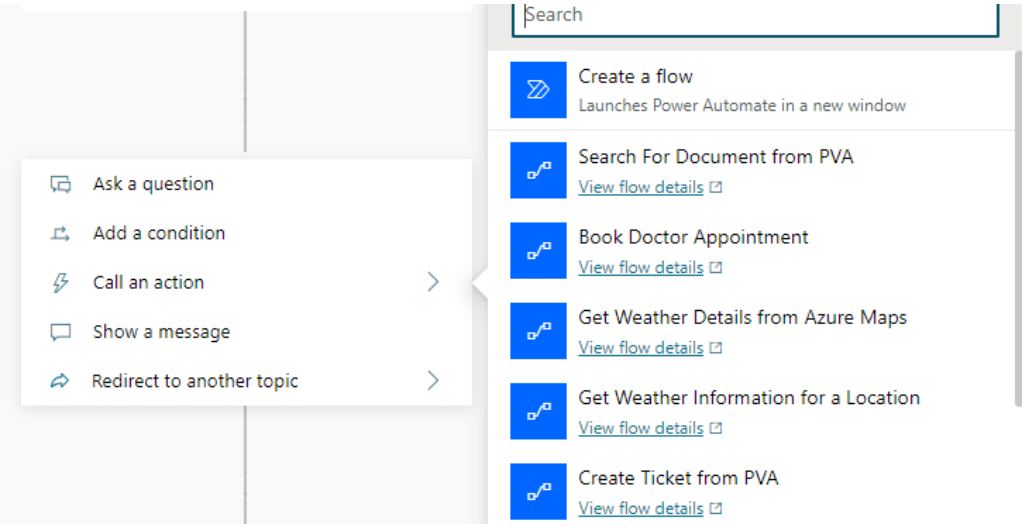


Finally, we will get back the return value from Power Automate in the respective output variables in each branch and show that in a message box and end the conversation with a survey. If we intend to add more logic, we can extend the conversation with more question actions.

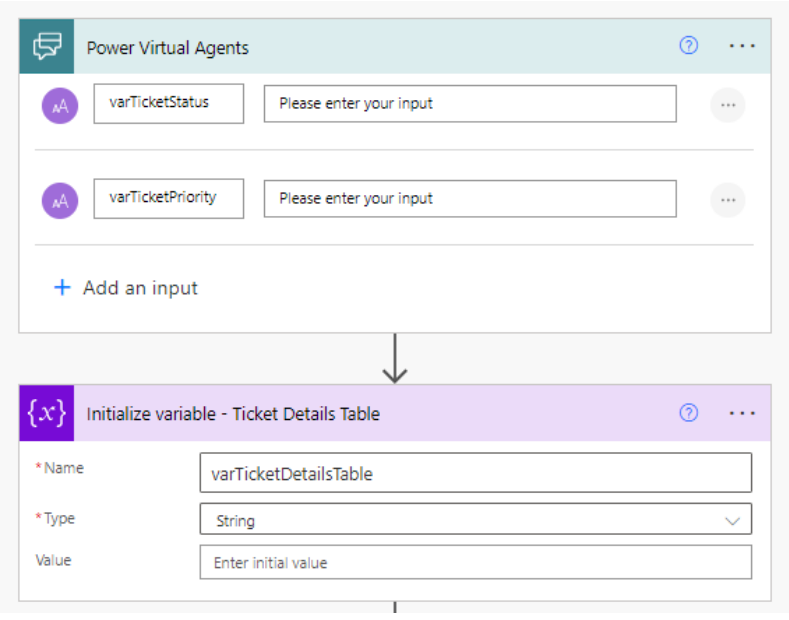


Setting up the flow

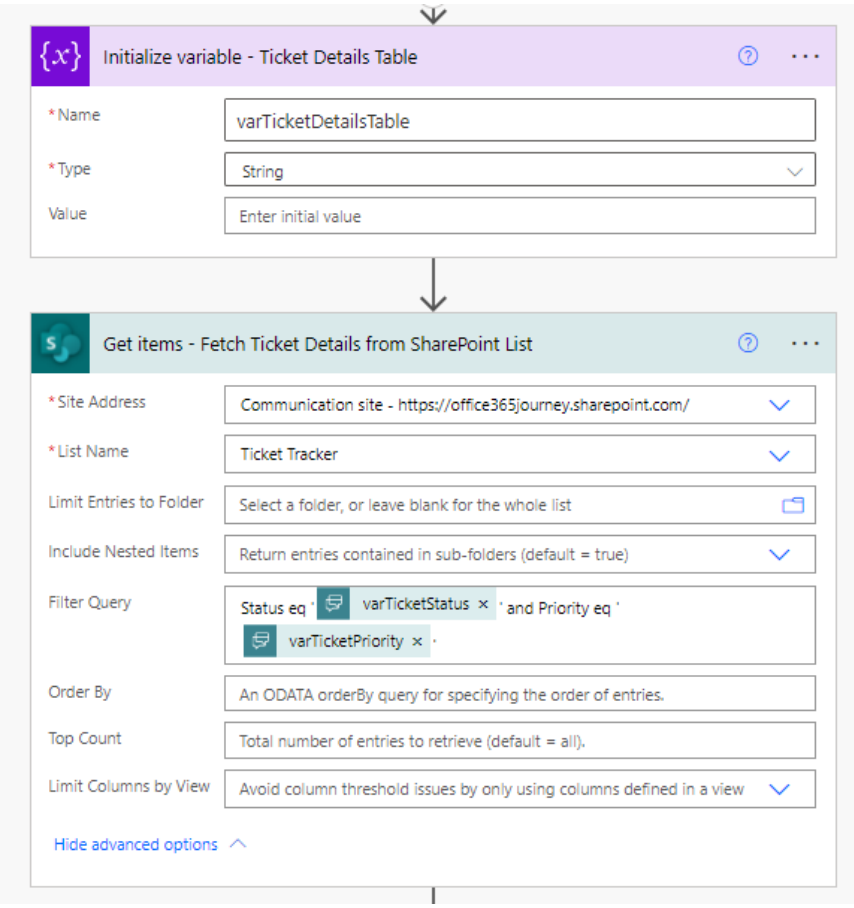
When we click on Call an action, it will list the create a flow action clicking on which the Power Automate will open in the new window.



We will add 2 text inputs to hold the values of ticket status and ticket priority passed from PVA



Let’s initialize a variable - varTicketDetailsTable to hold the ticket details. Followed by that, we will add the get items action which will pick the items that match the ticket status and ticket priority mentioned in the filter query



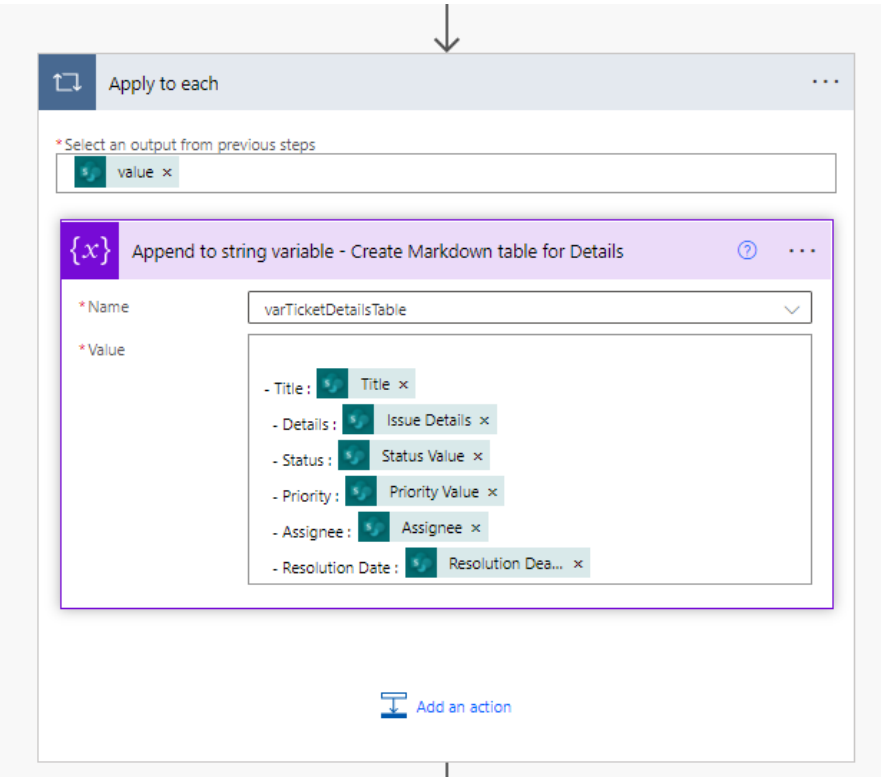
Followed by this, we will add an apply to each action which will loop through the returned items. As PVA do not support html tables, we will use markdown language to create a bulleted list of items to show in the chat window. To create an unordered list, add dashes (-), asterisks (*), or plus signs (+) in front of line items. Indent one or more items to create a nested list.In this sample we will be using the below mark down syntax :

```
- First item
- Second item
- Third item
    - Indented item
    - Indented item
- Fourth item
```

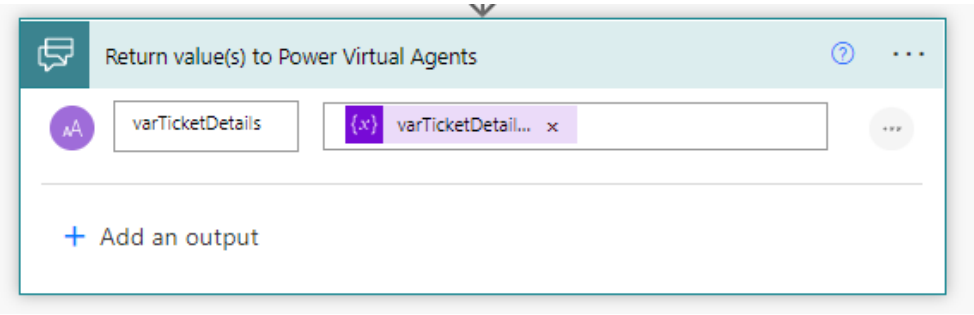
Which will generate the output as :

- First item
- Second item
- Third item
    - Indented item
    - Indented item
- Fourth item

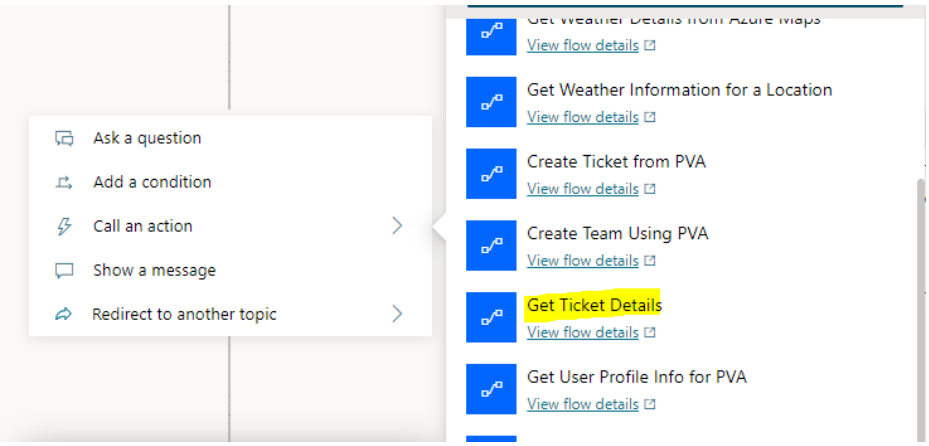
So, in the value box, we will add the hyphen markdown followed by the column name and the corresponding value.



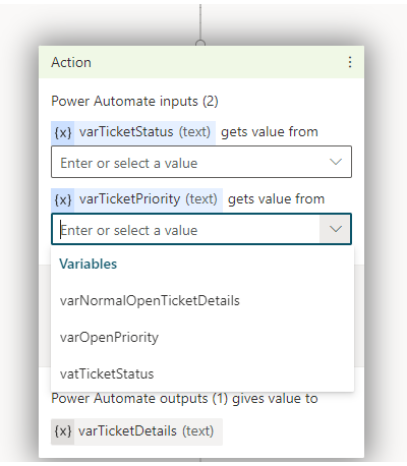
Let's return back the variable to the Power Virtual Agents.



In the Power Virtual Agents, we can call this flow by clicking on call an action and selecting the Power Automate by the name.

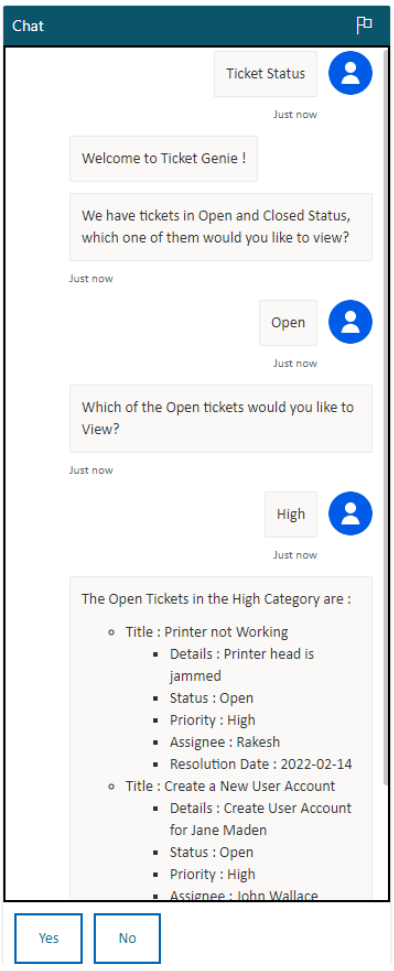


Pass the values into the corresponding variables declared in the Power Automate



Test the bot

Let's test the bot by adding a trigger word – Ticket Status. Thus it has invoked the PVA and used both the inputs entered by the user to pick the corresponding ticket details from the back end.



Summary

Thus we saw how we can use Power Virtual Agents to automate the enquiry to a ticket system to get details about the Open/Closed, Normal/High Priority tickets without the involvement of an intermediary resource.

Integration API and Leverage Azure Maps with Power Virtual Agents

Introduction

The Power of Power Virtual Agents can be extended by integrating it with third party APIs that brings in information from the outside world to the chat bot. We can easily integrate the APIs by creating a connection and establishing trust with that API.

In this article we will see how to create a basic Weather Bot that picks weather details from Microsoft Azure Maps API.

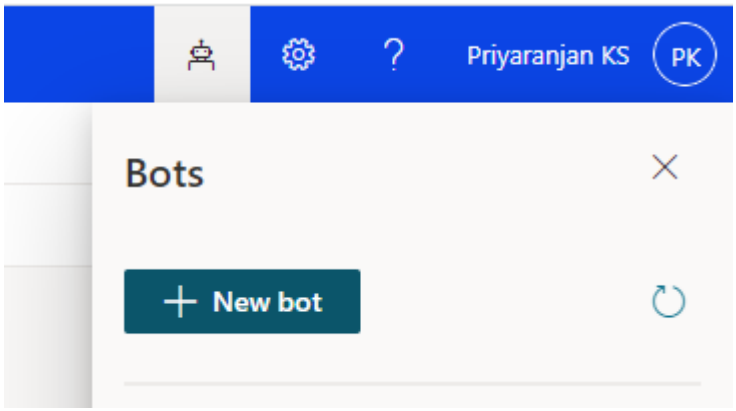
Business Use Case

We have a requirement where we need to have an automated weather enquiry system that will pick the weather information for a specific zip code location.

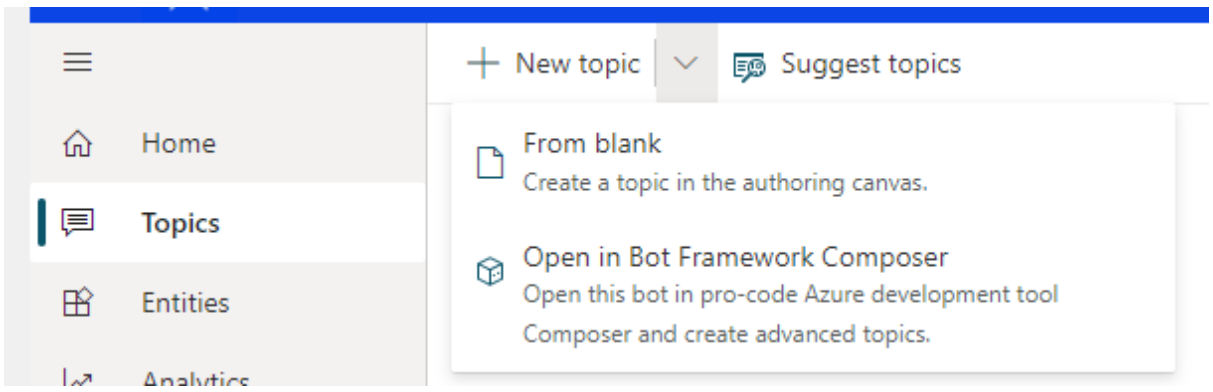
To achieve this, we will use Azure Maps which provides developers with powerful geospatial capabilities. It provides various endpoints to fetch data related to Maps, Search, Routing, Traffic, Weather, Time Zones, Geolocation. The official documentation can be found here.

Implementation

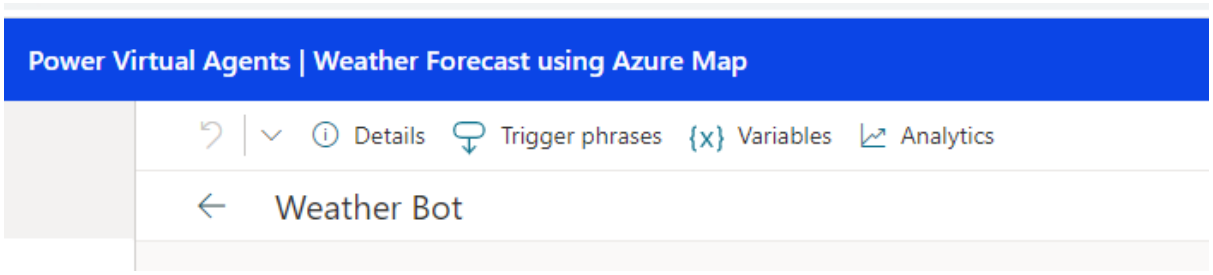
To create a bot that consumes Azure Maps services, we will go ahead to https://web.powerva.microsoft.com/ and create a bot by selecting the top right option



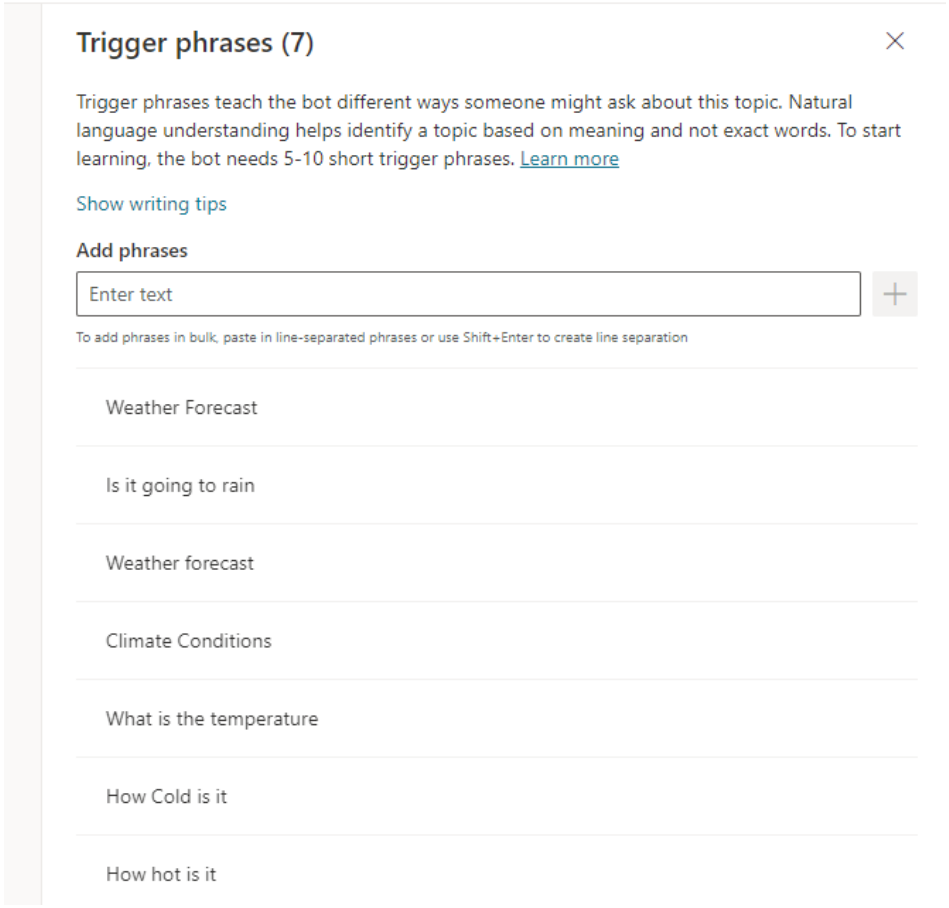
Which will provide us with the window to name the Bot and the language to be used. Once the bot is created, we will create a blank topic inside the bot that will define the way the conversation will be triggered and driven with the end user.



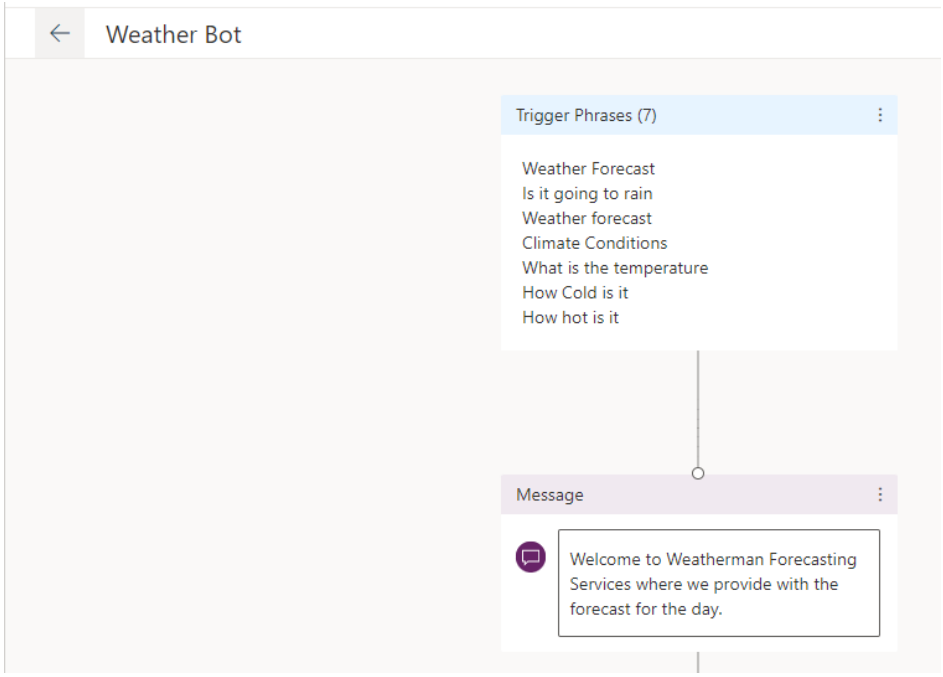
We have named the bot as Weather Bot and to define the phrases that will initiate the conversation, click on trigger phrases.



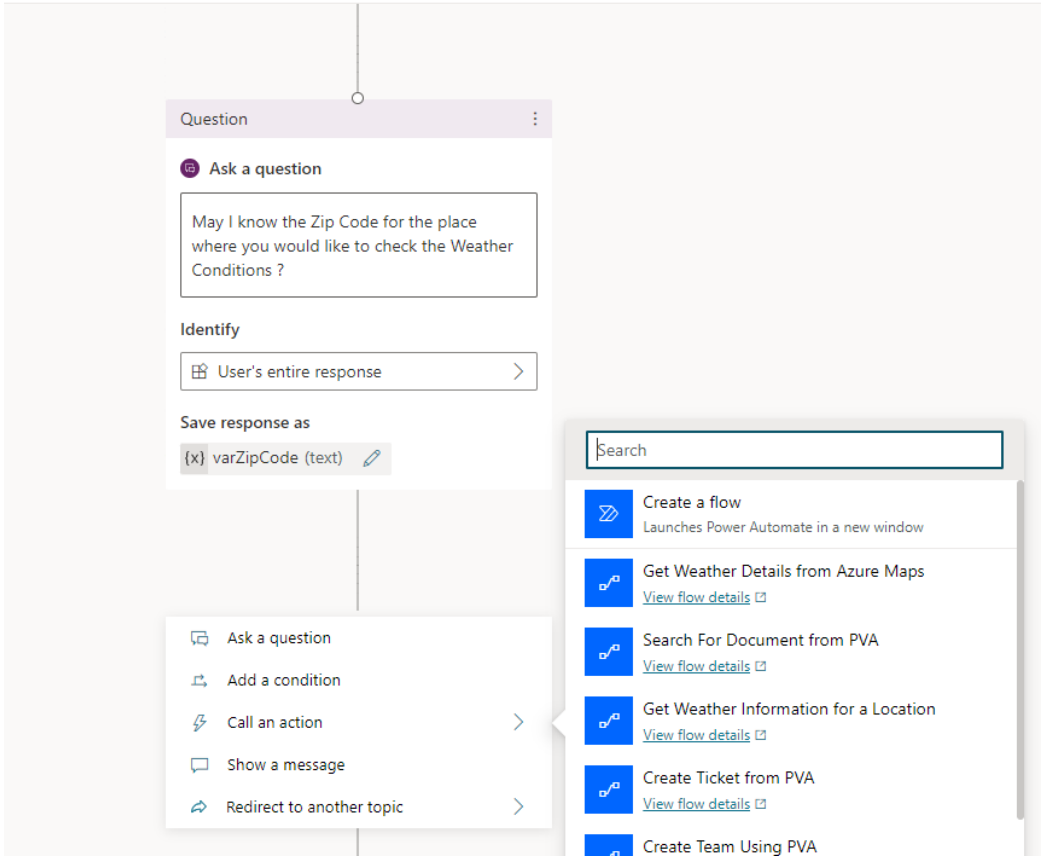
The trigger phrases will help in defining the words and phrases that will start the conversation with the bot. We have defined 7 such phrases and the more different and related words we add to it, better the chances of bot understanding the trigger condition.



Let’s add a welcome message to the Authoring Canvas

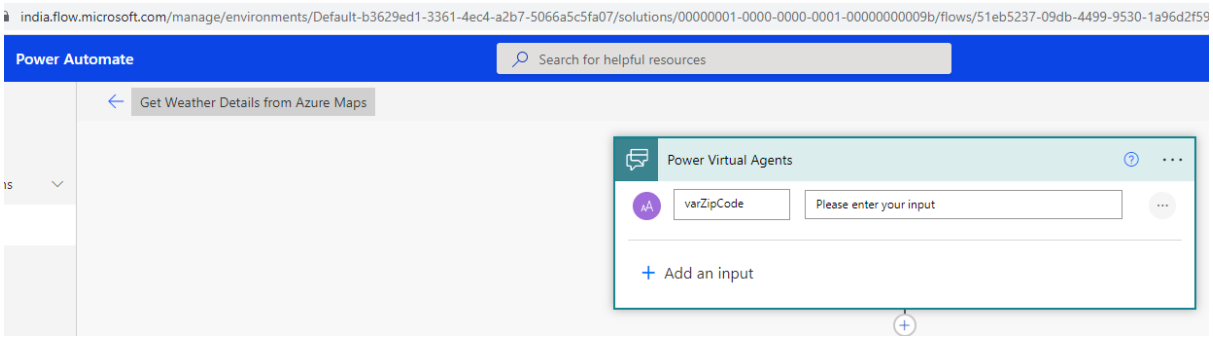


Followed by this we will request more information about the zip code at which the user needs to know the weather conditions using Ask a Question conversational node. We will save the data in the varZipcode variable. Let's create a Power Automate to which we will pass the zip code as parameter to involve azure maps weather end point. Click on Create a flow which will open Power Automate in the adjacent window where we can define the flow logic.



Create the Power Automate

We will add the input parameter to accept the zipcode from Power virtual agent calling node

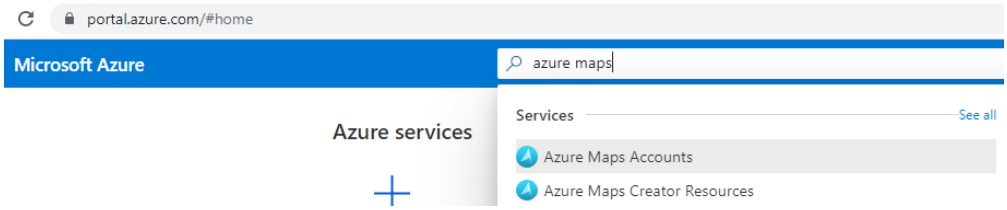


To use the weather api, we will need the latitude and longitude of the Zip Code provided by the user. To get the latitude and longitude details, we will initially use the Address API of azure maps. The API URL is https://atlas.microsoft.com/search/address/json

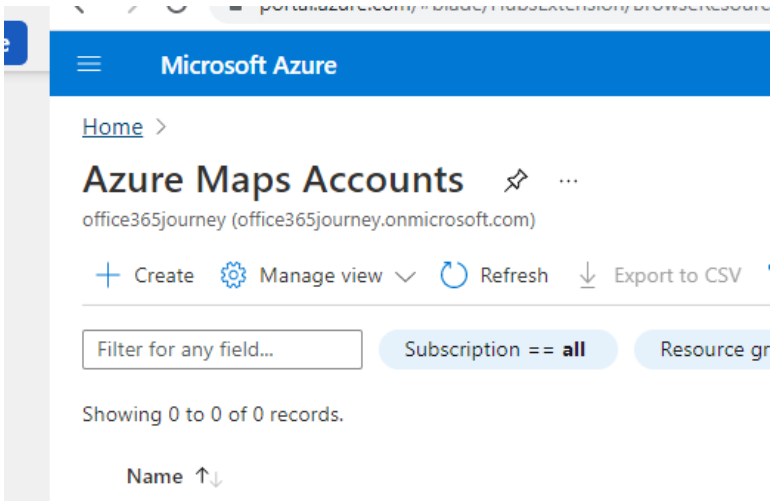
It takes the input query parameters :

query	Zip code variable
api-version	1
subscription-key	<Key obtained from the Azure Maps instance registration>

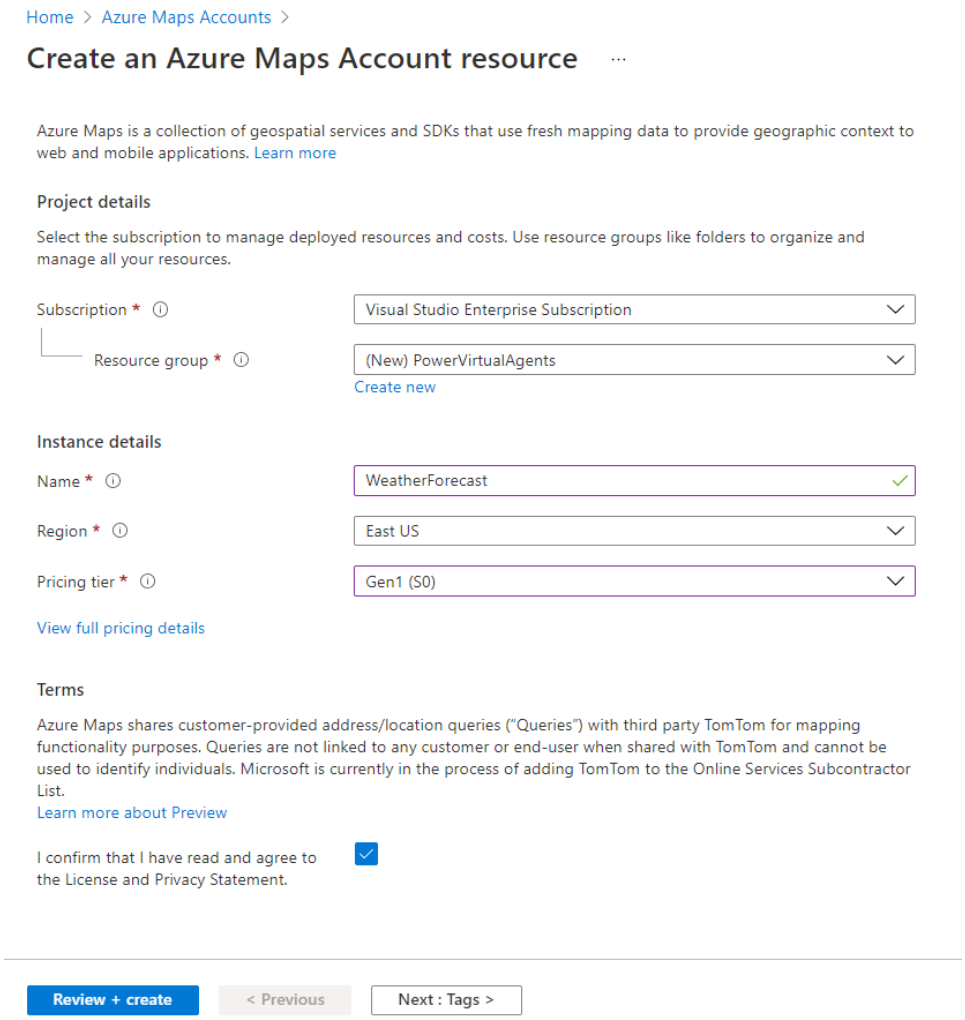
Before proceeding, lets create an instance of azure maps account by heading over to azure. Search for Azure Maps in the top search bar which will list Azure maps in the drop down and select it.



Click on create to provision a new azure maps account.



Provide the details and click on create to provision the azure maps account.



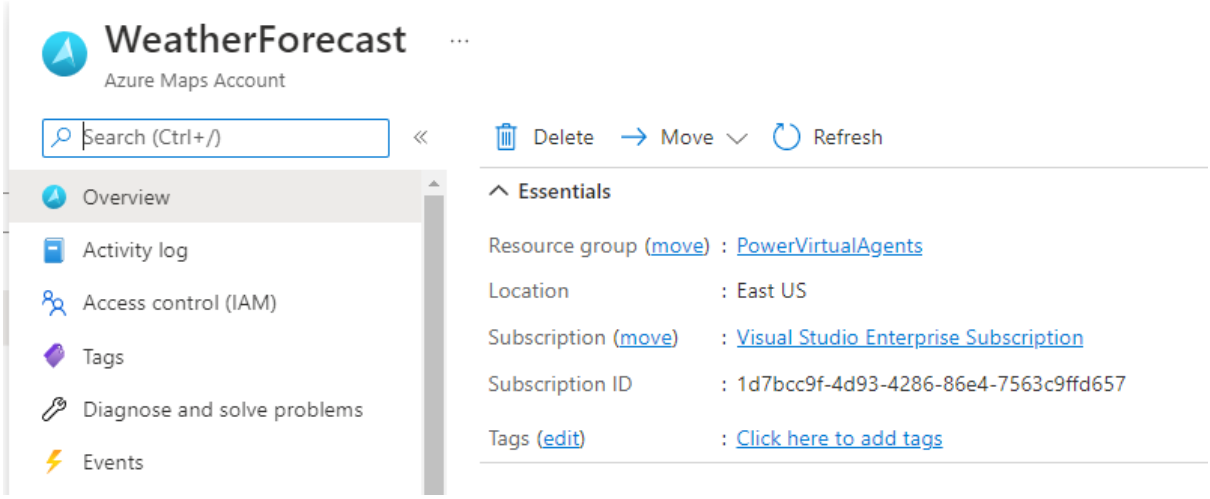
We have selected the Gen1 S0 pricing tier as we are doing a dev Proof of Concept.

Pricing tier targeted customers

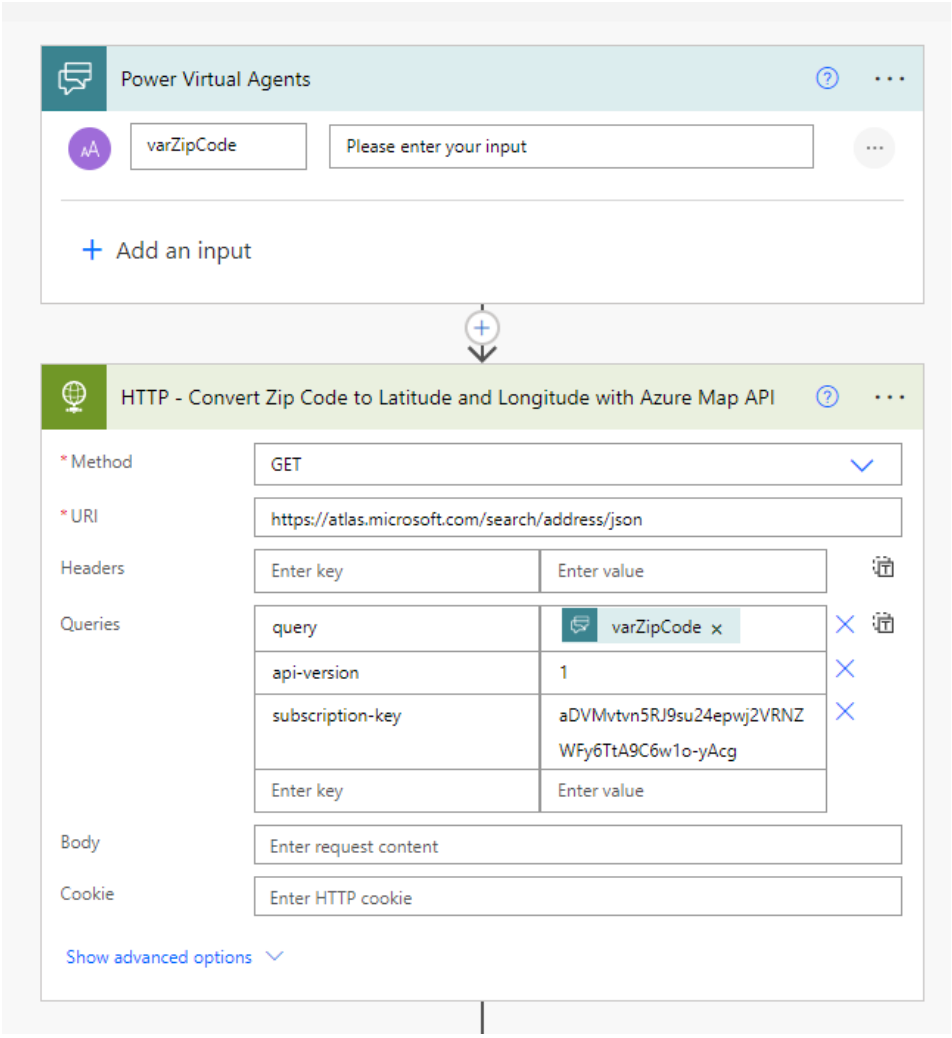
See the **pricing tier targeted customers** table below for a better understanding of Gen 1 and Gen 2 pricing tiers. For more information, see Azure Maps pricing. If you're a current Azure Maps customer, you can learn how to change from Gen 1 to Gen 2 pricing in the Manage pricing tier article.

Pricing tier	SKU	Targeted Customers
Gen 1	S0	The S0 pricing tier works for applications in all stages of production: from proof-of-concept development and early stage testing to application production and deployment. However, this tier is designed for small-scale development, or customers with low concurrent users, or both. S0 has a restriction of 50 QPS for all services combined.
	S1	The S1 pricing tier is for customers with large-scale enterprise applications, mission-critical applications, or high volumes of concurrent users. It's also for those customers who require advanced geospatial services.
Gen 2	Maps/Location Insights	Gen 2 pricing is for new and current Azure Maps customers. Gen 2 comes with a free monthly tier of transactions to be used to test and build on Azure maps. Maps and Location Insights SKU's contain all of Azure Maps capabilities. It allows you to achieve cost savings as Azure Maps transactions increases. Additionally, it has higher QPS limits than Gen 1. The Gen 2 pricing tier is required when using Creator for indoor maps.

After the successful creation of the Azure Maps account. head over to the overview section and fetch the subscription id which we will be using in the Power Automate.



Heading back to Power Automate, lets add an HTTP call to the Address Azure Maps API to fetch the equivalent latitude and longitude of the zip code.

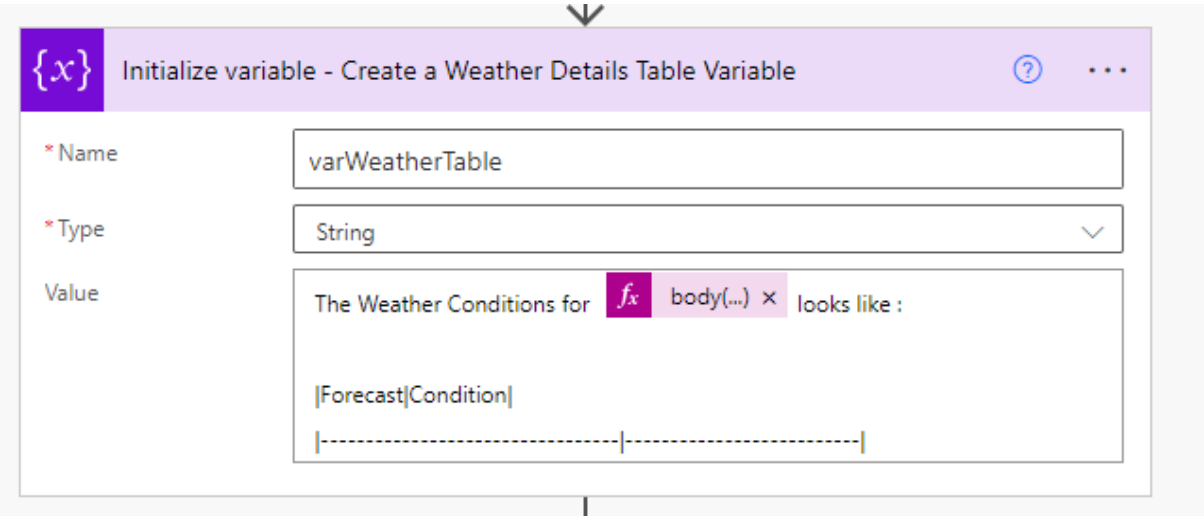


So as to display the weather details in the Power Virtual Agent as a table, we will be using mark down language and we will initialize a variable that will hold the header which is defined using the syntax:

|Forecast|Condition|

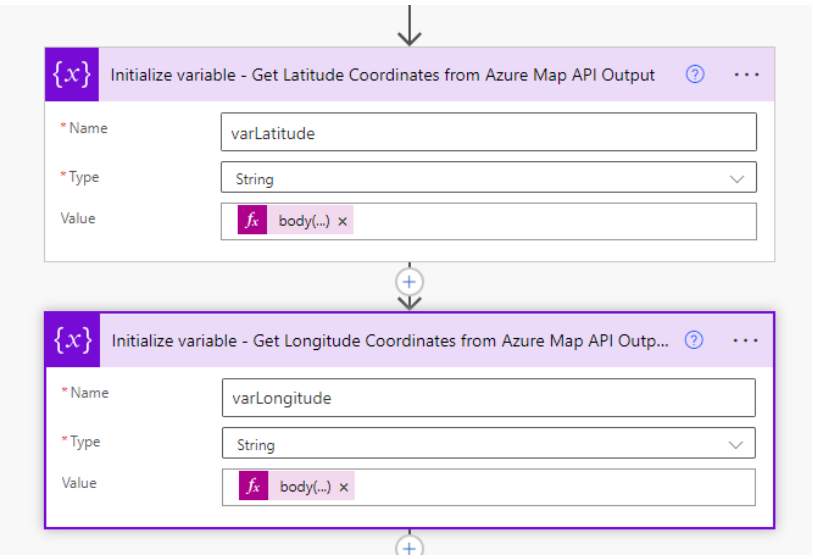
|---------- |----------|

To add a table, we use three or more hyphens (---) to create each column's header and use pipes (|) to separate each column. A detailed overview of mark down language can be seen here



We will now declare 2 variables to host the Latitude and Longitude values with the below expressions

Latitude Expression	body('HTTP_-_Convert_Zip_Code_to_Latitude_and_Longitude_with_Azure_Map_API')['results'][0]['position']['lat']
Longitude Expression	body('HTTP_-_Convert_Zip_Code_to_Latitude_and_Longitude_with_Azure_Map_API')['results'][0]['position']['lon']

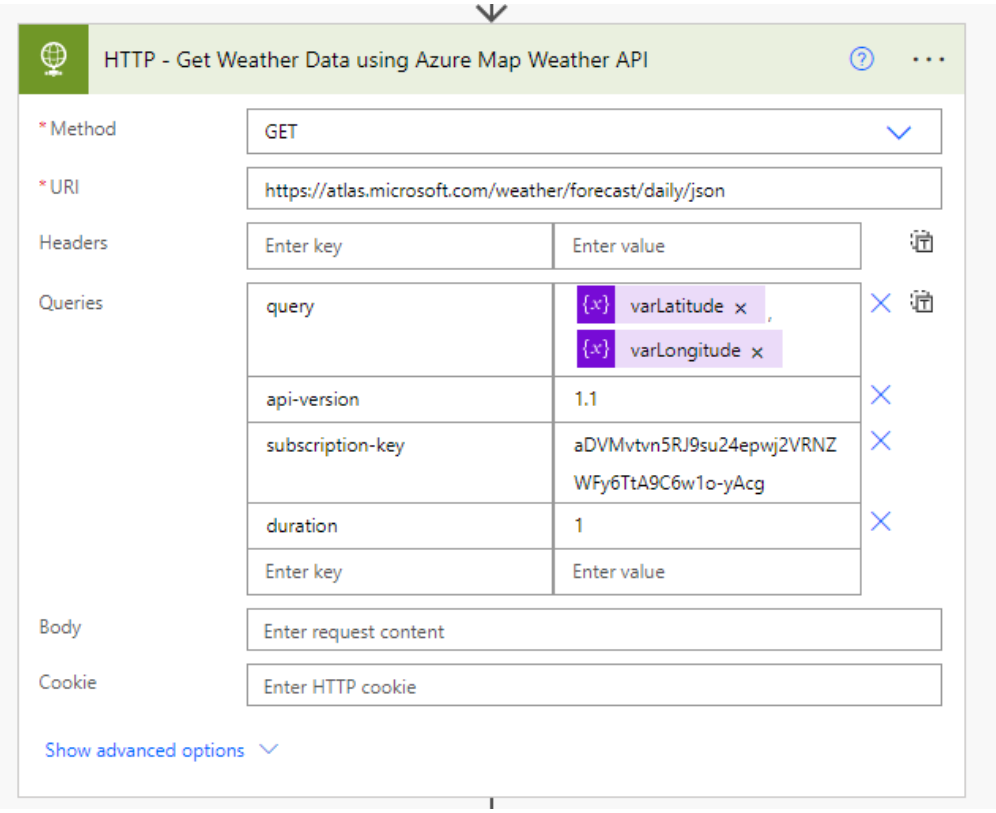


Now lets add the Azure Maps Weather API that gets the daily forecast details by issuing a Get Request to the URL : https://atlas.microsoft.com/weather/forecast/daily/json

We will be using the below query parameters :

query	<latitude variable>,<longitude variable>
Api-version	1.1
Subscription-key	<Azure Maps Account Subscription Key>
duration	1(In S0 Pricing tier, We can request daily forecast for the next 1, 5, 10, and 15 days)

The detail of the API is listed here



The output the of the HTTP call will be a JSON which we will append to the weather table variable in the mark down format by separating each column values with a Pipe Symbol. The expressions used to spot the required weather data from the returned JSON is described below :

|Summary|@{body('HTTP_-_Get_Weather_Data_using_Azure_Map_Weather_API')['summary']['phrase']}|

|Min Temperature|@{body('HTTP_-_Get_Weather_Data_using_Azure_Map_Weather_API')['forecasts'][0]['temperature']['minimum']['value']} Celcius|

|Max Temperature|@{body('HTTP_-_Get_Weather_Data_using_Azure_Map_Weather_API')['forecasts'][0]['temperature']['maximum']['value']} Celcius|

|Hours of Sun|@{body('HTTP_-_Get_Weather_Data_using_Azure_Map_Weather_API')['forecasts'][0]['hoursOfSun']}|

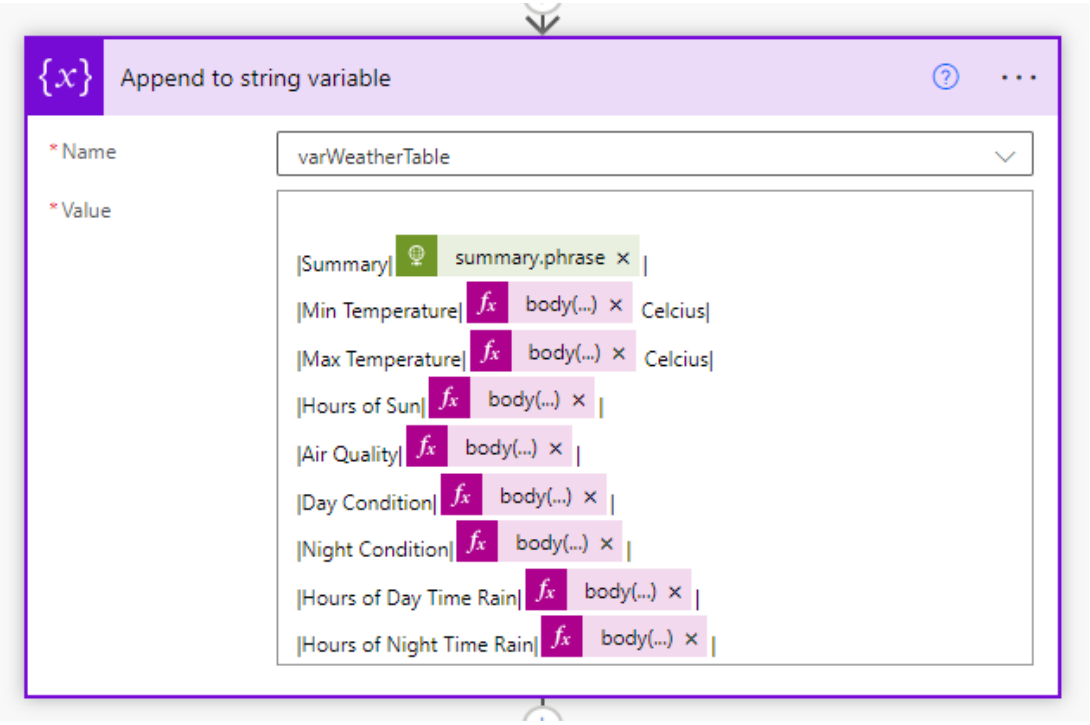
|Air Quality|@{body('HTTP_-_Get_Weather_Data_using_Azure_Map_Weather_API')['forecasts'][0]['airAndPollen'][0]['category']}|

|Day Condition|@{body('HTTP_-_Get_Weather_Data_using_Azure_Map_Weather_API')['forecasts'][0]['day']['shortPhrase']}|

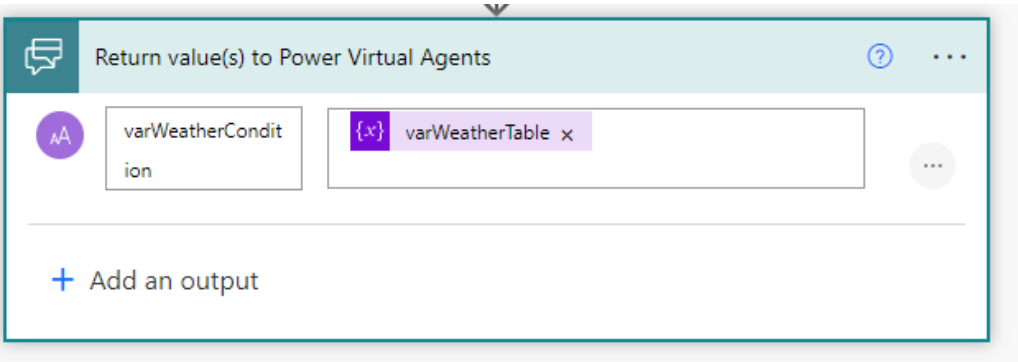
|Night Condition|@{body('HTTP_-_Get_Weather_Data_using_Azure_Map_Weather_API')['forecasts'][0]['night']['shortPhrase']}|

|Hours of Day Time Rain|@{body('HTTP_-_Get_Weather_Data_using_Azure_Map_Weather_API')['forecasts'][0]['day']['hoursOfRain']}|

|Hours of Night Time Rain|@{body('HTTP_-_Get_Weather_Data_using_Azure_Map_Weather_API')['forecasts'][0]['night']['hoursOfRain']}|

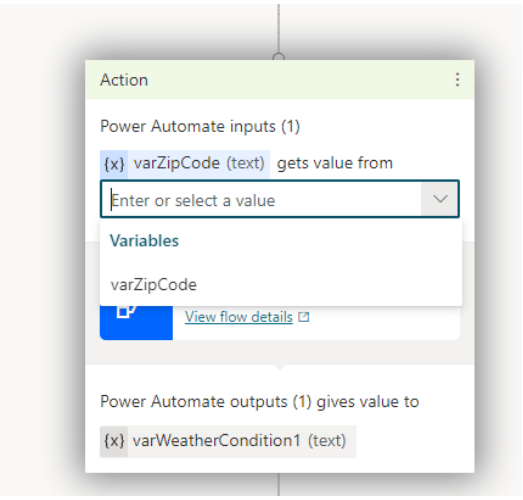


The final output is returned to Power Virtual Agent

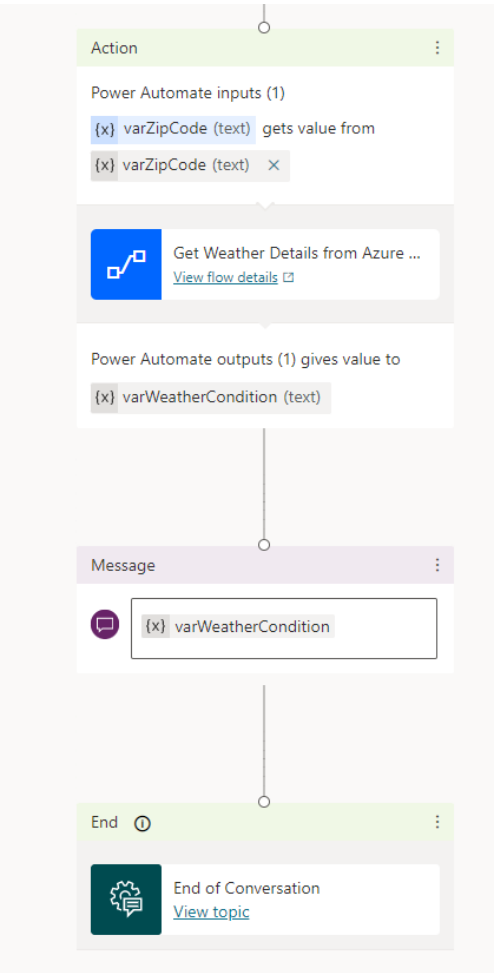


Calling Power Automate from PVA

We will add the Call an action node which gives us the list of power automate available for calling. We will select the recently created flow and pass the varZipCode variable as the input parameter

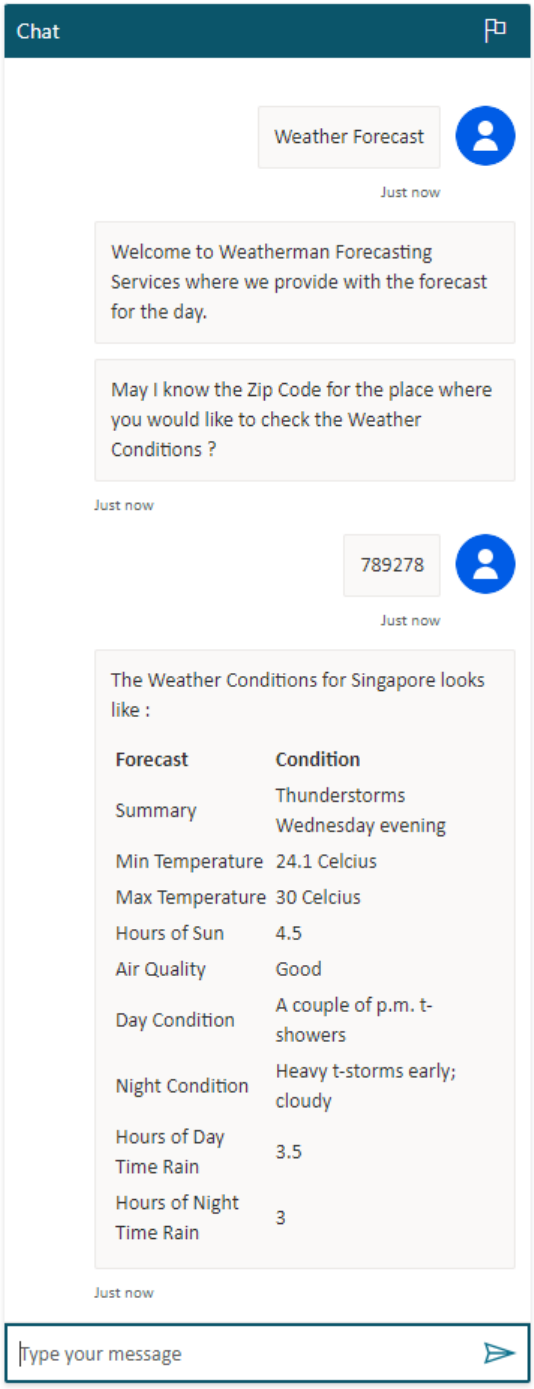


The output of the power automate will be stored in varWeatherCondition variable and show as a message to the end user. We will also add the end conversation node with a survey to close the conversation loop. In case of further continuing the conversation, we can ask further question node to expand the chat conversation possibilities.



Test the bot

Now lets test the bot using a trigger word Weather Forecast which will initiate the conversation with the bot that will convert the zip code into corresponding latitude and longitude using Azure Maps Address API and feed it into the Azure Maps Weather Daily Forecast API to get the forecast conditions for the next day.



Summary

Thus, we saw how we can use Power Virtual Agent to call Power Automate which in turn leverages Azure Maps API to fetch the weather conditions and show it as a table in Power Virtual Chat Bot using Mark Down language. We can extend the bot by publishing and deploying it into any of the channels like Teams, Slack, Facebook, custom website etc;

Migrate the Power Virtual Agent Bot across environments

Introduction

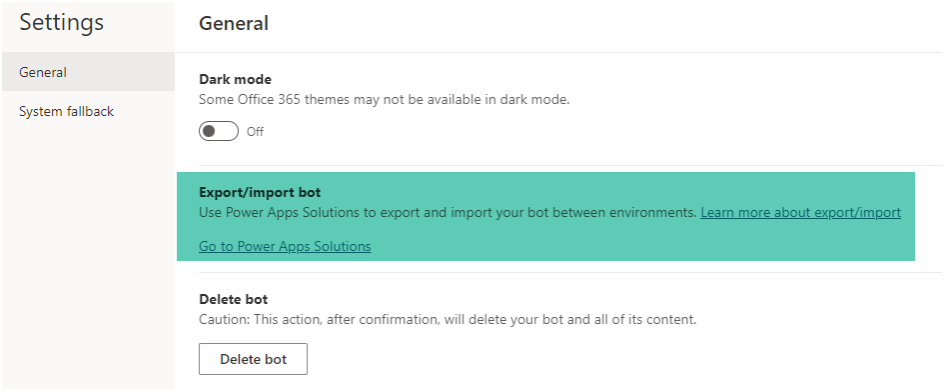
A bot that has bee created in one environment may need to be used in other tenants as well to enforce reusability or because we are moving it from the Dev to the Prod environment.

Power Platform provides the capability to export and import bots using Solutions so that bots can work across multiple tenants with almost zero rework.

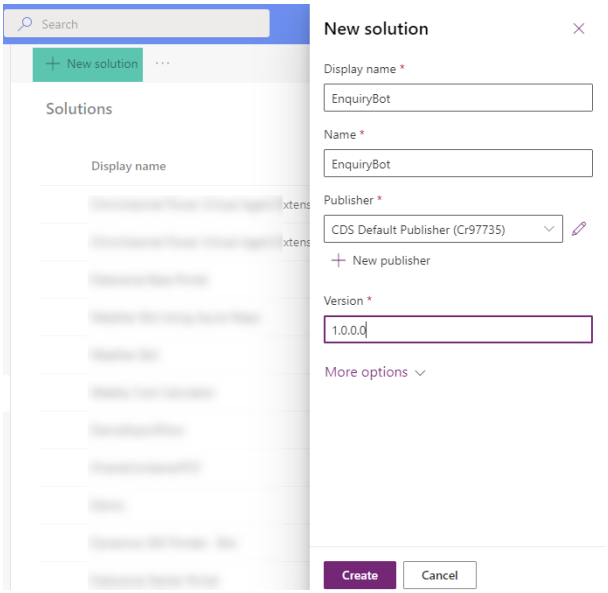
Note: The bot maker will require the minimum System Customizer security roles to use this feature.

Add bots to Solution

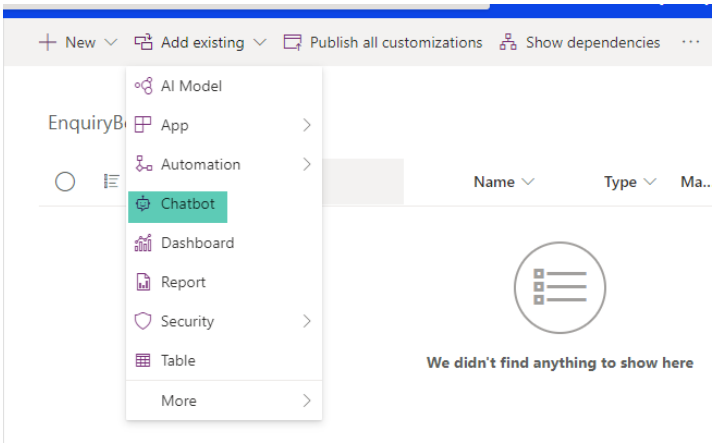
We will be making use of Solutions which acts as a container for one or more bots. In effect we wont exporting the bot, but the entire solutions are exported. Let's create a solution by heading over to the Gear icon and select General Settings.



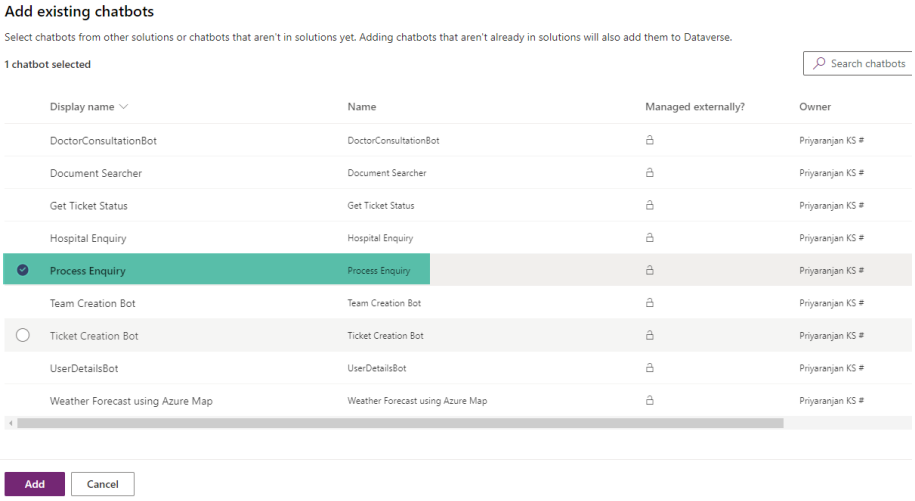
Click on Go to Power Apps Solution which will take us to Power Apps . Click on New Solution and provide the details using which the solution will be created.



Once the solution is created, click on "Add existing" drop down and select Chatbot so that we can browse and add the bot that we are trying to export.

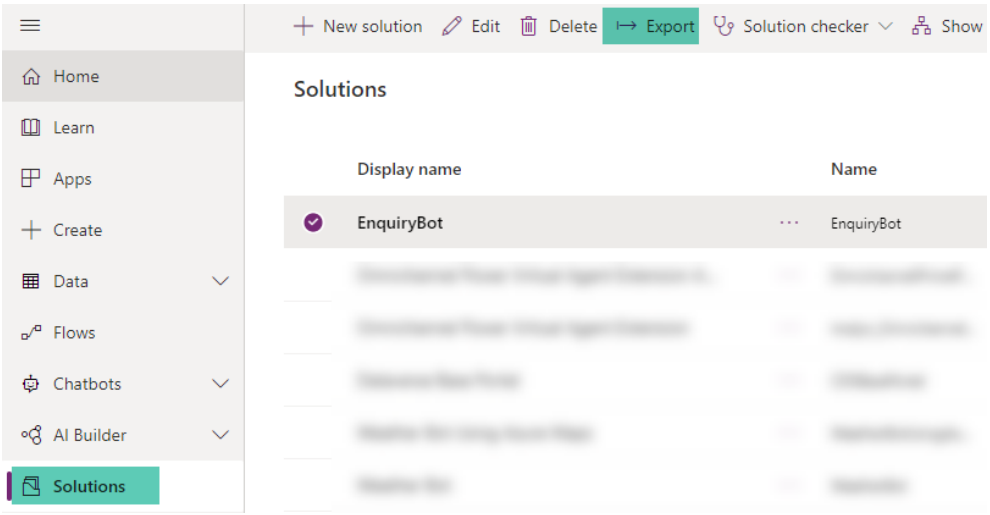


Select the bot and click on Add.

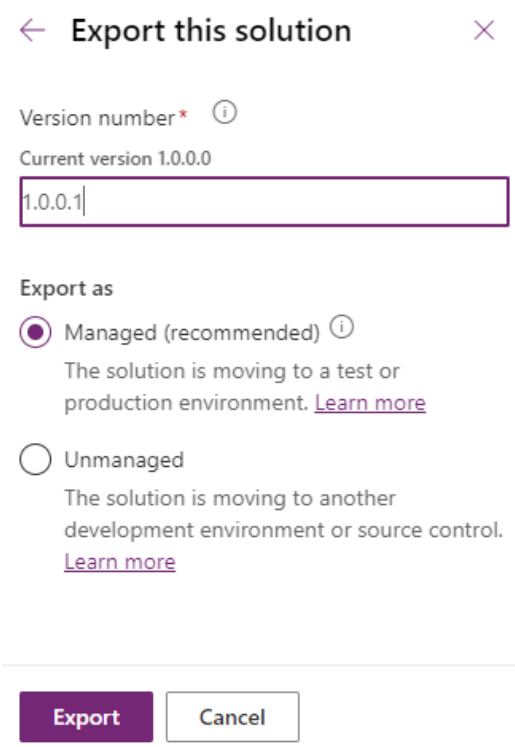


Export the Solution

To export the Bot, we will go to the Solutions tab and select our Solution where the bot has been added and click on Export.

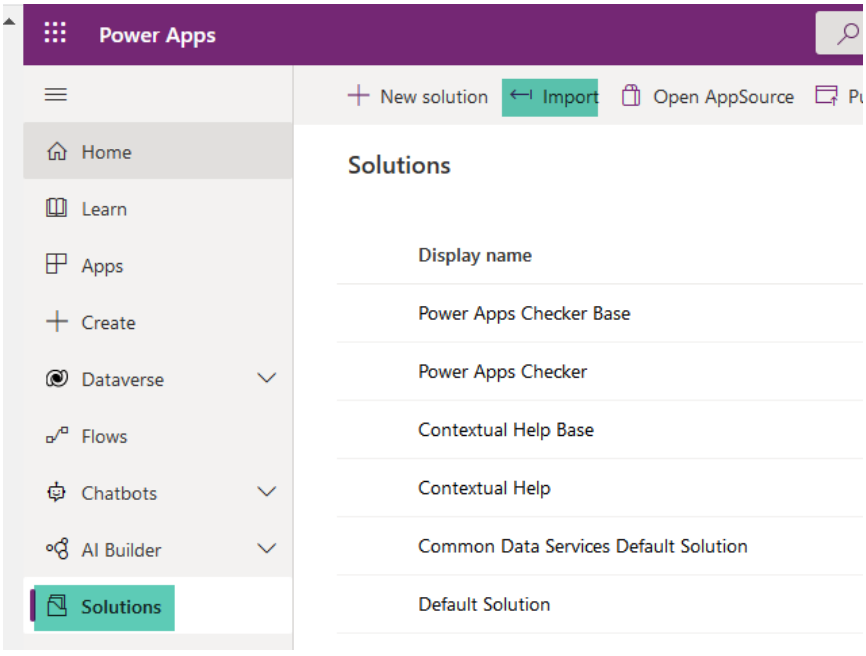


Based on the destination to which the bot is being exported select either manager or unmanaged option. Unmanaged solutions are used in development environments while you make changes to your application. Managed solutions are used to deploy to any environment that isn't a development environment for that solution like UAT or Prod.

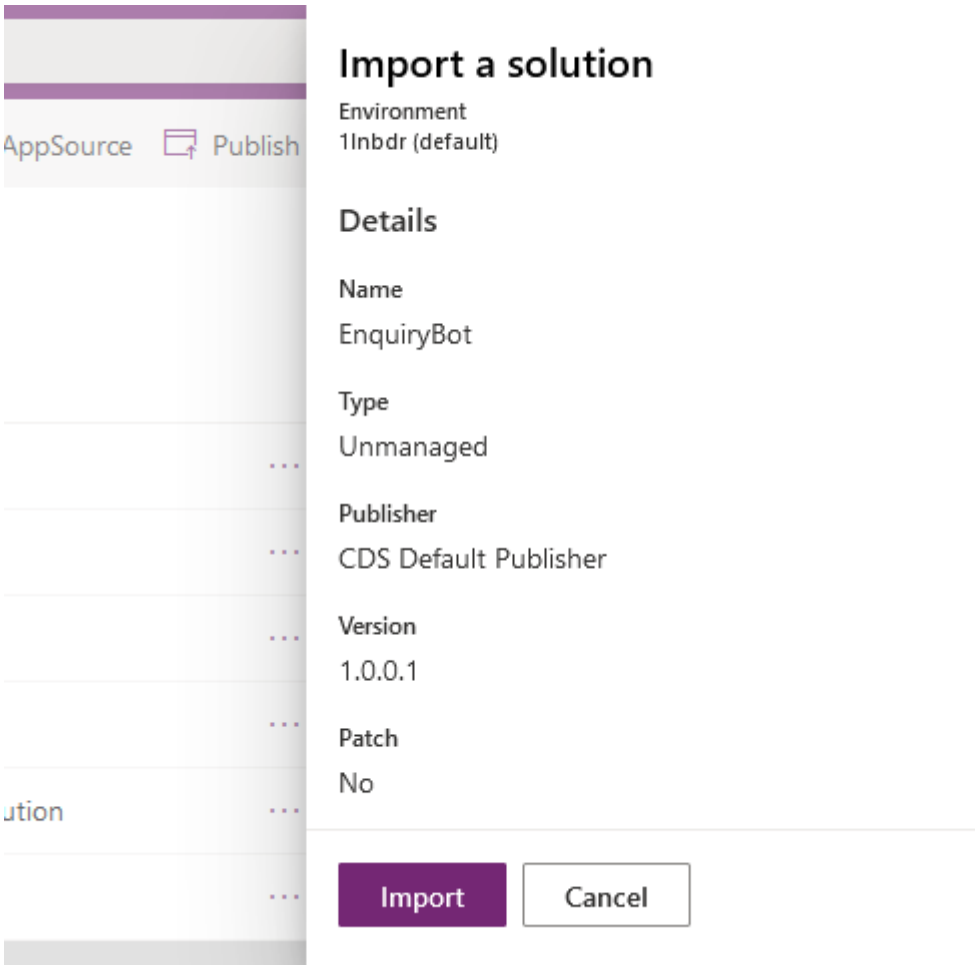


Import Solution

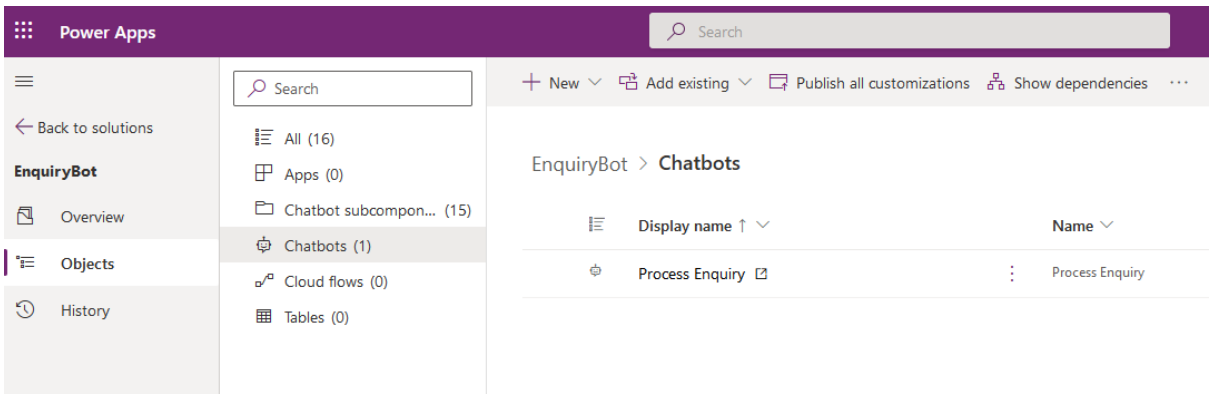
Now let’s head over to the new tenant where we want to import the solution. Visit make.powerapps.com and select Import option from the Solutions tab



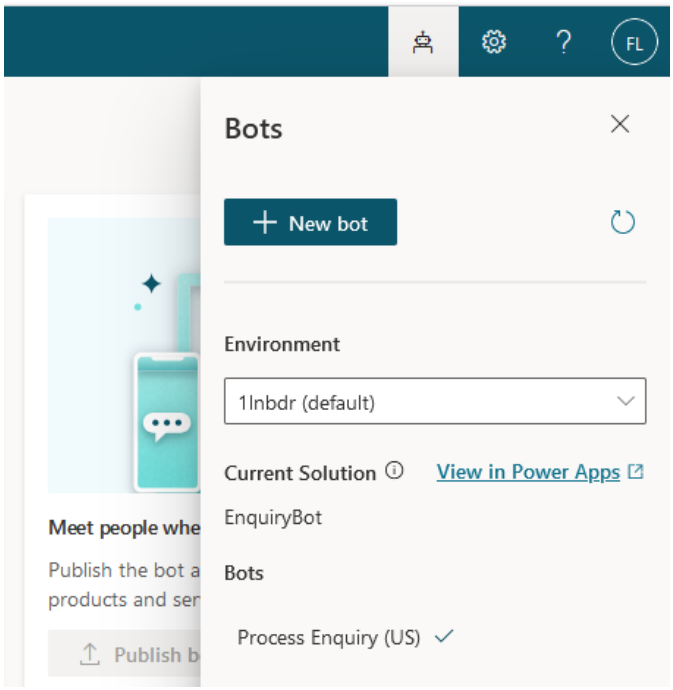
It will open a left pane from where you can browse and select the previously exported zip file. Click on Import.



We can see the bot in the imported solution in the new tenant

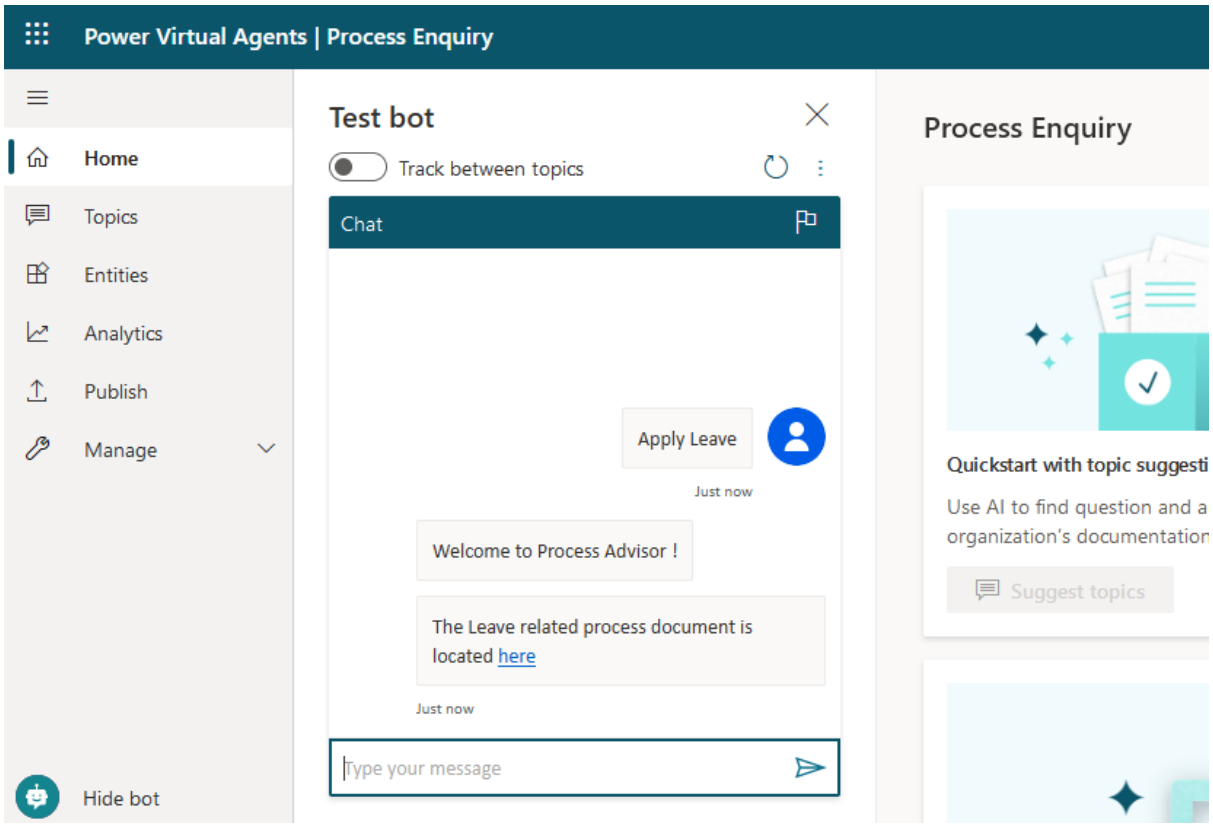


Now let's head over to Power Virtual agents in the new tenant. Click on the Bot icon in the top right corner and it will list out the available bots and here we can see the recently imported bot.



Test the Imported Bot

Doing a quick test of the bot, we can see that it works as expected even without making any further changes.



Summary

Thus, we saw how to export a bot using solutions and get it to work in another tenant by importing the bit along with the solution

Implementing Only For Teams Authentication in Power Virtual Agents

Introduction

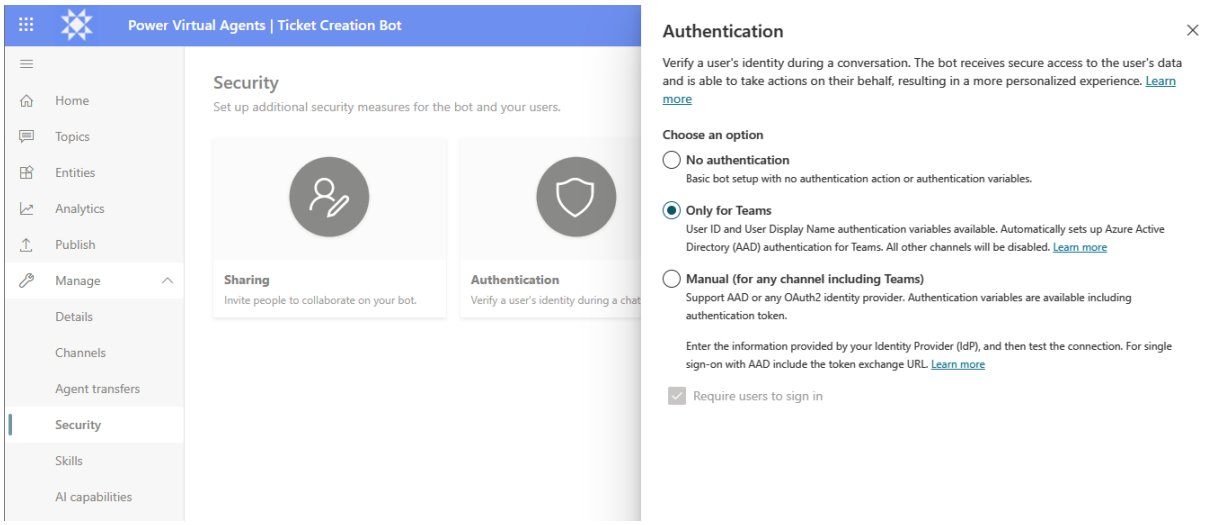
In this section we will see how to create a basic bot and use the Only for Teams Authentication. When choosing this option, only the Teams channel will be available for publishing and sharing the bot. All other channels will be disabled, and a warning will be displayed. This configuration option is optimized for Teams channel usage. It automatically sets up Azure Active Directory (Azure AD) authentication for Teams without the need for any manual configuration.

It uses the Teams authentication itself to identify the user, meaning the user will not be prompted to sign-in while in Teams.Once the bot is provisioned, lets head over to the Manage section in the left pane -> Authentication -> Select Only for Teams

When choosing this option, only the Teams channel will be available for publishing and sharing the bot. All other channels will be disabled, and a warning will be displayed. This configuration option is optimized for Teams channel usage. It automatically sets up Azure Active Directory (Azure AD) authentication for Teams without the need for any manual configuration. The following variables will be available in the authoring canvas after the Only for Teams option is selected:

The following variables will be available in the authoring canvas after the Only for Teams option is selected:

- UserID
- UserDisplayName



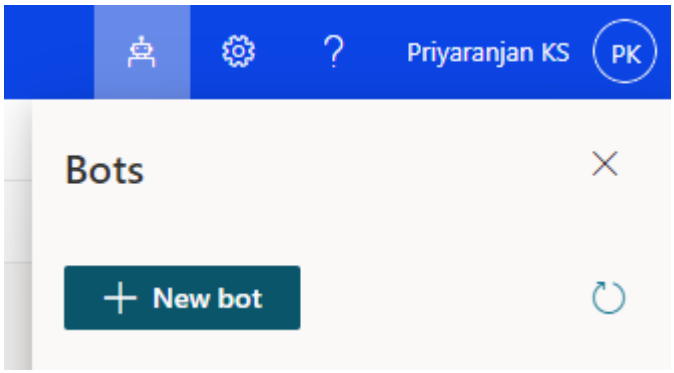
Business Use Case

We will try to create a bot that get user details from the User Profile. We will give the user an option to select whether he needs to fetch personal details or the user details of another user. For personal details fetching, we will use the UserID variable that comes along with the Teams Authentication enablement and for getting the user profile details of another user, we will get the display name of the concerned user and use Power Automate action to get the respective details.

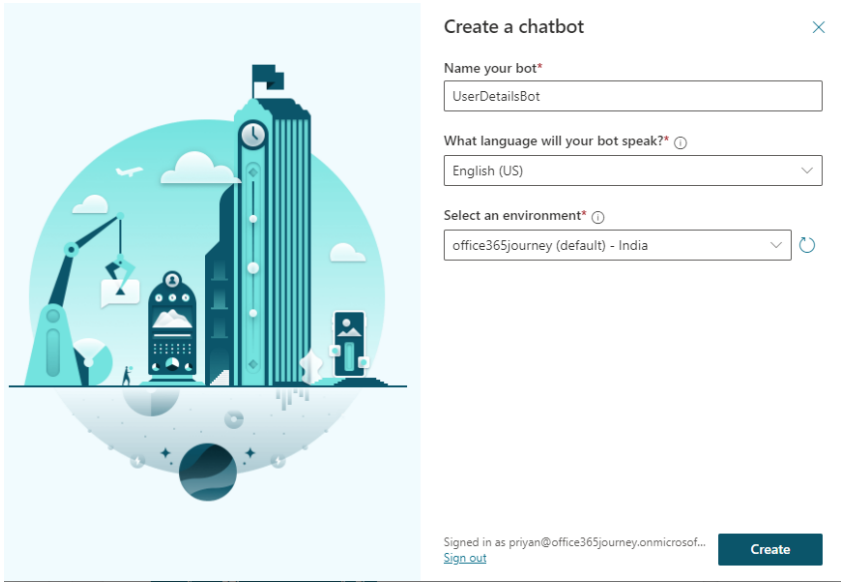
Implementation

To implement the Automated Enquiry system, we will use Power Virtual Agents where we can create a bot to attend to the queries and give automated responses based on the information we have in

the system. To start with we will head over to https://web.powerva.microsoft.com/ and click on the Bot symbol on the top right corner which lets you create a new bot

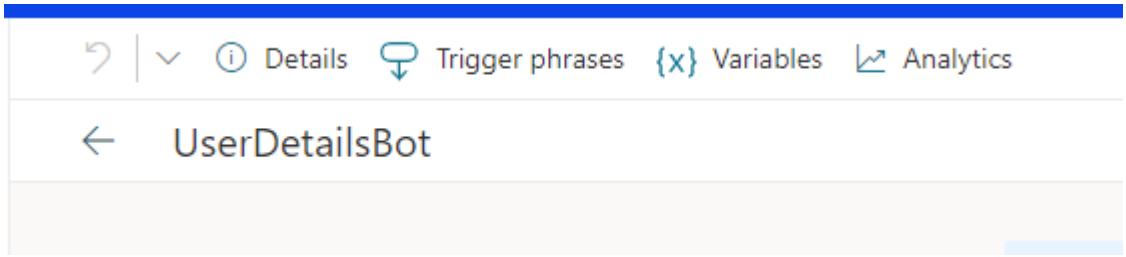


Specify the name and language for use in the bot as well as the environment where it should be provisioned.



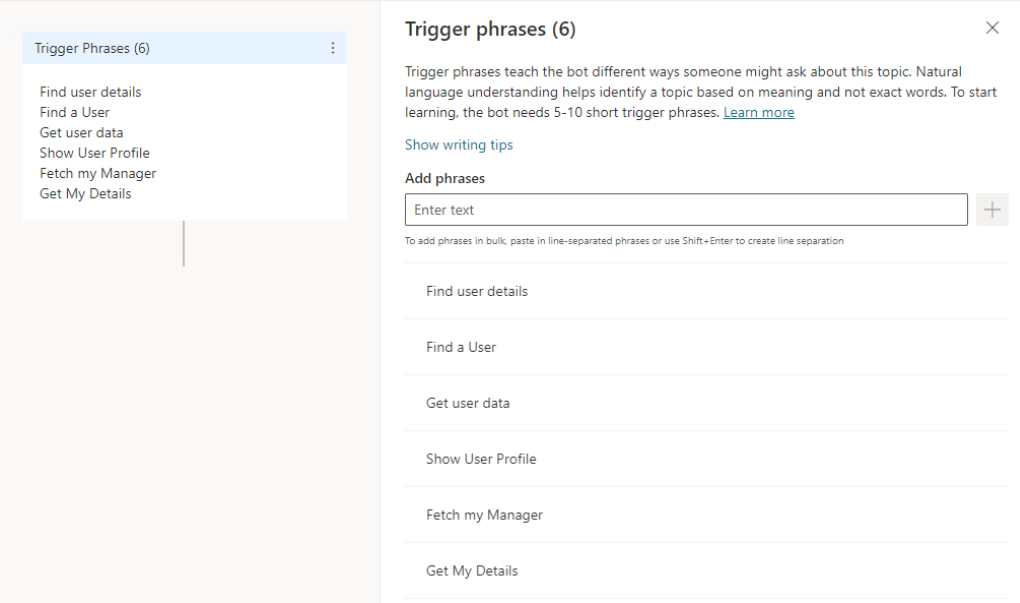
It will create a basic bot where we can add the customizations needed for our requirement. We can see the section Topics which by default lists multiple conversation topic listed out. A topic defines how a bot conversation will be initiated and how the bot would respond to user interactions.

As the first step, lets create a topic that will define a general conversation starter which will pick information about the user. We can have multiple topics created in a bot and based on the way conversation is proceeding, we can navigate between multiple topics. Click on New Topic and Select From blank to create a Topic and we will name it UserDetailsBot.

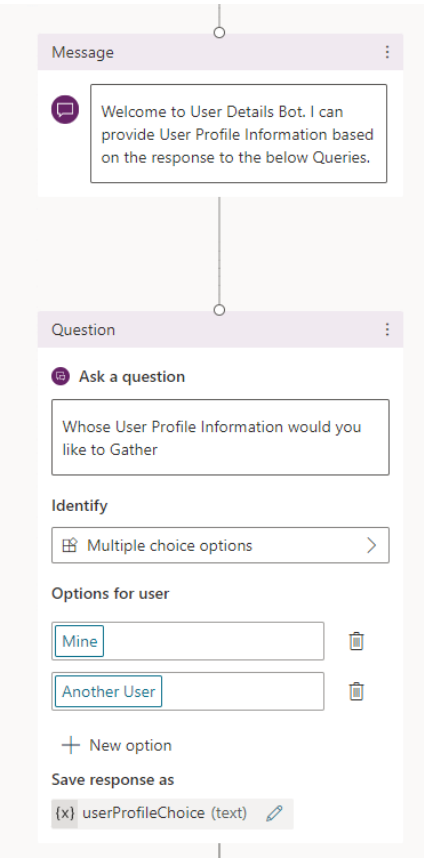


To trigger the topic, so that the control flows to that specific topic, we will define few words called Trigger Phrases that will transfer the control to these topics. The trigger phrases can be single

keywords or a group of words and to cover a broad spectrum of possible trigger conditions, It is good to mention 5-10 different and still related phrases. Click on Trigger phrases that will open a right pane where we can add the trigger words.

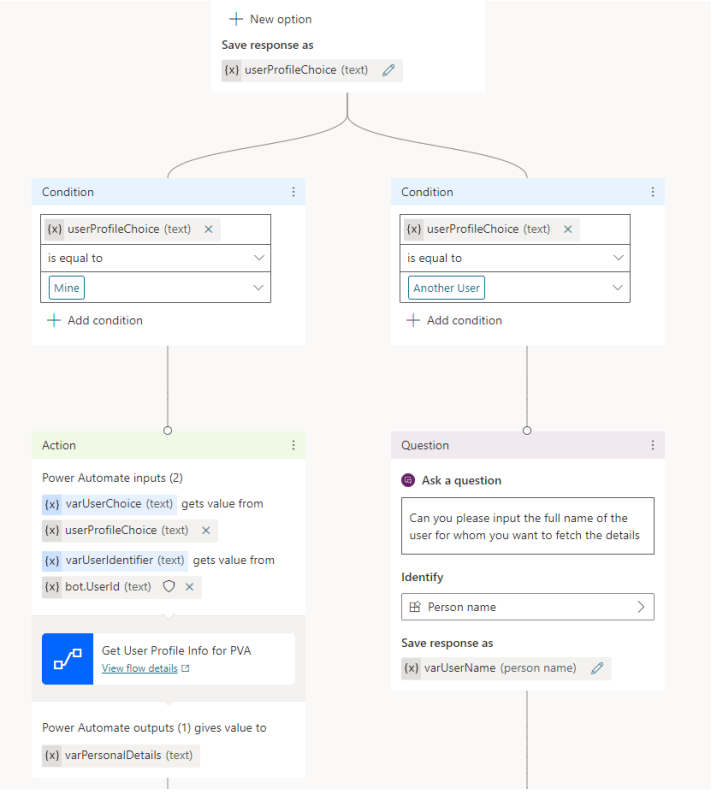


Let's add a message box with the welcome message and add the node add a question to ask for more details from the user related to whose profile information he would like to view – Own/Another User.The input from the user will be saved in the variable userProfileChoice.

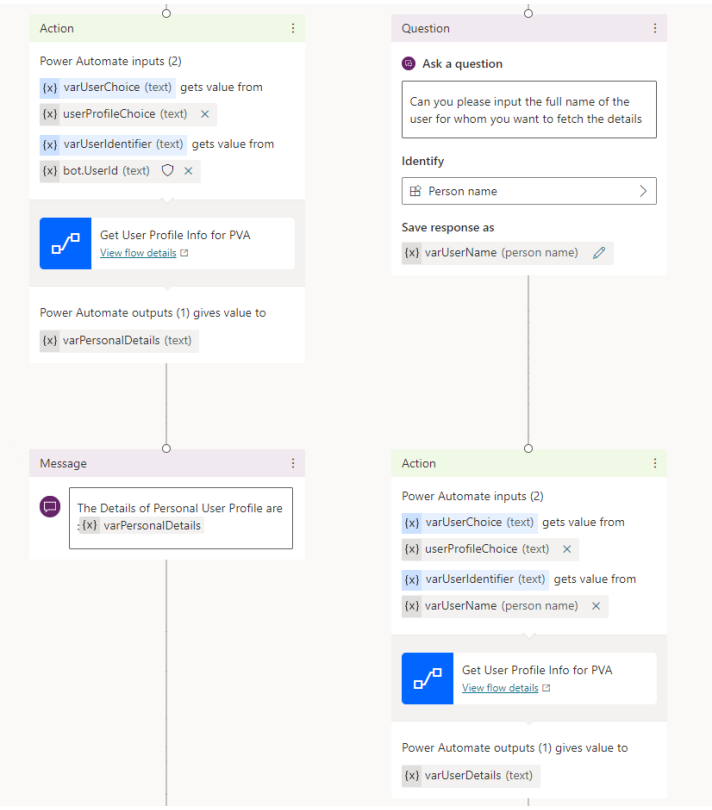


Based on the choice selection,PVA will automatically create 2 branches for getting the personal details Or for another user. In case of userProfileChoice = Mine , we will add the call an action to

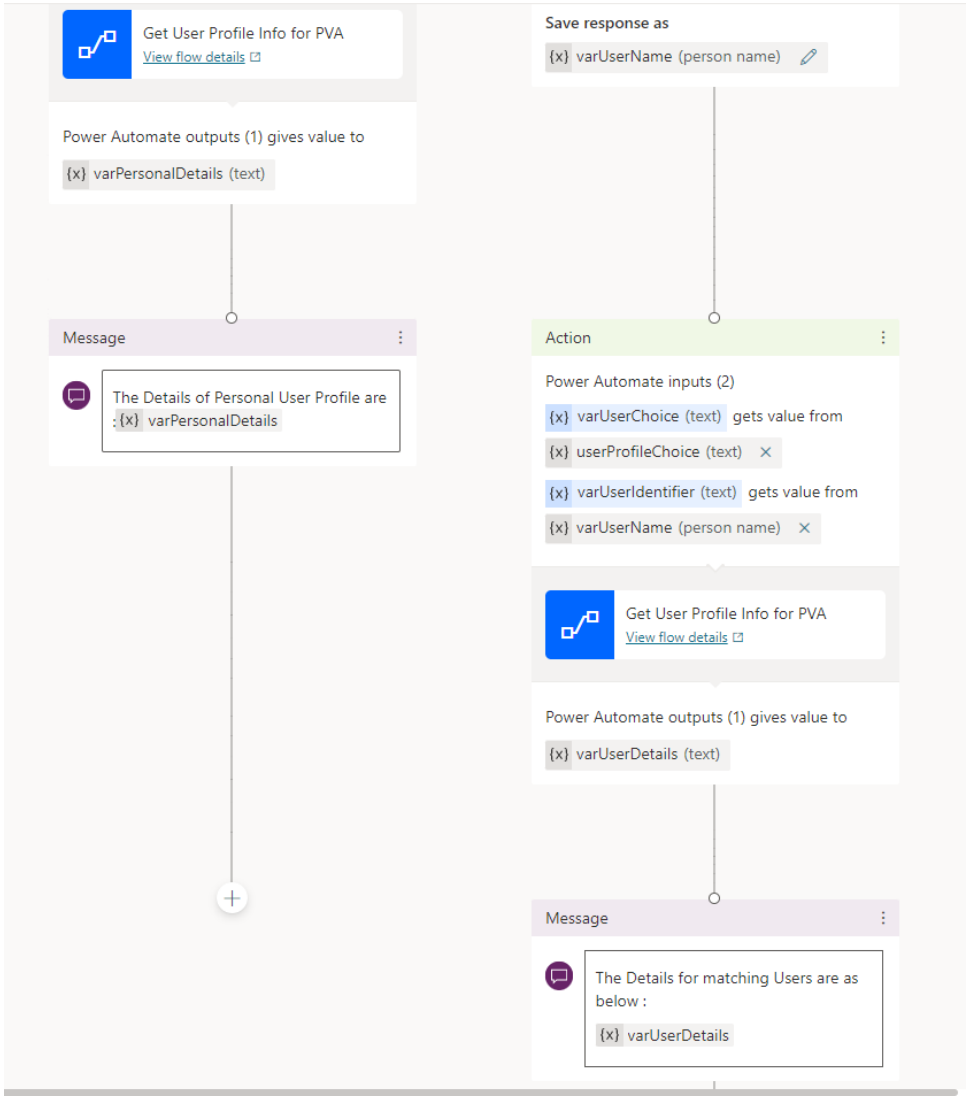
create a power automate and pass the choice value and the user id which we will get from the system variable bot.UserId.



While in case of another user, we will ask for more information about the user name of the person whose details has to be fetched and will call the power automate by passing in the choice value and the user name obtained from the user.

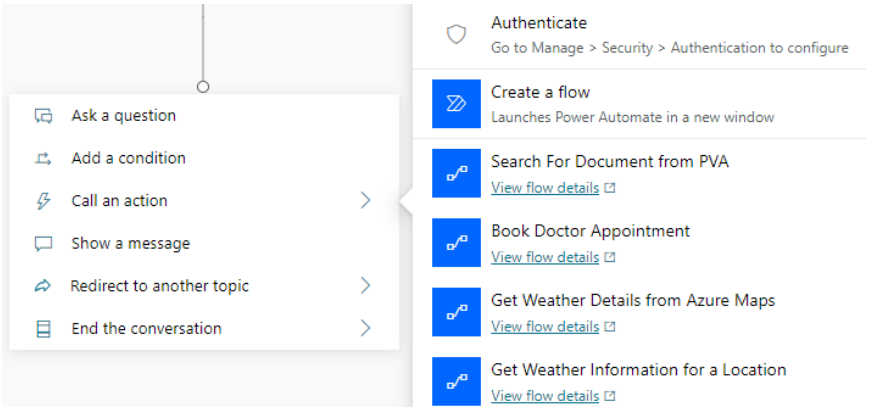


In both the branches we will show the details using a message box

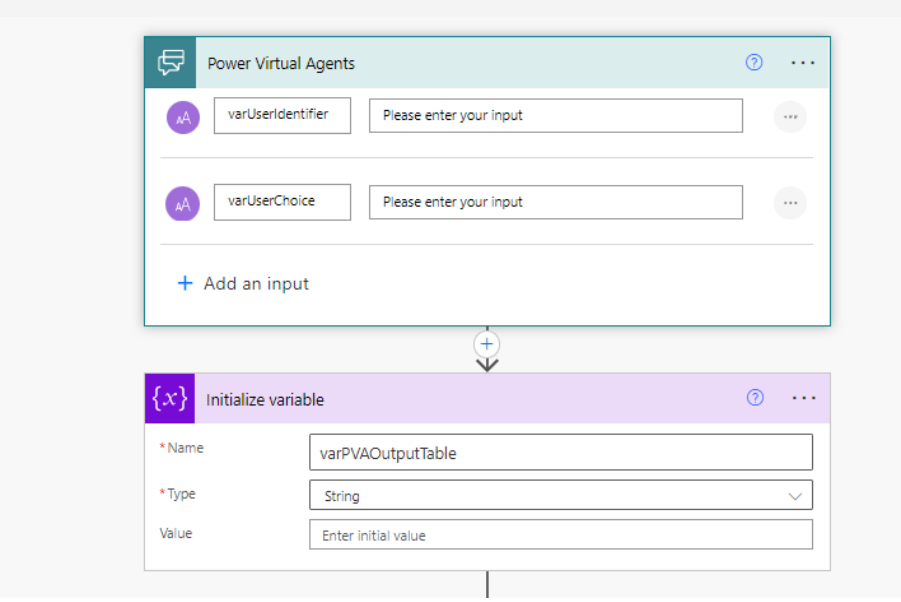


Create the Power Automate

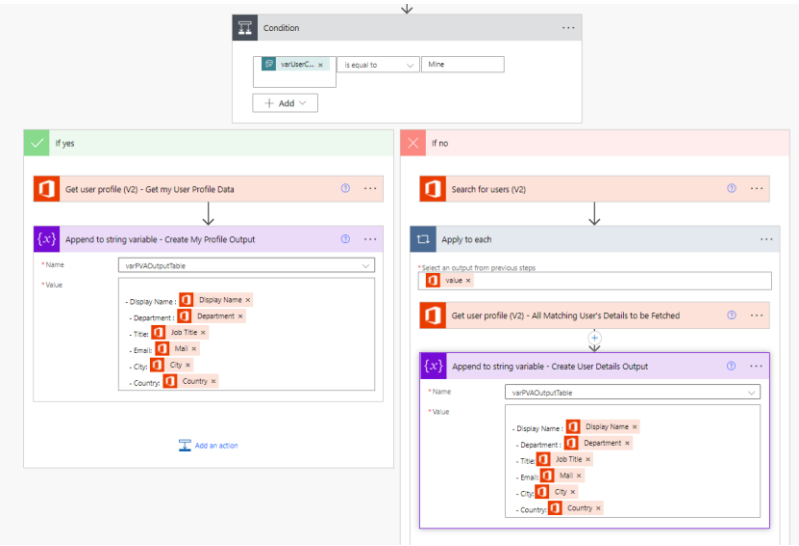
To create the power automate we used in the above PVA, click on call an action and select Create a flow.



This will open Power Automate in new tab where we can add 2 variables to get the choice value and the user identifier from PVA. We will also initialize a variable to hold the data that needs to be passed back to PVA.



We will then add a condition to check the choice value selected by the user. If it evaluates to Mine, will add the logic in the left branch else the logic will be added to the right branch for getting another user's details.



In case the selected option is mine, the system variable User ID will be available in the flow which we will use directly in the Get User Profile action to get the details of the current user. We will then append it to the output variable using markdown language.As PVA do not support html tables, we will use markdown language to create a bulleted list of items to show in the chat window. To create an unordered list, add dashes (-), asterisks (*), or plus signs (+) in front of line items. Indent one or more items to create a nested list.In this sample we will be using the below mark down syntax :

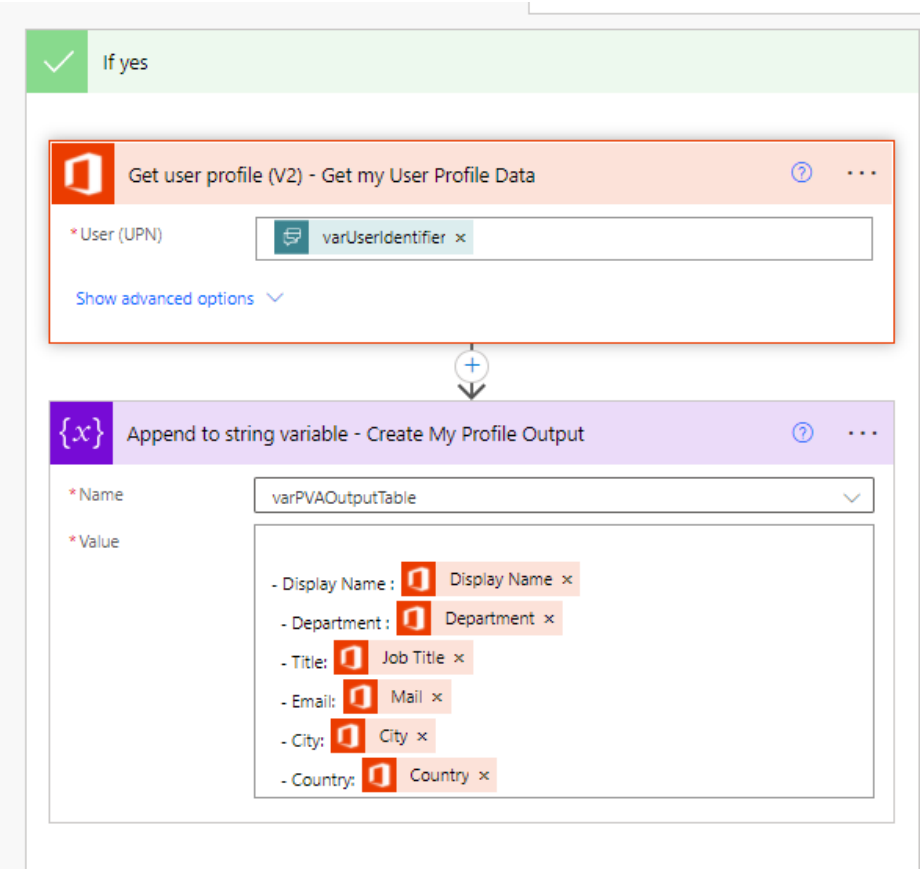
```
- First item
- Second item
- Third item
    - Indented item
```

```
    - Indented item
- Fourth item
```

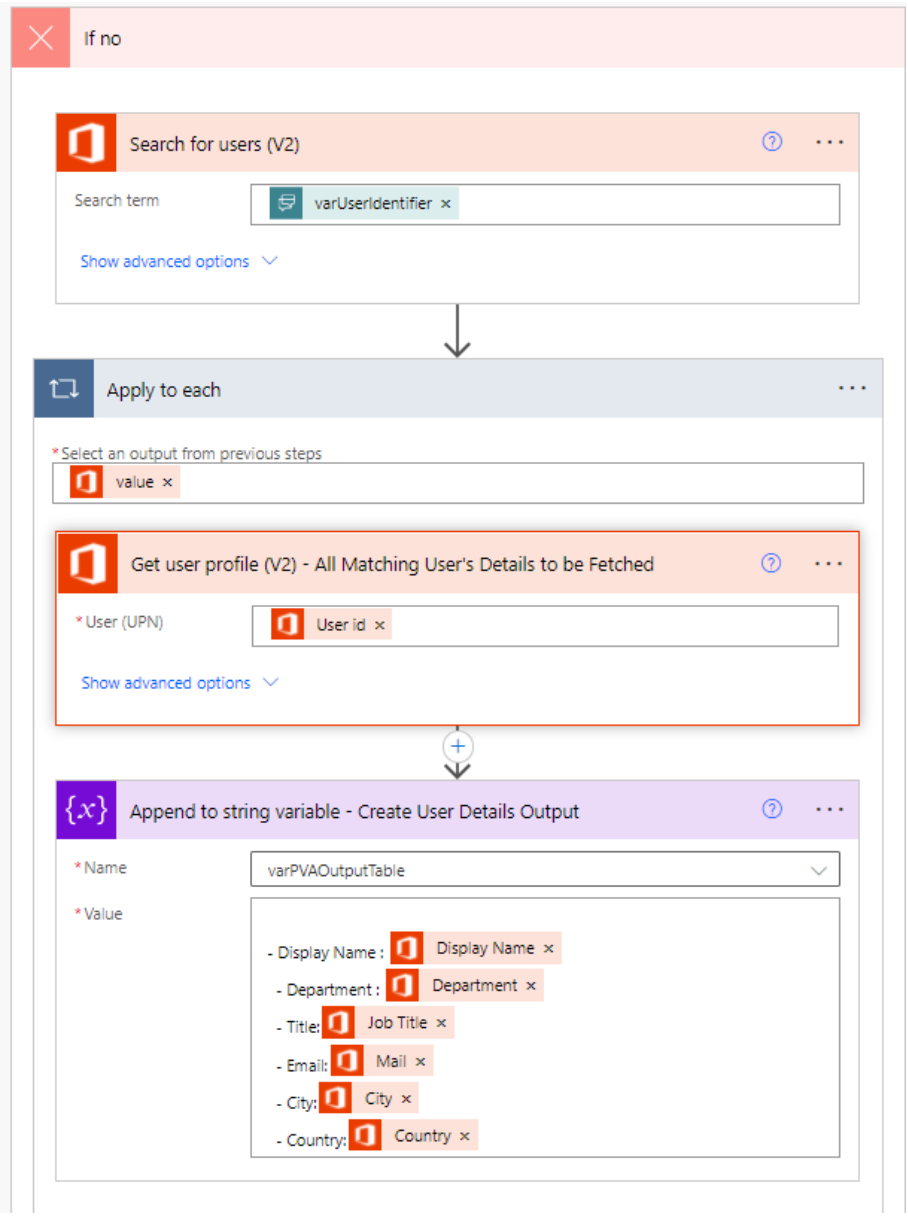
```
Which will generate the output as :
```

- First item
- Second item
- Third item
    - Indented item
    - Indented item
- Fourth item

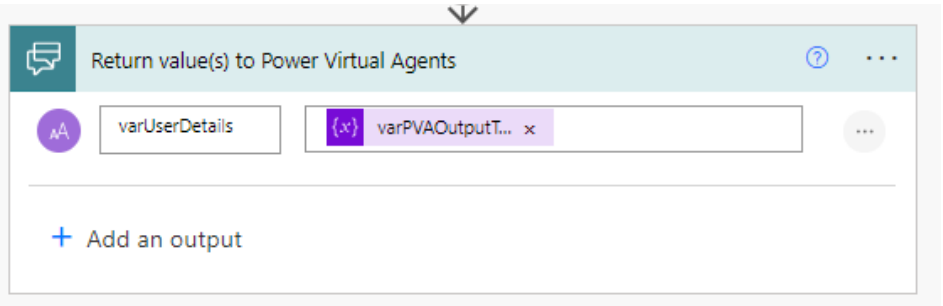
So in the value box, we will add the hyphen markdown followed by the User Profile Column name and the corresponding value.



In case the user has selected the choice to get another user's details, the right branch will be called and as we have only the user's name and not the id, we cannot use the get user profile action which expects the ID, instead we will call the Search for users and pass the name to it. It will return a collection of users with matching names. We will use an apply to each action to loop through the output and get the ID of each user in the collection and call the Get User Profile action to get the details of each user. The detail of each user is then appended to the output variable using markdown language.

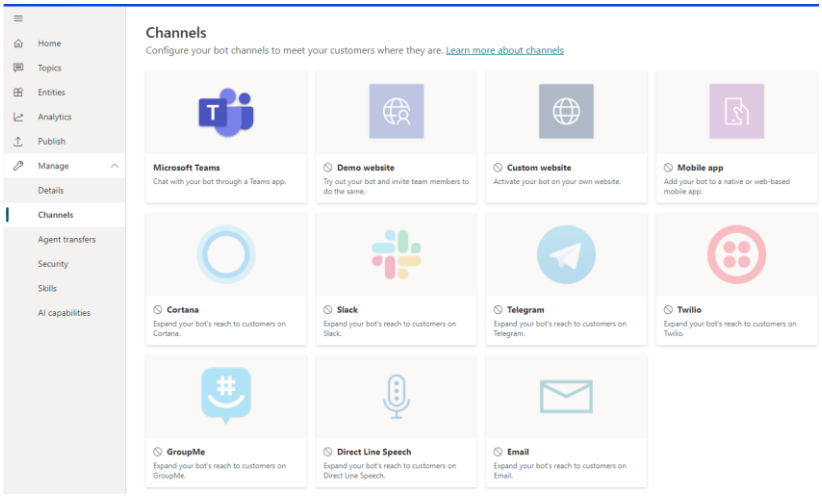


Let's pass the output variable back to the PVA

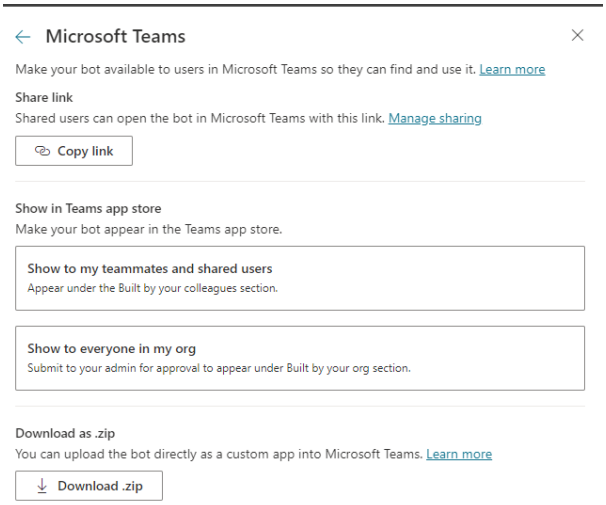


Publish the bot

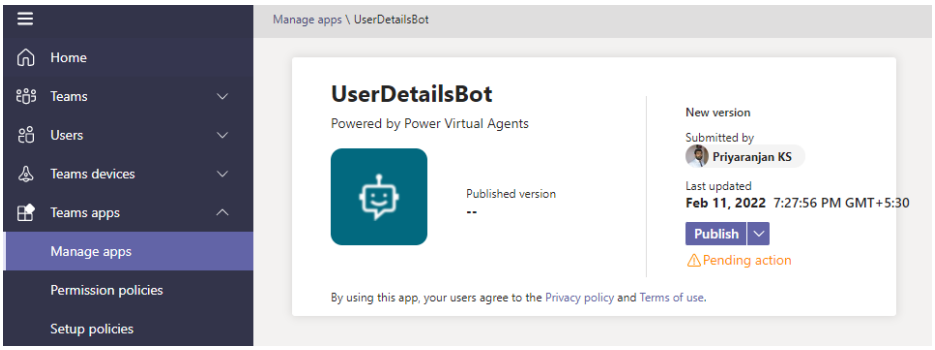
If we try to publish and deploy the bot, we will be able to see only the Teams channel and all other channels with be greyed out as we have used Teams Only Authentication



Selecting Teams will provide us the option to either download the zip and upload as a custom app or send the app to the teams admin for approval so that he can publish it and make it available to the whole organization.



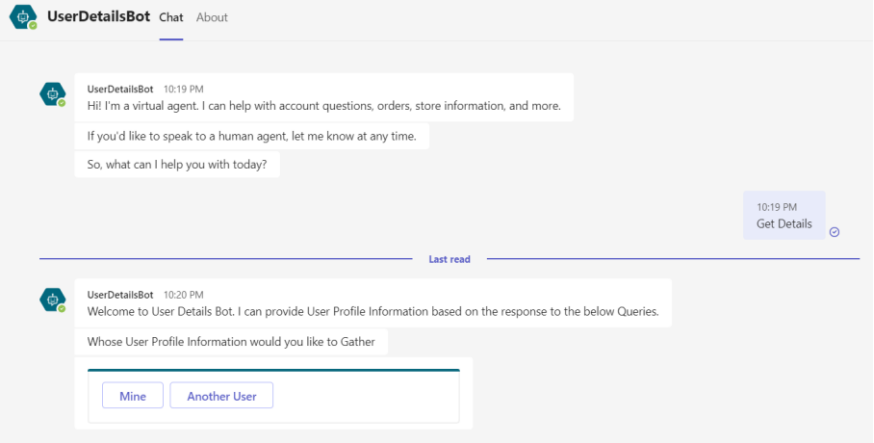
The admin can publish the bot from the Teams admin centre



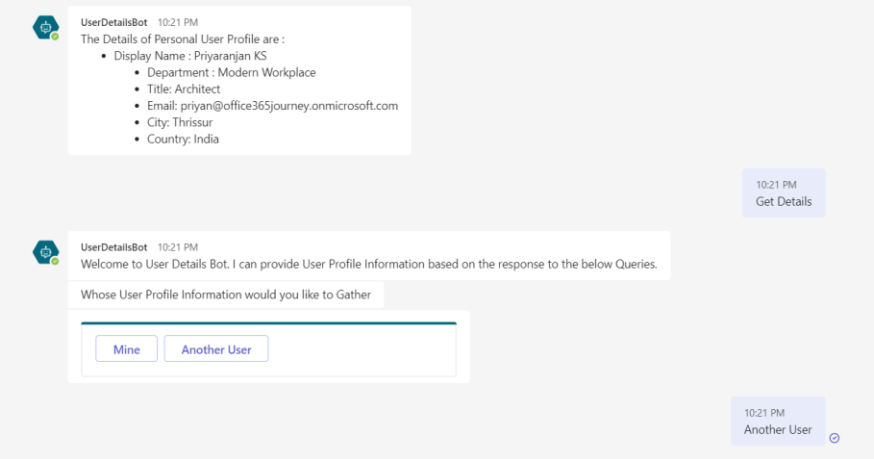
After some time, we will be able to see it in Teams and can add the bot as an app.

Test the bot

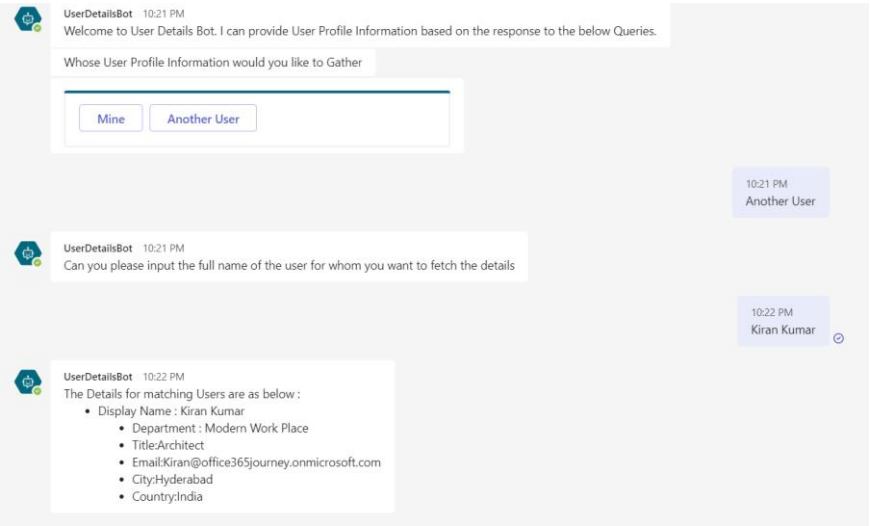
Let's test the bot using the trigger keyword – User Details which asks for choosing a selection Mine/Another User.



Selecting Mine picks personal user profile details. Let's also try fetching details of another user.



Selecting another user choice will ask for details of the other user's name and pick the details of the user



Summary

Thus we saw how to use the Only Teams Authentication to use the system variable bot.userId to work with one of the business requirements where we can pass it to Power Automate for further processing and we also saw how to deploy the bot to teams for a seamless authentication experience.

Send Proactive Message to Microsoft Teams from Power Virtual Agent

Introduction

Once we have published your bot and made the bot available to end users in Microsoft Teams, we can notify users in Microsoft Teams with proactive messages and adaptive cards. Proactive messages use Power Automate flows to deliver the content. This feature enhances the interactive nature of a Power Virtual Agent implementation.

Business Use Cases

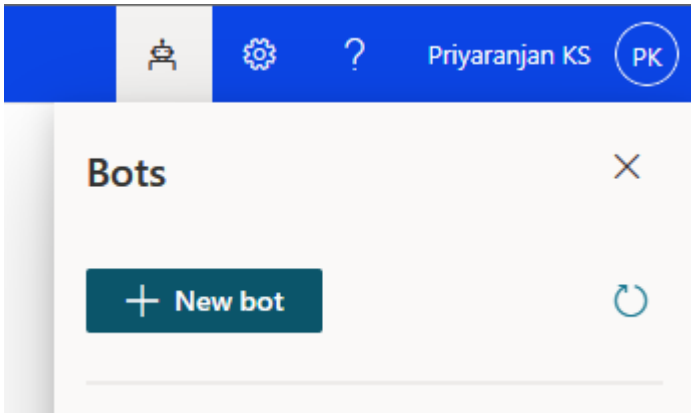
A scenario in which this becomes handy is when:

- You are creating a Ticket with the Support team via Power Virtual Agent and on creation of the ticket, a set of users (Resolvers) will be informed with a message about the created ticket
- On update of the ticket, the ticket creator will be informed with a message about the change
- At times, the ticket would need more information and we can send an adaptive card to the user, to fetch the missing details.

In this article we will see how we can send a proactive message to the user about the details of the created ticket, and we will also explore how we can inform the user about the changes made to an already created ticket with a message.

Implementation

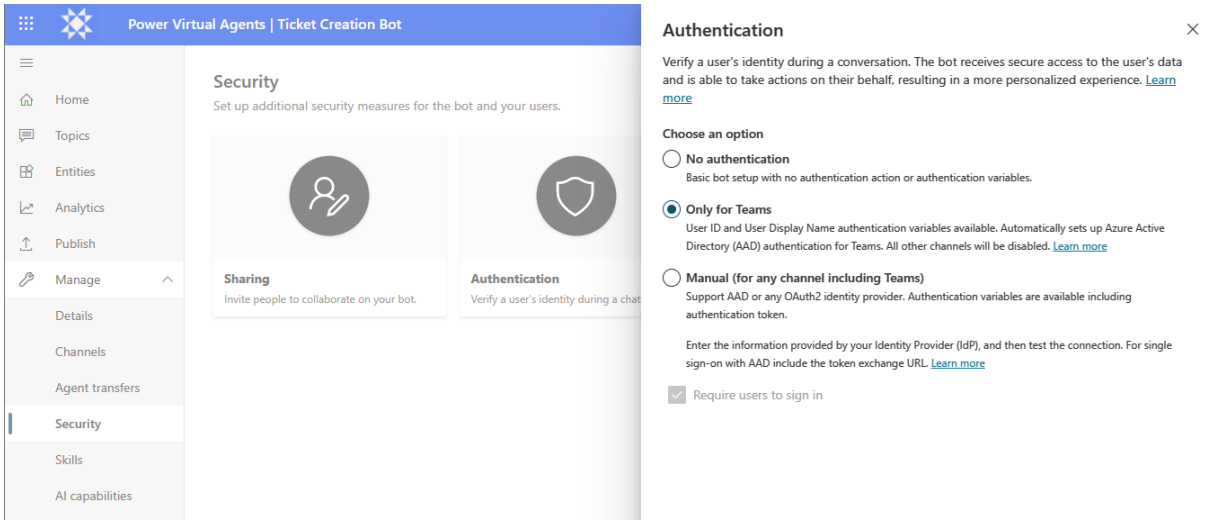
Let's create a bot using which we can create a ticket/issue in SharePoint List.We will start the bot creation by selecting the bot option from the top right corner and select "New bot". It will open up the window where we can specify the bot's name and language.



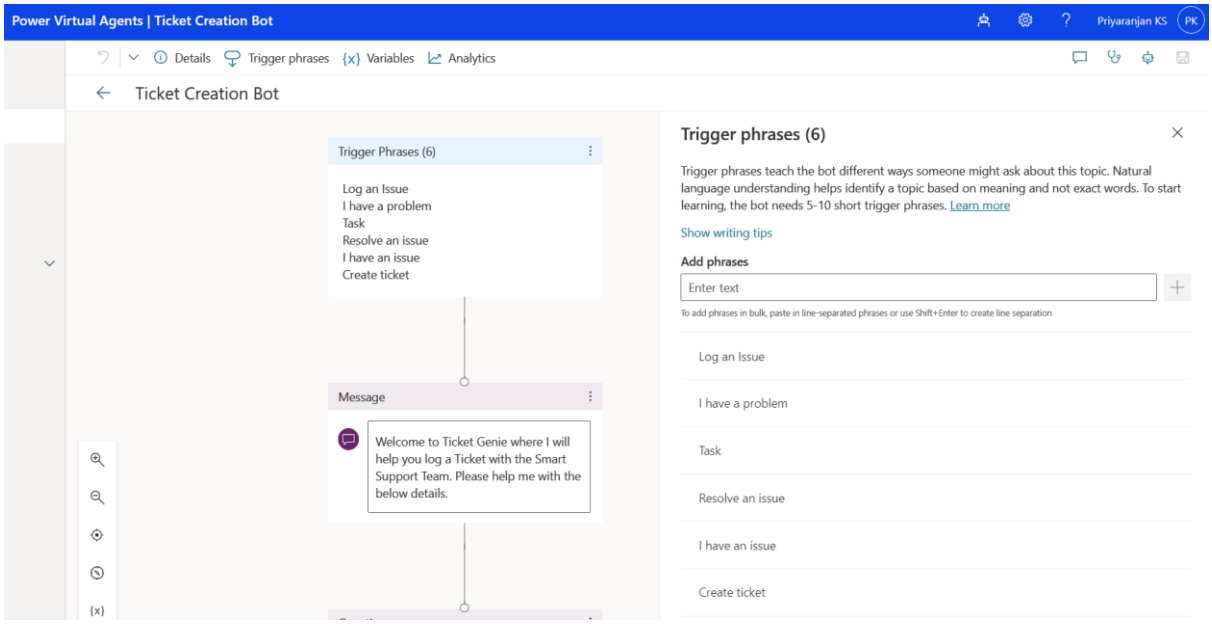
Once the bot is provisioned, lets head over to the Manage section in the left pane -> Authentication -> Select Only for Teams

When choosing this option, only the Teams channel will be available for publishing and sharing the bot. All other channels will be disabled, and a warning will be displayed. This configuration option is optimized for Teams channel usage. It automatically sets up Azure Active Directory (Azure AD) authentication for Teams without the need for any manual configuration. The following variables will be available in the authoring canvas after the Only for Teams option is selected:

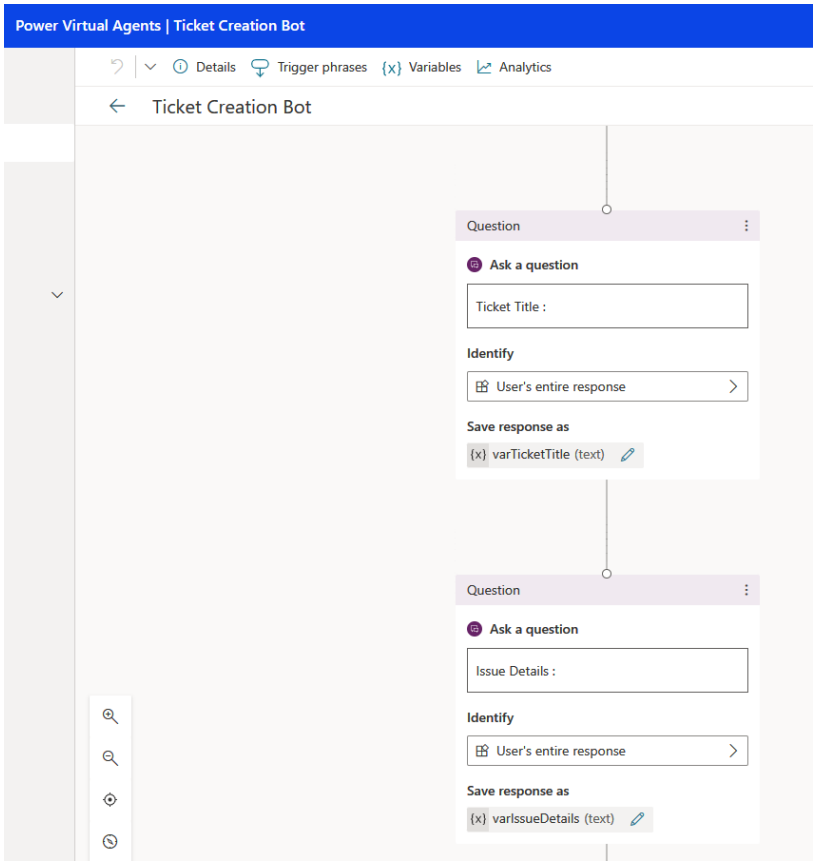
- UserID
- UserDisplayName



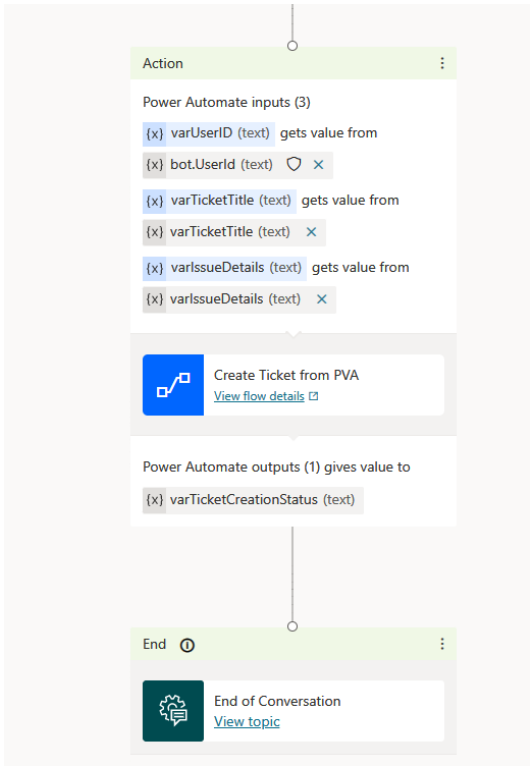
Now let's create a new topic and select the trigger phrases to define the words that will invoke the conversation with the bot. The trigger phrases will help in defining the words and phrases that will start the conversation with the bot. We have defined 6 such phrases and the more different and related words we add to it, better the chances of bot understanding the trigger condition. We will also add a welcome message to greet the user.



To accept the input from the user regarding the ticket title and description we will ask 2 separate questions and store the user input in the respective variables

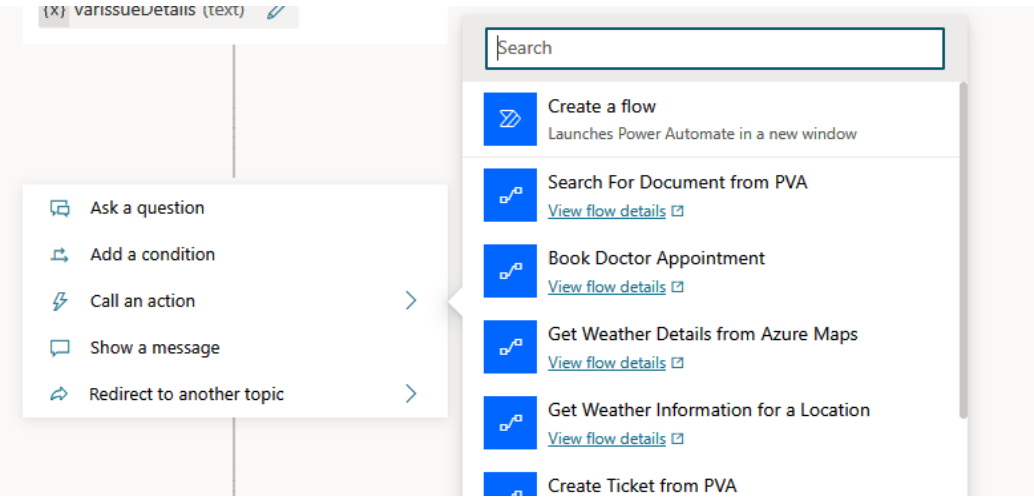


We will then call the Flow which we have created (described in the next section) to create the ticket and do further processing and end the chat with a survey to get the feedback from the user.

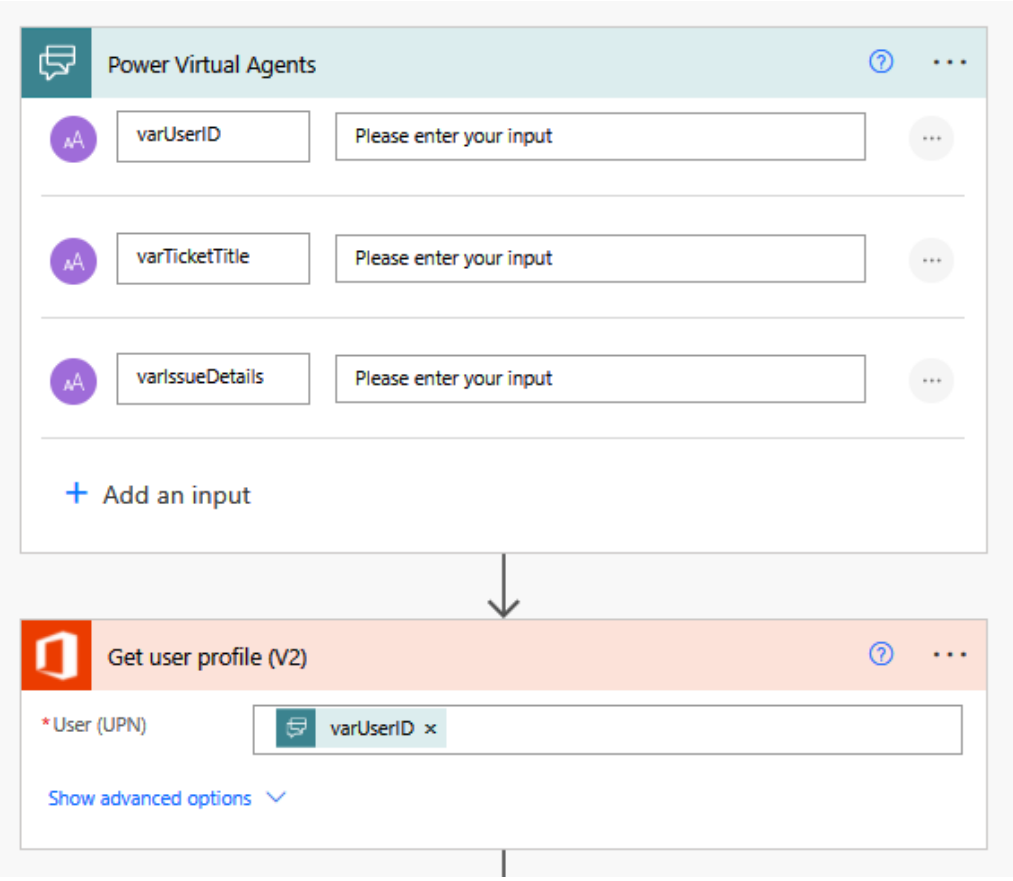


Add the Ticket Creation Flow

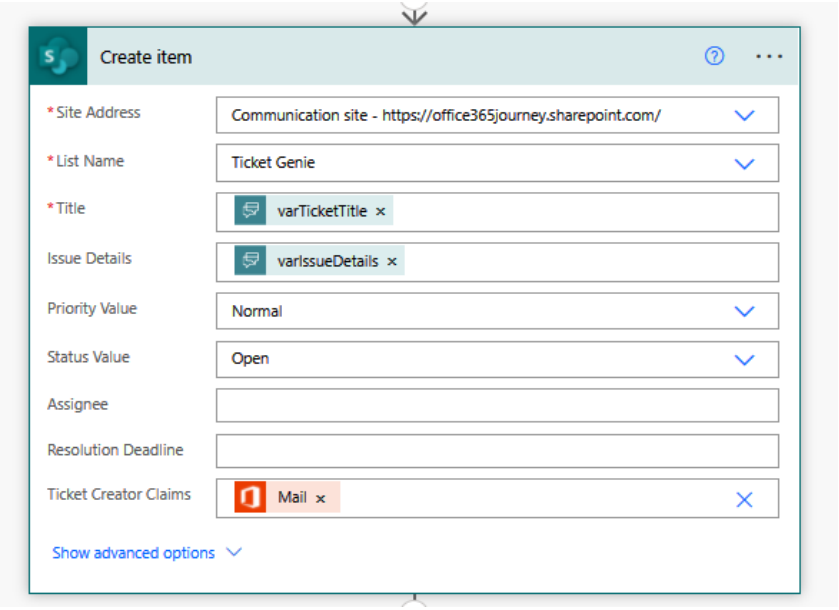
The ticket creation flow can be created from the Power Virtual Agent by Selecting Create a flow from the below node.



Within the flow, we will define 3 input variables to accept the UserID, Ticket Title and Ticket Detailers. As a follow up action, we will use the received User ID to pass it to the Get User Profile action to fetch the details like User Email.

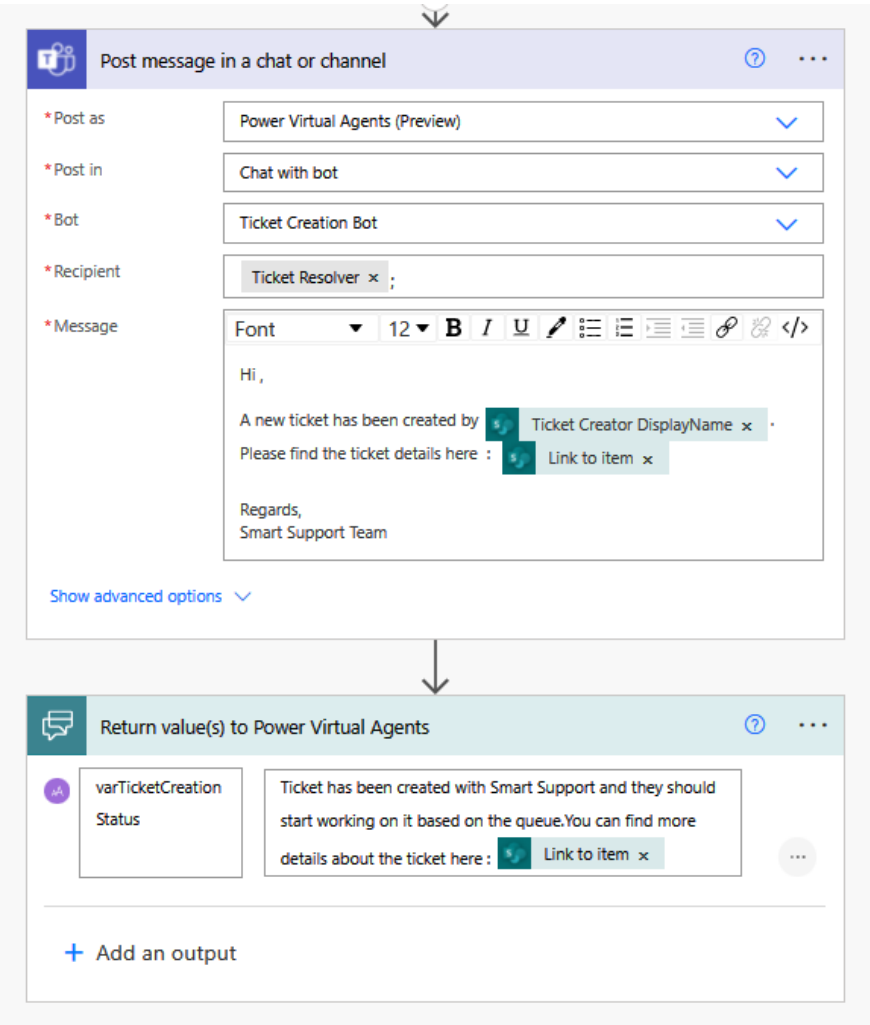


Followed by that, we will add the action to create the ticket/issue as a list item in SharePoint List and update the fields like Title, Issue Details and Ticket Creator Email ID.



Adding Proactive Message

Once the ticket has been created, lets send out a Proactive message to the Ticket Resolver intimating him about the ticket creator and the link to the ticket so that he can start working on the ticket. The message will be posted into the Teams chat by the Power Virtual Agent.



Finally, we will send back the response about the ticket creation and the link to the item so that user can track it.

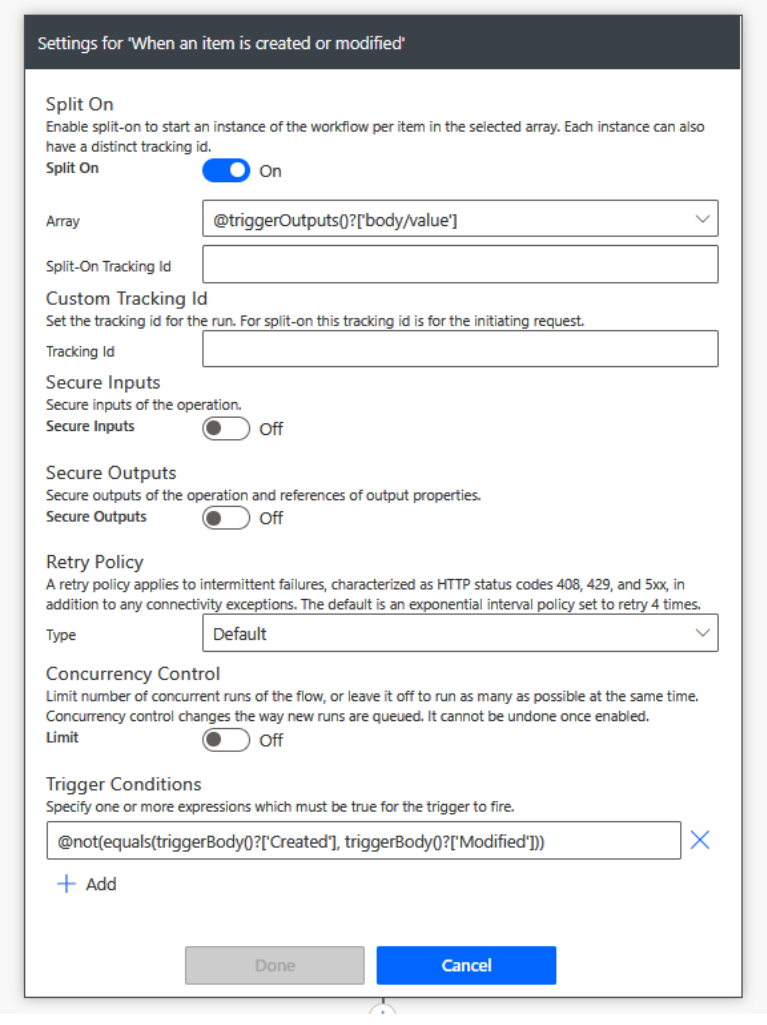
Thus, this completes the Power Virtual Agent Bot creation for creating the ticket by accepting input from the user.

Proactive Message on Ticket Updation

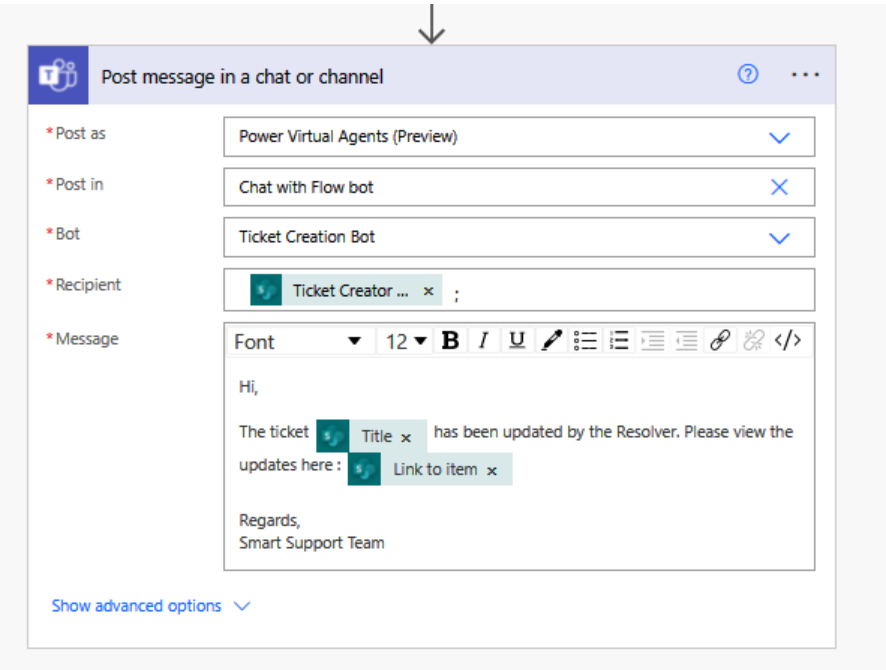
Now let's see how we can create a standalone flow attached to the list where the ticket/issue resides that sends out proactive message to the ticket creator when an update to the ticket happens.

We will create an automated flow that, get triggered on item update.To ensure that the flow is triggered only update and not on created event, we will click on the 3 dots of the action->Settings and update the trigger condition with the expression that ensures Modified Time is not equal to Created time which is a good way to check the event is a modification trigger.

@not(equals(triggerBody()?['Created'], triggerBody()?['Modified']))



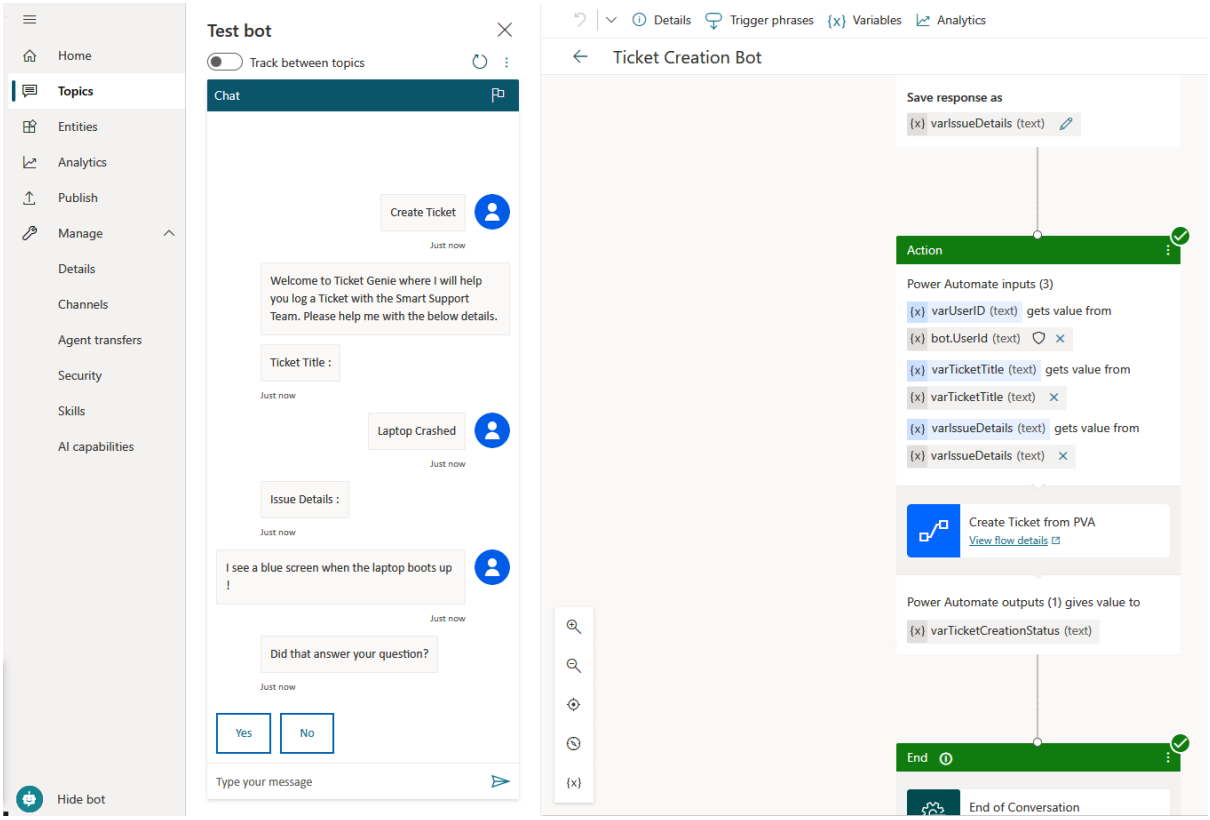
We will then add the action to post the proactive message to the Ticket Creator with the ticket details



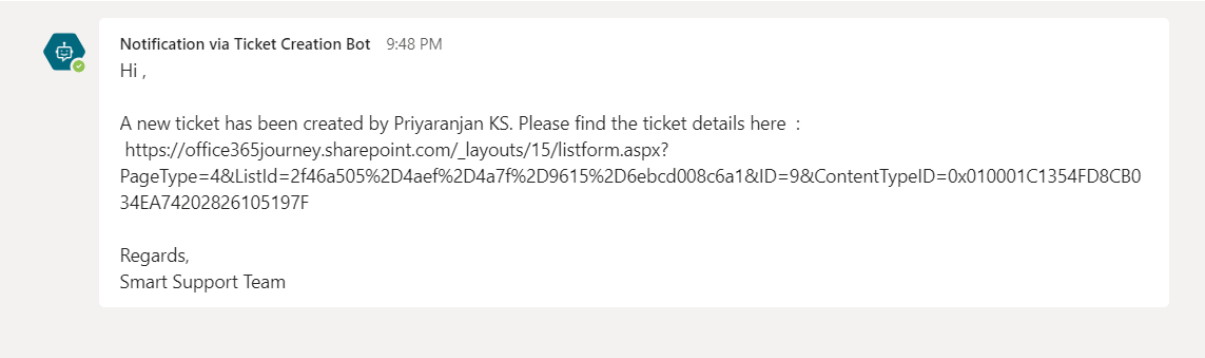
Thus, the flow will intimate the user proactively via a message in the Teams which would appear as sent by the bot

Test the Implementation

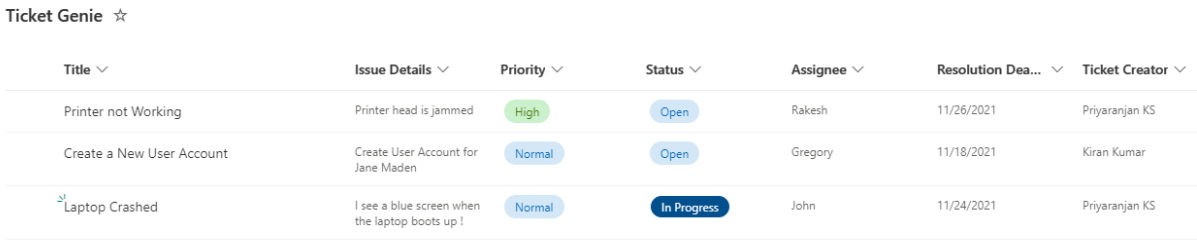
To test the implementation, lets trigger the bot with the trigger word – Create Ticket and pass on the issue details. This will trigger the Power Automate for ticket creation.



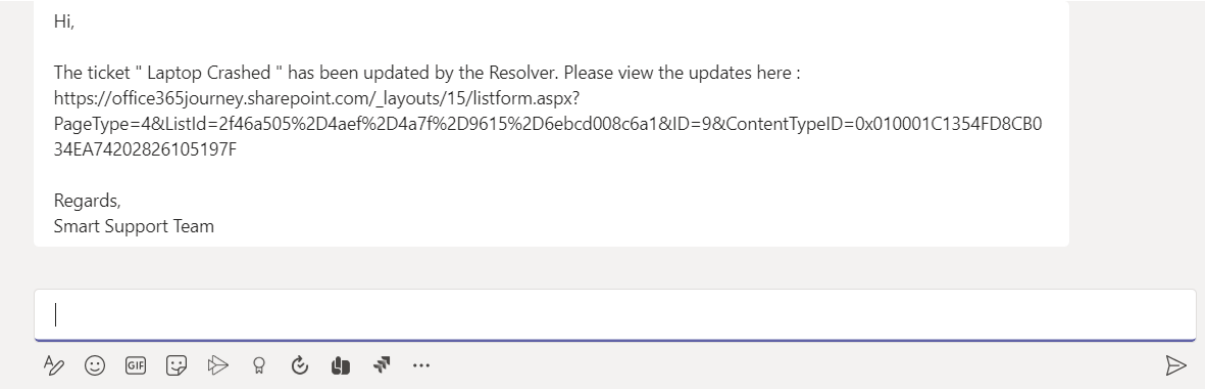
By checking the Ticket Resolver's Teams, we can see that, the ticket has been created and the ticket details as well as the ticket creator name has been sent as a proactive message by the Bot.



Same way, lets make an update to the created ticket and move it to In Progress



The stand-alone flow will be triggered, and the update will be shared as a proactive message to the ticket creator.



Summary

Thus, we saw how to provision a Ticket Creation Bot and sent out Proactive messages to the Ticket Resolver and at the same time send a Proactive message to the Ticket Creator on any updates. In the next blog, we will see how to send Adaptive cards proactively from the Bot to the User Chat.

Implement Manual Authentication in Power Virtual Agents

Introduction

Power Virtual Agents helps us create intelligent conversational chat bots without any code implementation. With these bots we can engage with customers and employees in multiple languages across websites, mobile apps, Facebook, Microsoft Teams, or any channel supported by the Azure Bot Framework

In this article we will see how to create a basic bot and use the Manual Authentication. With manual authentication we can configure any Azure AD, Azure AD V2, or OAuth compatible identity provider and use it with any channel listed in the PVA channels list.

Once Manual Authentication is configured, the following variables will be available in the authoring canvas:

- UserID
- UserDisplayName
- AuthToken
- IsLoggedIn

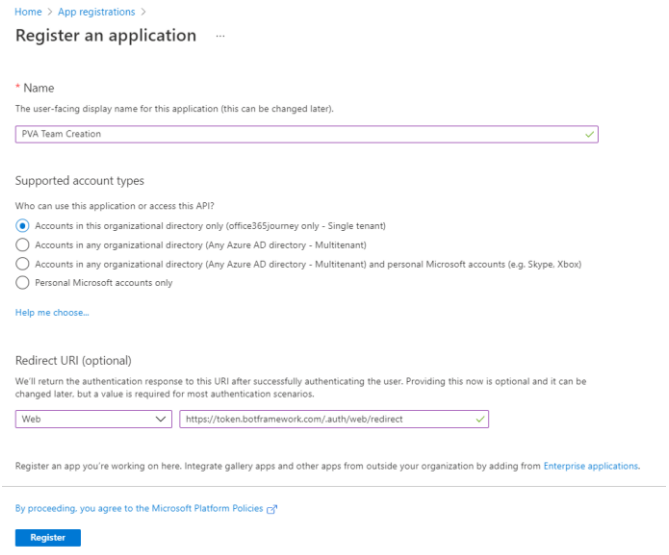
Business Use Case

We will try to implement a Teams creation bot that will take in the needed teams creation parameters from the user using PVA bot and invoke a power automate to call Graph API and use the authentication token created during the login sequence of the PVA to authorize the team creation action.

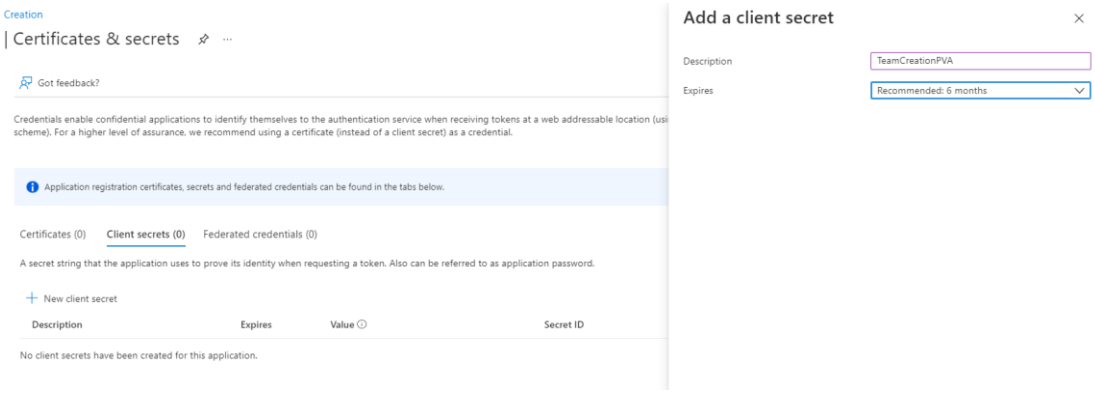
Register Azure AD Application

To use Manual Authentication in Power Virtual Agents, we have to register an application in Azure AD, create the client secret and use in in the authentication section within Power Virtual Agent.So lets head over to Azure Portal and register the application by clicking on New Registration . Specify the name of the app and add the redirect URL as https://token.botframework.com/.auth/web/redirect

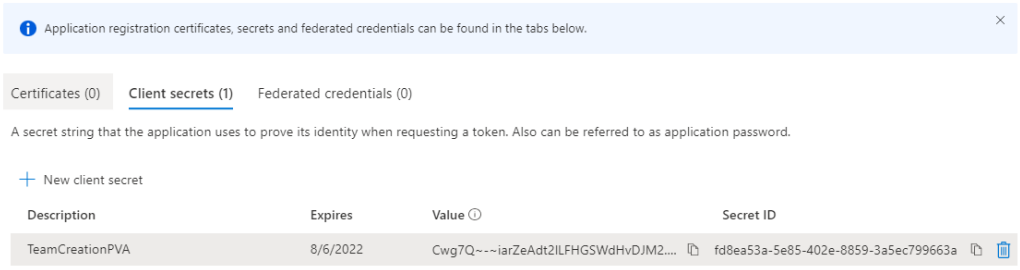
Click on Register.



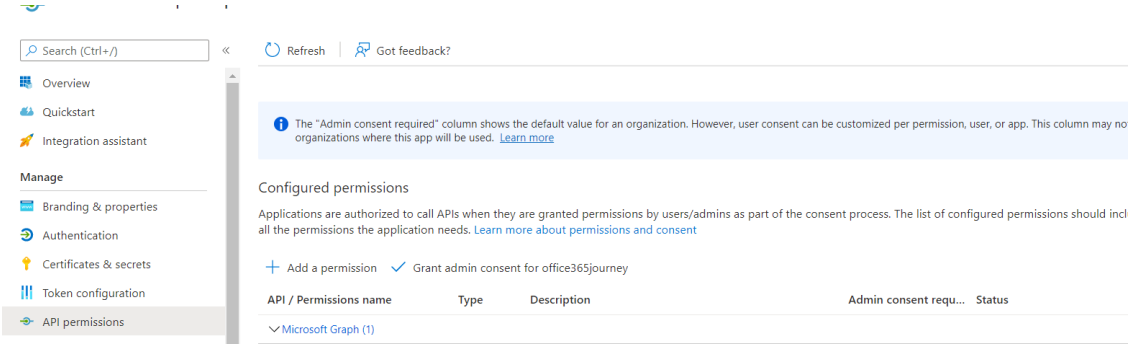
Once the App is registered, from the left pane select Certificates and Secrets. Click on New client secret and specify the Expiry window as well.



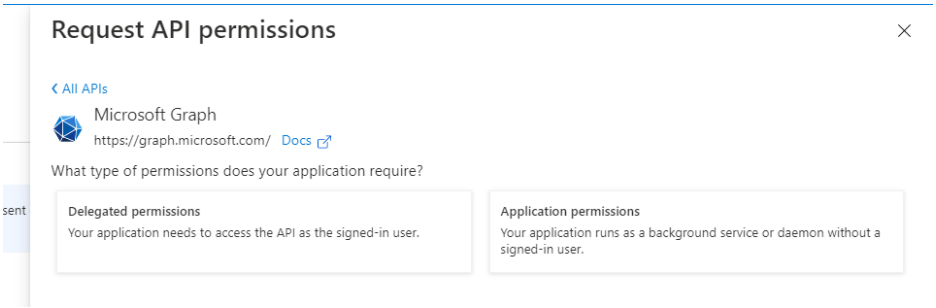
Copy the client secret as it will be hashed once we move out of the window.



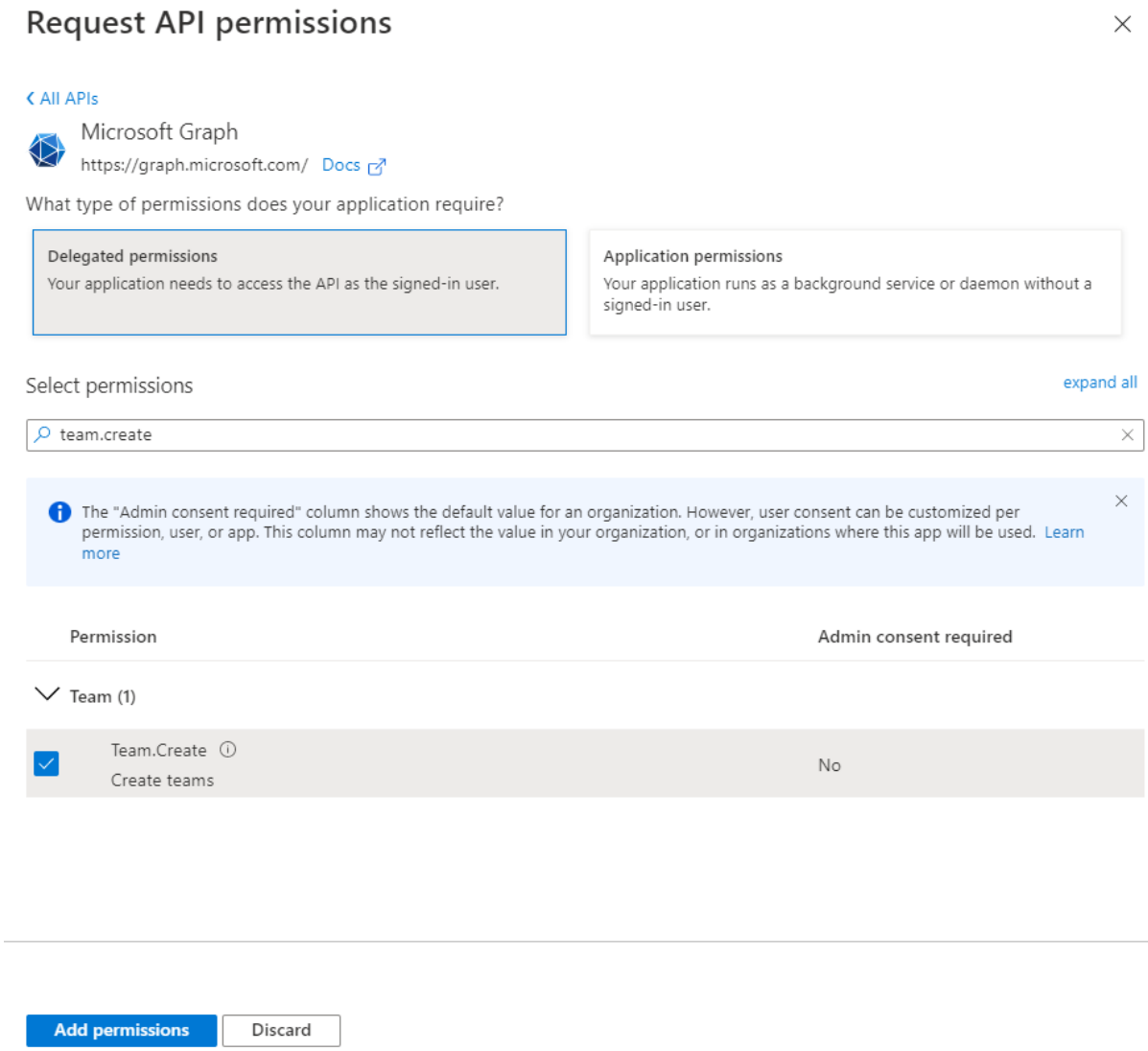
Now we also need to set the permissions for the application because we will be using the application to authenticate to Graph to create a team. From API permissions, click on Add a Permission



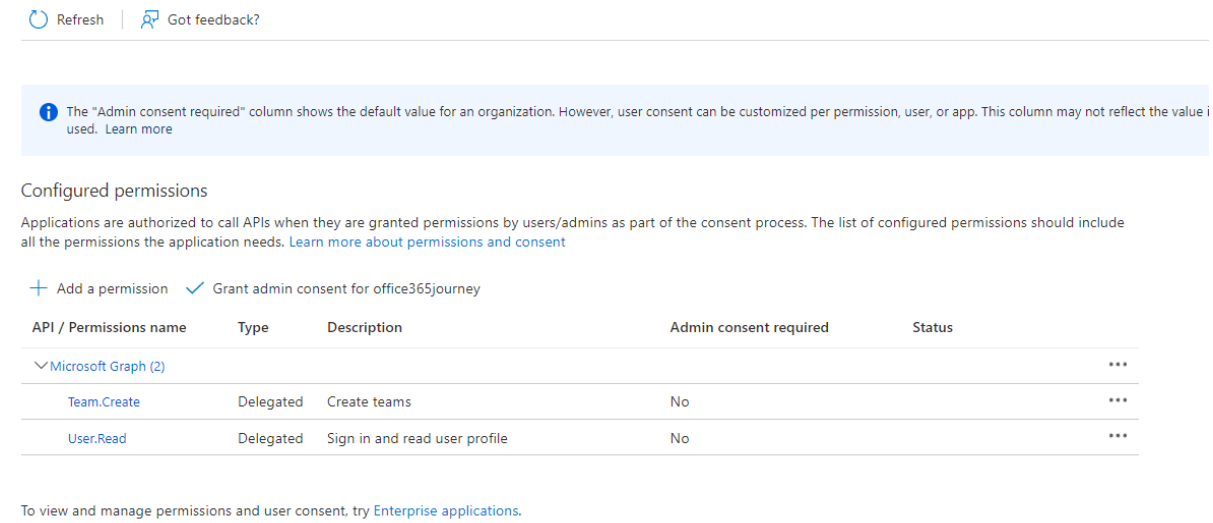
This will open the Right Pane which will list Delegated and Application Permissions option. Since we want to run the Team creation on behalf of the Application(Power Automate) we will select Application permissions in this case.



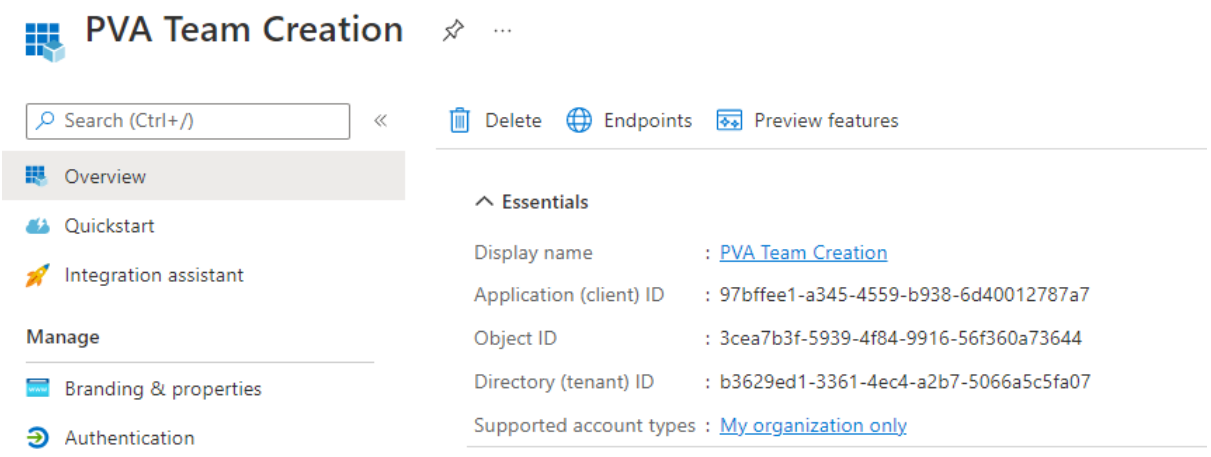
Search for Team.Create Permission and click on Add Permission



This will add the permission to the Application

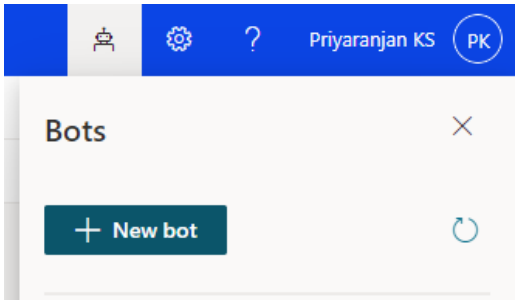


From the Overview tab, select the Application and Directory ID as we will be using it inside PVA.



Implementation

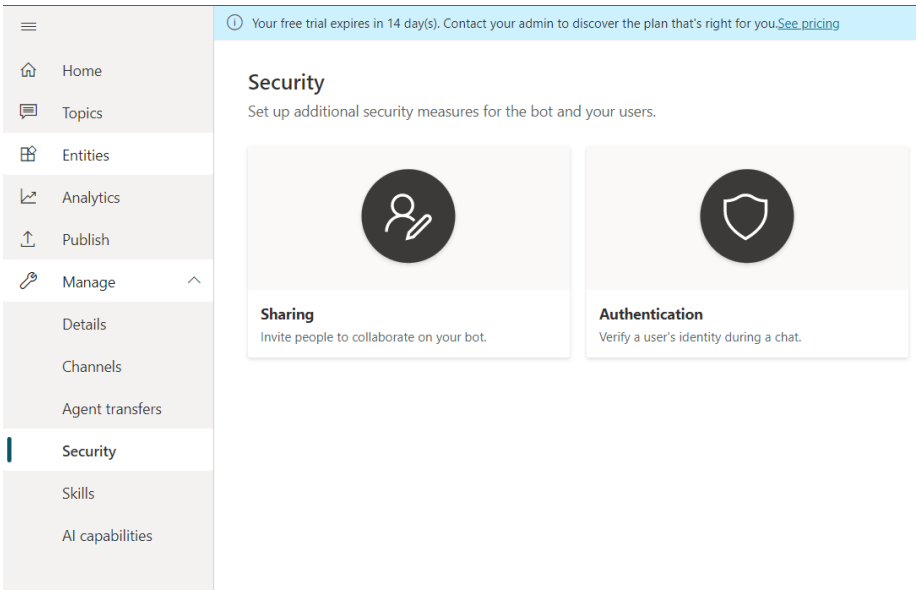
To implement the Automated Enquiry system, we will use Power Virtual Agents where we can create a bot to attend to the queries and give automated responses based on the information we have in the system. To start with we will head over to https://web.powerva.microsoft.com/ and click on the Bot symbol on the top right corner which lets you create a new bot



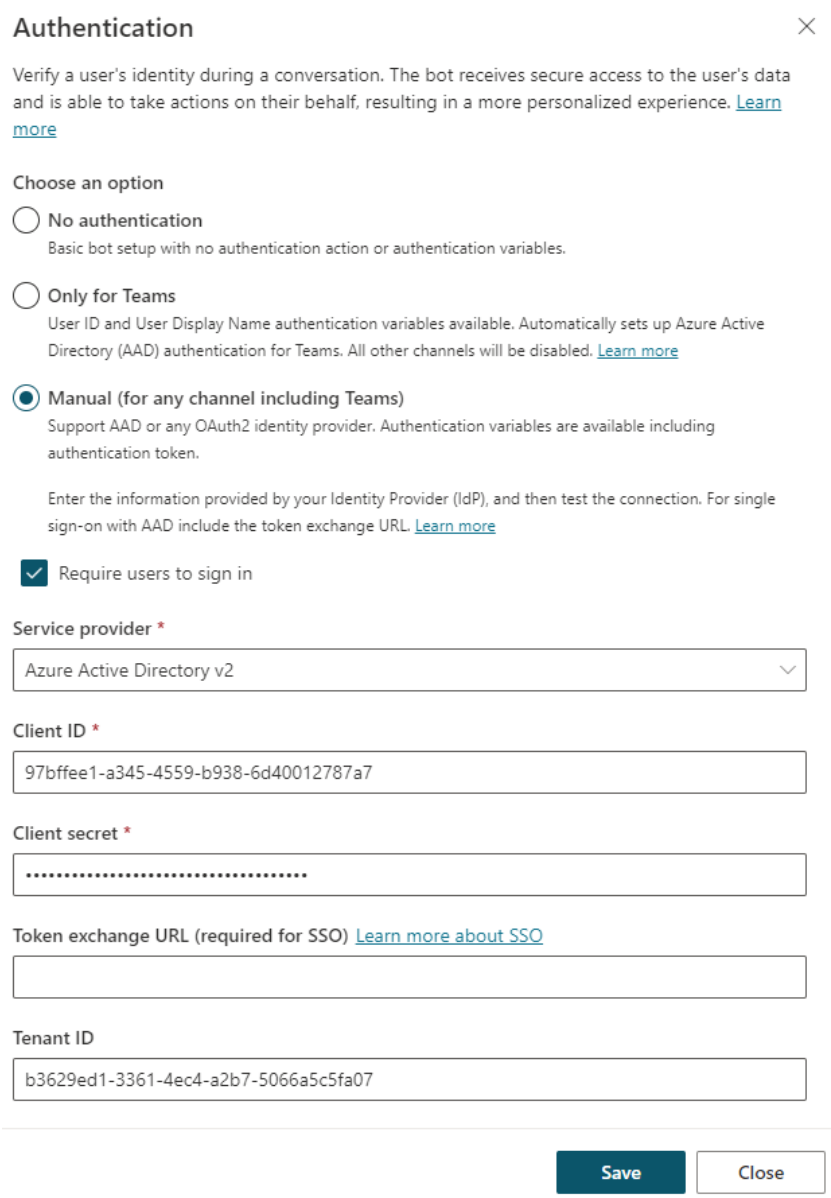
Specify the name and language for use in the bot as well as the environment where it should be provisioned. It will create a basic bot where we can add the customizations needed for our requirement.

Add Manual Authentication Details

Once the bot is provisioned, head over to Manage -> Security -> Authentication



Select manual authentication and add the Client ID, Secret and the Tenant ID of the recently created application and click on Save.

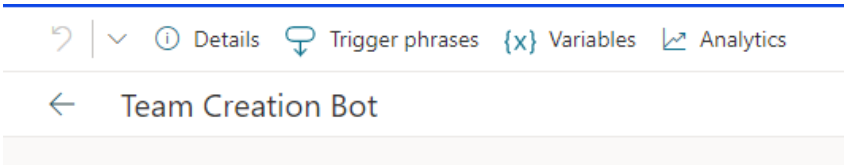


Thus, we have completed the setting up of Manual Authentication for the bot. Now lets create a topic to get started with the building of the bot.

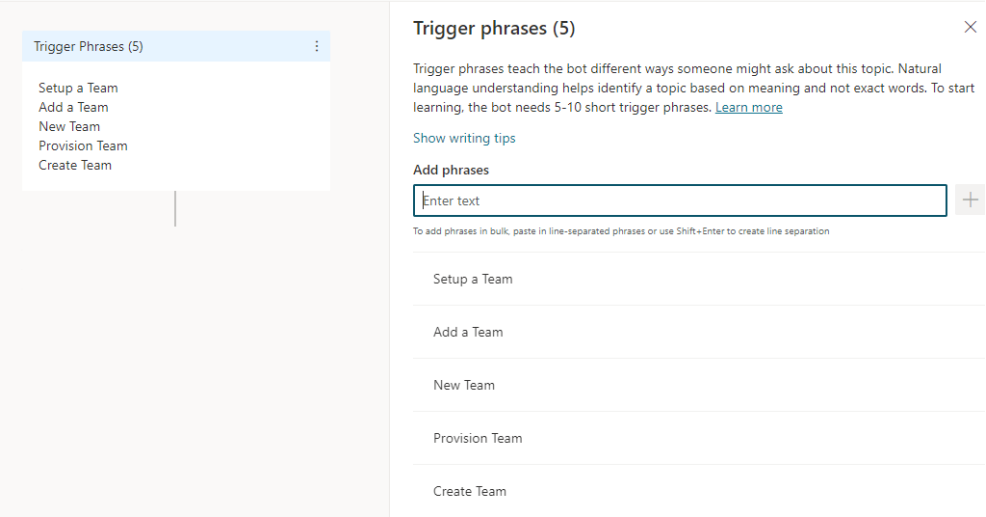
Create Topics

We can see the section Topics in the left pane which by default lists multiple conversation topic listed out. A topic defines how a bot conversation will be initiated and how the bot would respond to user interactions.

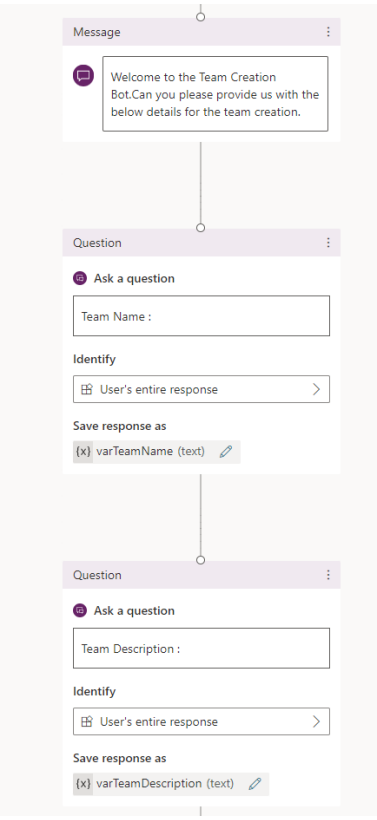
As the first step, lets create a topic that will define a general conversation starter which will pick information about the user. We can have multiple topics created in a bot and based on the way conversation is proceeding, we can navigate between multiple topics. Click on New Topic and Select From blank to create a Topic and we will name it Team Creation Bot.



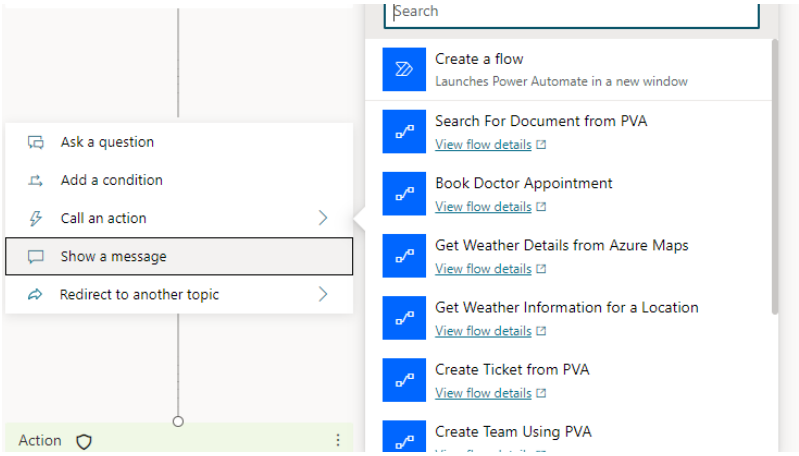
To trigger the topic, so that the control flows to that specific topic, we will define few words called Trigger Phrases that will transfer the control to these topics. The trigger phrases can be single keywords or a group of words and to cover a broad spectrum of possible trigger conditions, it is good to mention 5-10 different and still related phrases. Click on Trigger phrases that will open a right pane where we can add the trigger words.



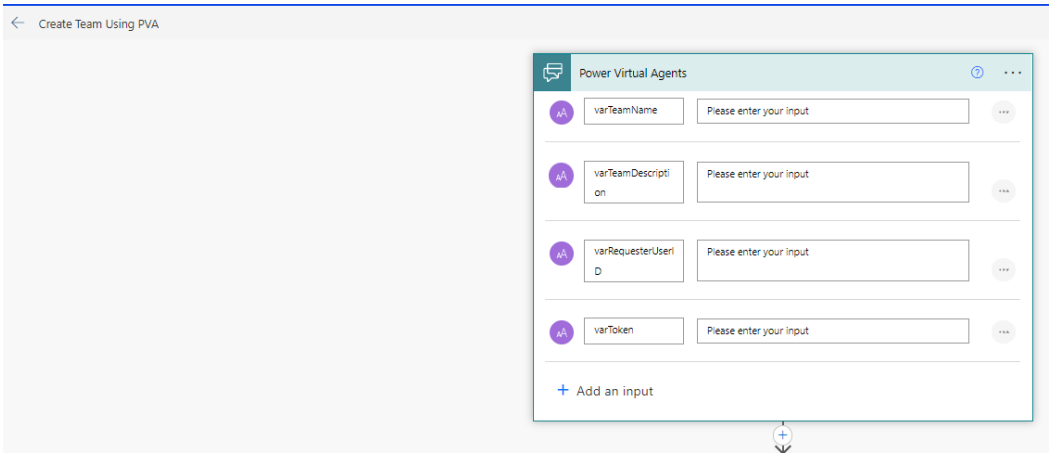
As the next step, lets add a message box with welcome message followed by 2 questions that will take input from the user regarding the Team name and description.



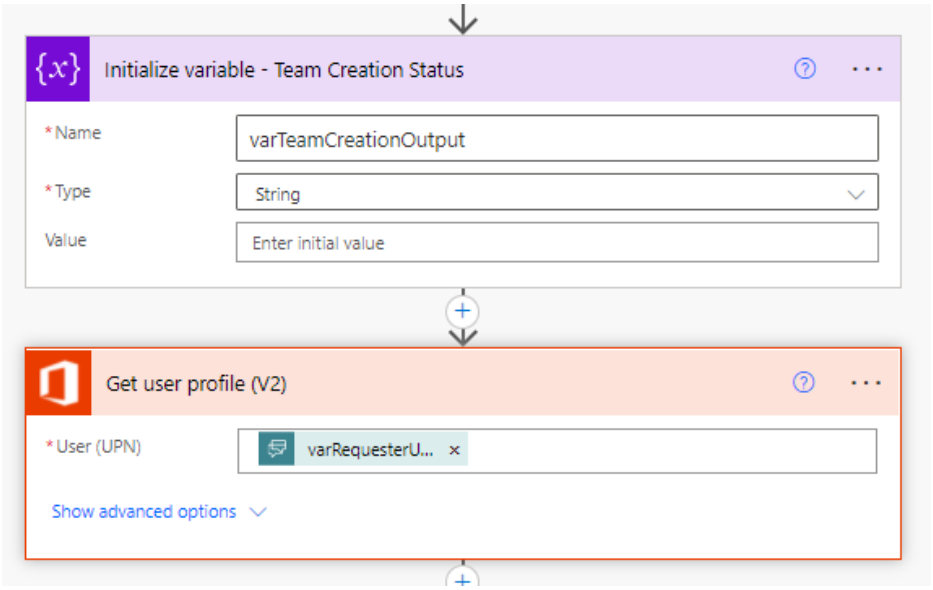
Click on Call an action node and select create a flow to design the power automate that will provision the team



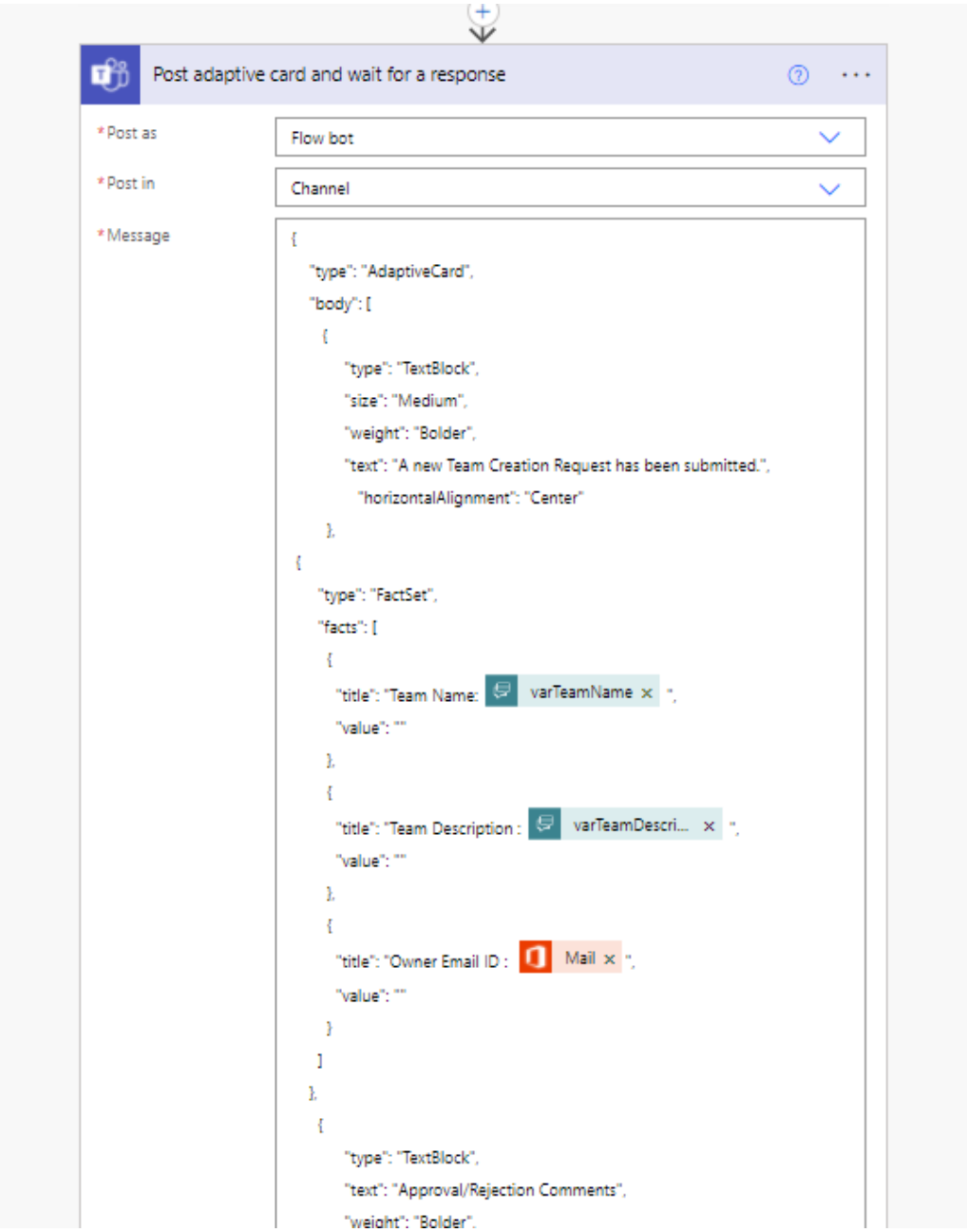
This will open Power Automate Designer in new tab where we will define 4 text inputs to get the values from PVA for Team Name, Team Description, Requester UserId and Authentication Token



Let declare a variable to hold the output that will be passed back to PVA at the end of the flow. Followed by that we will add a Get User Profile action and pass the UserId which we get as a system variable in PVA. Using this action, we will get the Email ID to be used in Teams Creation action



As part of the Team creation process, we will need to get an approval from a channel in a Team which contains a set of approvers. This way we can identify genuine requests and reject the requests which do not have a business justification



For more details on building adaptive card and its syntax you can refer to the articles

-Send an Adaptive Card to Teams using Power Automate

- Working with Adaptive Cards to send Scheduled Power Automate Notifications to Microsoft Teams

- Adaptive Card: Fetch and Display User Profile Image

The JSON used in the adaptive card has been designed using the adaptive card designer.The body of the adaptive card uses the below JSON:

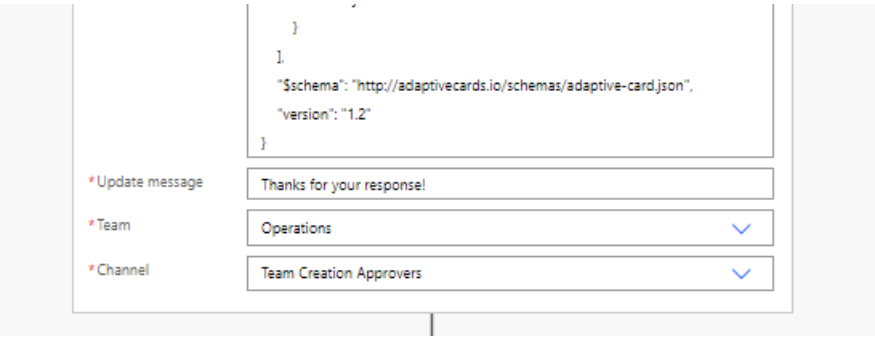
*Message


```
{
  "type": "AdaptiveCard",
  "body": [
    {
      "type": "TextBlock",
      "size": "Medium",
      "weight": "Bolder",
      "text": "A new Team Creation Request has been submitted.",
        "horizontalAlignment": "Center"
    },
{
  "type": "FactSet",
  "facts": [
    {
      "title": "Team Name: varTeamName ×",
      "value": ""
    },
    {
      "title": "Team Description : varTeamDescri... ×",
      "value": ""
    },
    {
      "title": "Owner Email ID : Mail ×",
      "value": ""
    }
  ]
},
    {
      "type": "TextBlock",
      "text": "Approval/Rejection Comments",
      "weight": "Bolder",
      "wrap": true
    },
    {
      "type": "Input.Text",
"id": "commentsID",
      "placeholder": "Enter the Approval/Rejection Comments"
    }
  ],
 "actions": [
  {
      "type": "Action.Submit",
      "title": "Approve",
     "id": "approveID"
    },
 {
      "type": "Action.Submit",
      "title": "Reject",
     "id": "rejectID"
    }
  ],
  "$schema": "http://adaptivecards.io/schemas/adaptive-card.json",
  "version": "1.2"
}
```

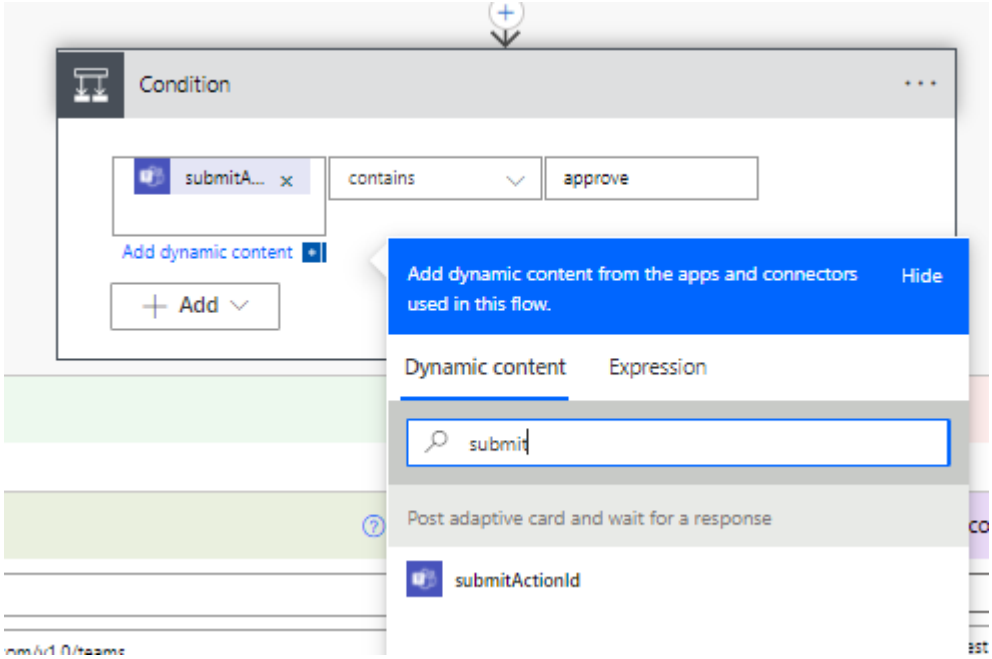
```
{
  "type": "AdaptiveCard",
  "body": [
    {
      "type": "TextBlock",
      "size": "Medium",
      "weight": "Bolder",
      "text": "A new Team Creation Request has been submitted.",
        "horizontalAlignment": "Center"
    },
{
    "type": "FactSet",
    "facts": [
     {
      "title": "Team Name: @{triggerBody()['text']} ",
      "value": ""
     },
     {
      "title": "Team Description : @{triggerBody()['text_1']} ",
      "value": ""
     },
     {
      "title": "Owner Email ID :  @{outputs('Get_user_profile_(V2)')?['body/mail']}",
      "value": ""
     }
    ]
  },
    {
      "type": "TextBlock",
      "text": "Approval/Rejection Comments",
      "weight": "Bolder",
```

```
            "wrap": true
        },
        {
            "type": "Input.Text",
"id": "commentsID",
            "placeholder": "Enter the Approval/Rejection Comments"
        }
    ],
 "actions": [
    {
            "type": "Action.Submit",
            "title": "Approve",
           "id": "approveID"
        },
 {
            "type": "Action.Submit",
            "title": "Reject",
           "id": "rejectID"
        }
    ],
    "$schema": "http://adaptivecards.io/schemas/adaptive-card.json",
    "version": "1.2"
}
```


It would be posted in the Team Creation Approvers Channel in the Operations Team.



As the Adaptive card contains 2 action buttons with ID ApproveID and RejectID, clicking on either one of them will notify Power Automate of the corresponding action and we can get to know which button was clicked by checking the submitActionId. So, we will check if the submitActionId contains the text approve to understand if approve or reject button was clicked.



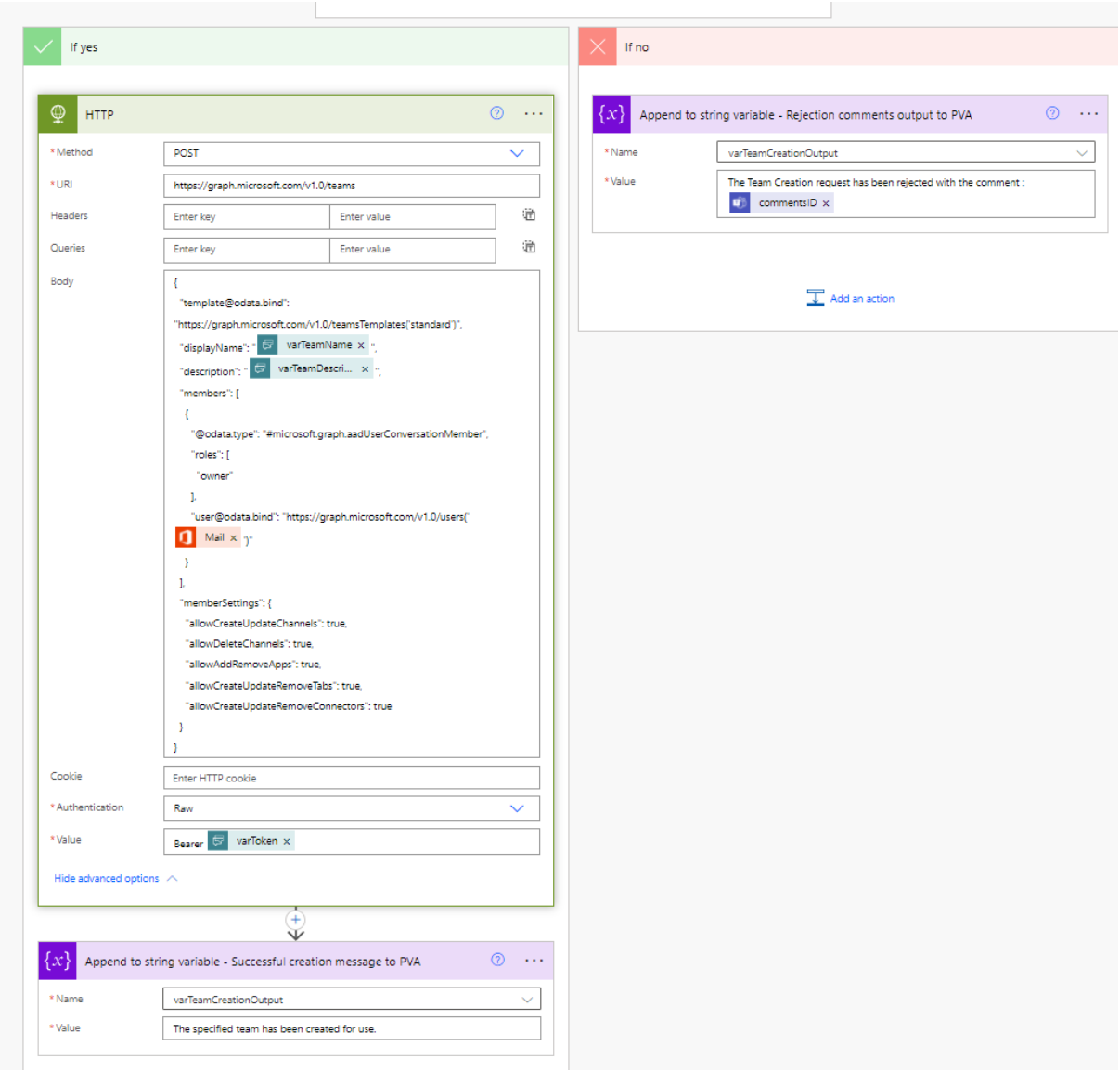
Note: The Output of Adaptive card will not be visible after the action if the message body contains some dynamic content like below. So to see submitActionId in the output, temporarily remove dynamic content from adaptive card message and after making use of submitActionId from the dynamic output, put the below variables back in position.

```
        "text": "A new Team Creation Request has been submitted.",
        "horizontalAlignment": "Center"
      },
{
  "type": "FactSet",
  "facts": [
    {
      "title": "Team Name: varTeamName ",
      "value": ""
    },
    {
      "title": "Team Description : varTeamDescri... ",
      "value": ""
    },
    {
      "title": "Owner Email ID : Mail ",
      "value": ""
    }
  ]
},
  {
```

If the approve button was clicked, we will enter the left branch and add an HTTP action to call the graph end point to URL https://graph.microsoft.com/v1.0/teams a

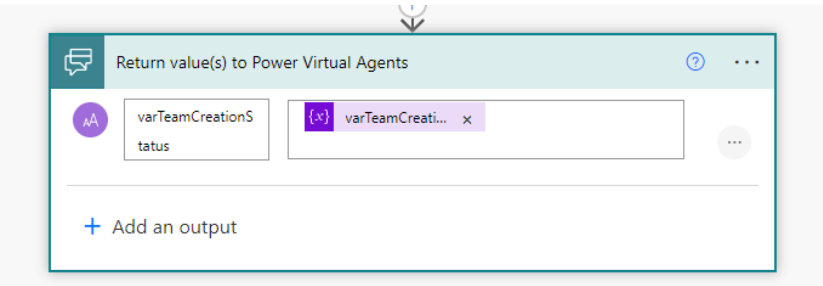
and pass the body to create the team with the use inputted team name and description. We will also use the email id we have extracted previously as the owner email id in the team creation body. We have added few team settings so that by default they will be applied on creation. To process the authentication, we will use the already authenticated token which we have passed from the PVA system variable and use that in the Value section. Post the http action, we will add the action to assign the output variable with a successful team creation message.

```
{
 "template@odata.bind": "https://graph.microsoft.com/v1.0/teamsTemplates('standard')",
 "displayName": "@{triggerBody()['text']}",
 "description": "@{triggerBody()['text_1']}",
 "members": [
  {
   "@odata.type": "#microsoft.graph.aadUserConversationMember",
   "roles": [
    "owner"
   ],
   "user@odata.bind":
"https://graph.microsoft.com/v1.0/users('@{outputs('Get_user_profile_(V2)')?['body/mail']}')"
  }
 ],
 "memberSettings": {
  "allowCreateUpdateChannels": true,
  "allowDeleteChannels": true,
  "allowAddRemoveApps": true,
  "allowCreateUpdateRemoveTabs": true,
  "allowCreateUpdateRemoveConnectors": true
 }
}
```



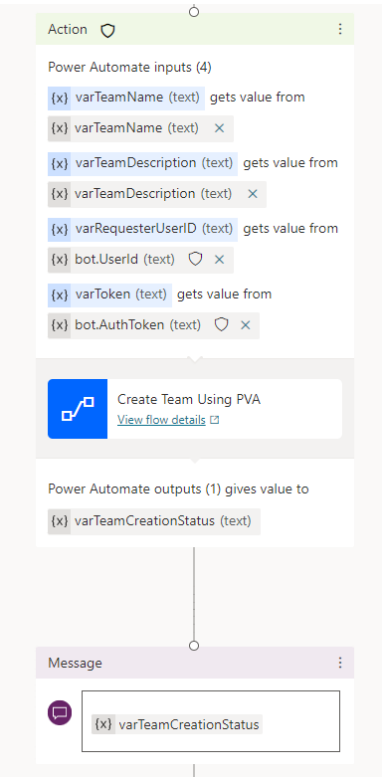
If the approver had rejected from adaptive card, we would fetch the corresponding comments from the adaptive card and add it to the output variable so that we can show it to the end user.

Finally, we will return back the output variable to PVA



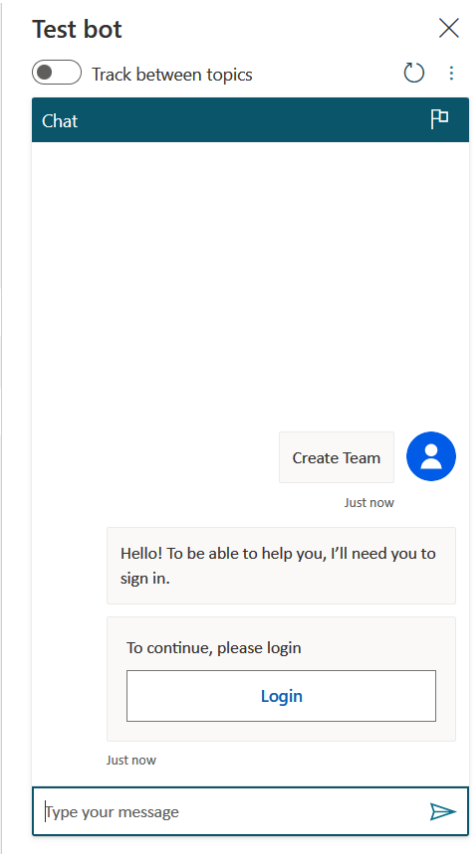
Call the Flow

Heading back to power automate, we can call the above created flow from PVA and pass the 4 inputs which will be used in the flow and finally add the message box which will show the success/failure of the team creation process.

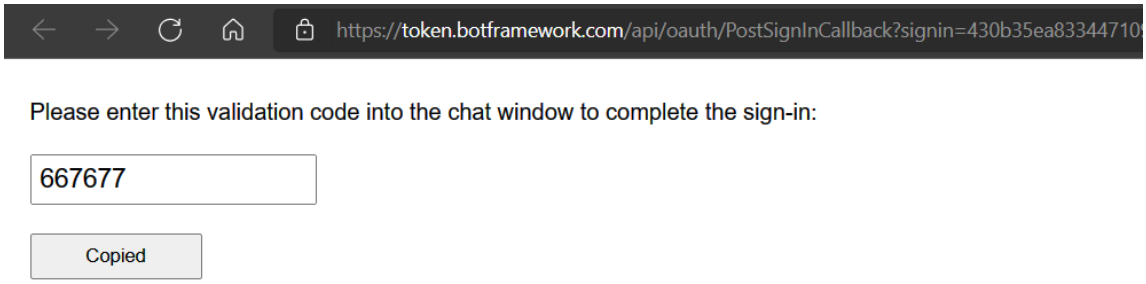


Test the bot inside Test Window

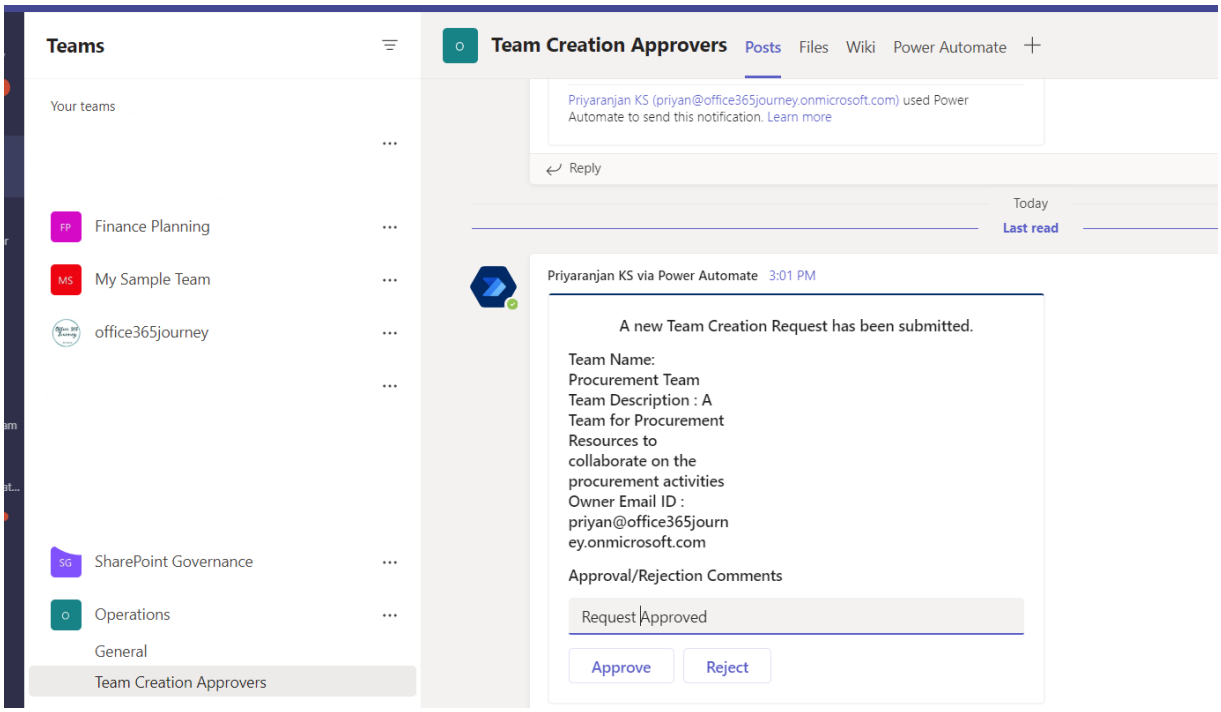
As part of testing the bot,let type in the trigger word like Create Team



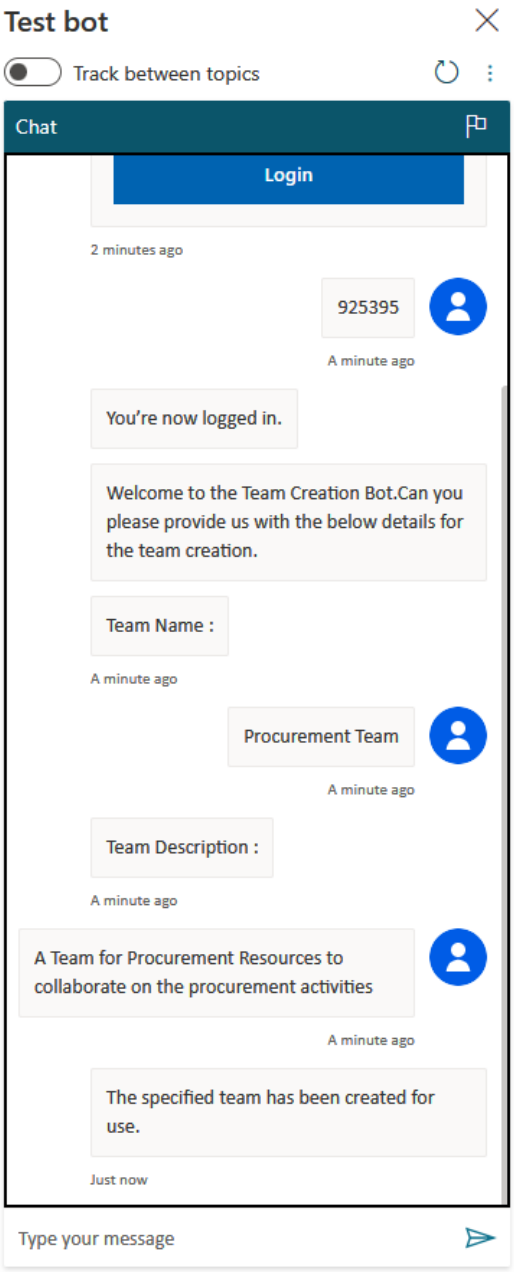
It will open a tab where we can get the verification code to complete the sign in.



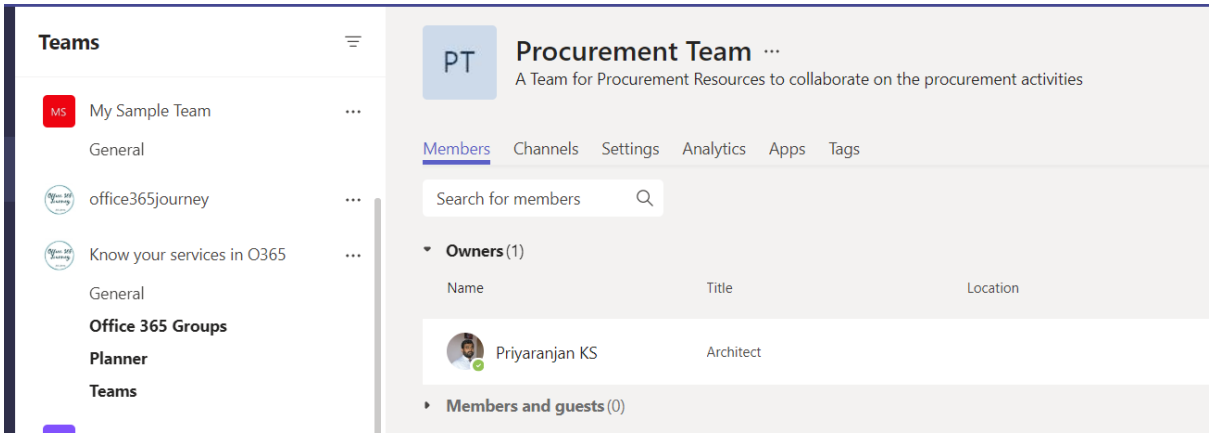
Copying and pasting the code back to the PVA chat window, will complete the authentication handshake. We can specify the Team Name and Description so that it will be used for the provisioning of teams upon the approval. The approval request has been generated as an adaptive card in Approvers Channel in Teams.



We will go ahead and approve it which will provision the team and share back the output to the test chat window.

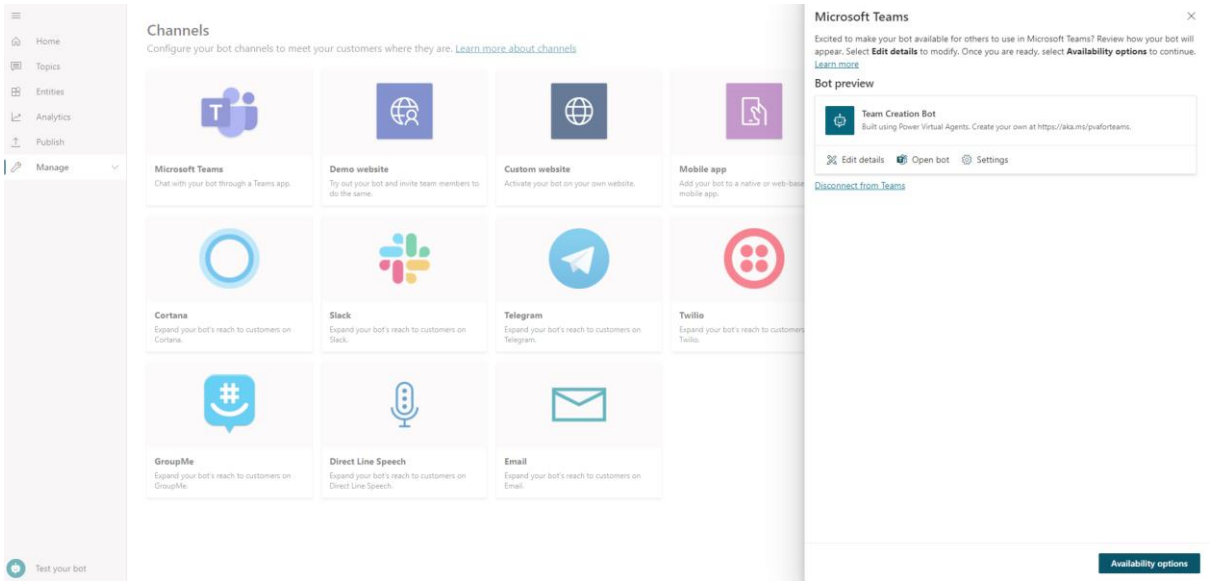


The Team is now available for collaboration in Microsoft Teams.

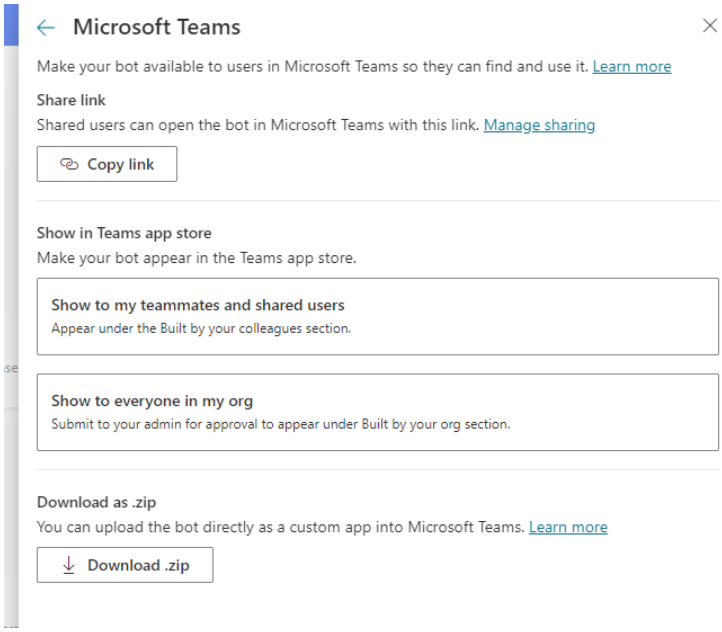


Publish the bot

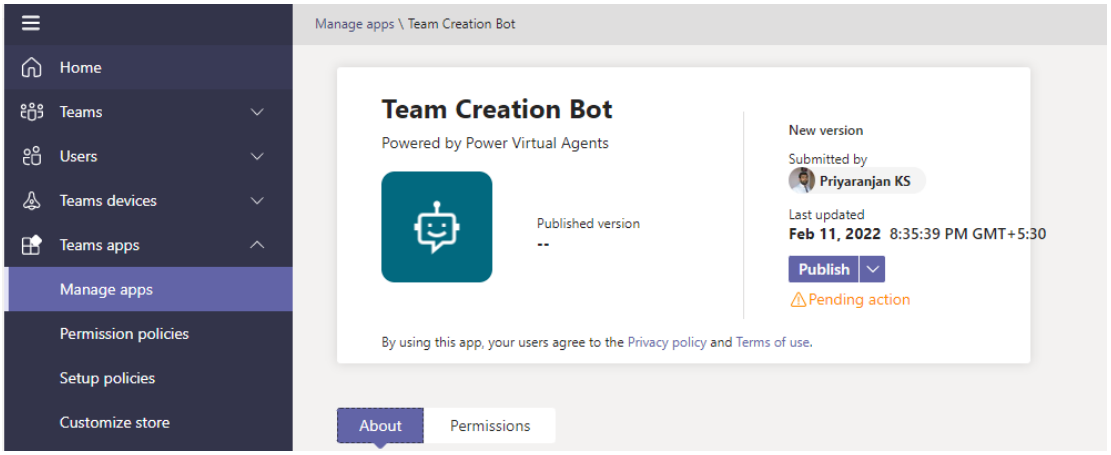
After successful testing, Let's publish the bot to teams' channel by heading over to Publish tab in the left pane. We will select Team as the preferred channel.



Clicking on Availability options will open the pane to publish the Bot for admin approval so that everyone in the org can use it. Or you download the zip and upload as custom app for personal use as well.

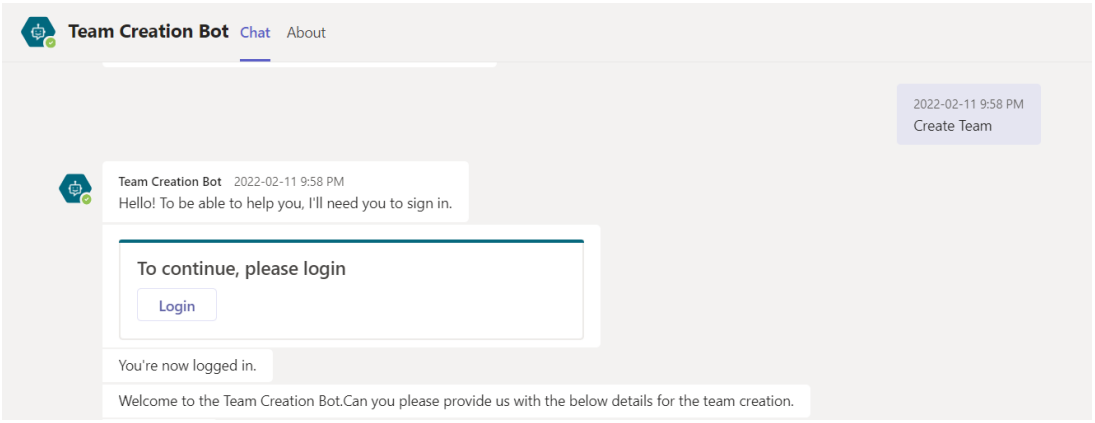


If we have selected Show to everyone in org, tha bot will appear in the Manage apps section where the admin can publish it for organizational use.

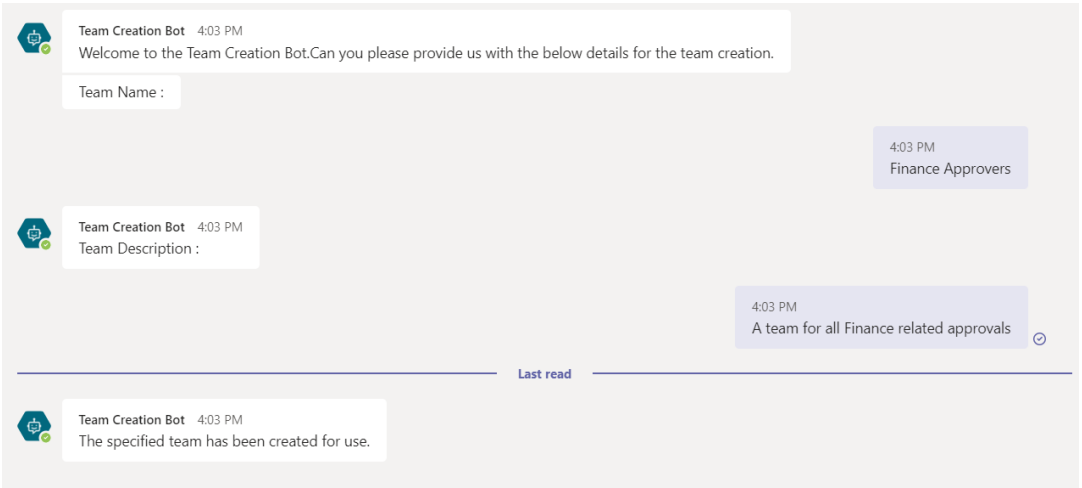


Test the bot in Teams

Once it is approved by the admin from the Teams admin centre, it will be available for use in teams. We can use the bot just the way we tested and used in the Test Windowpane in Power Virtual Agents where we would get the Authentication prompt.



Upon Authentication, we will enter the details and the Team will be provisioned and the status will be updated back in the bot.



Summary

Thus, we saw how to set up an Authentication Provider for use with Power Virtual Agent bots which will perform an authentication against the provider to continue with the Bot Conversations.

Integration with AI Builder

Introduction

With Power Virtual Agents, the use cases are limitless and based on the requirements we can keep pushing the boundaries of its abilities. By integrating Power Virtual Agents with Power Automate and AI builder we can further expand the horizons of its usage by leaps and bounds.

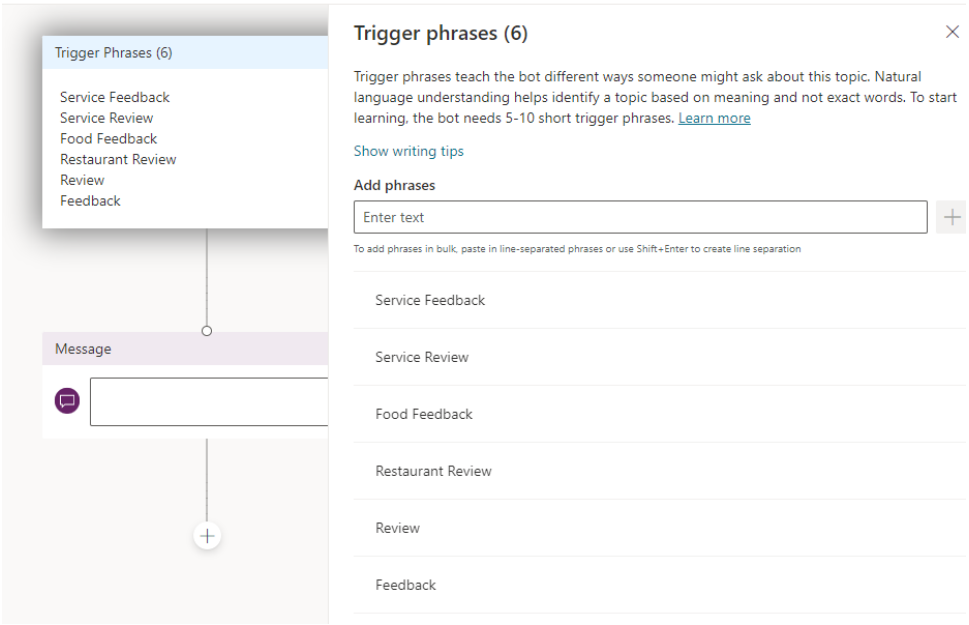
In this article we will see how to leverage AI builder and make use of Sentiment Analysis to analyse the textual feedback from the user

Scenario

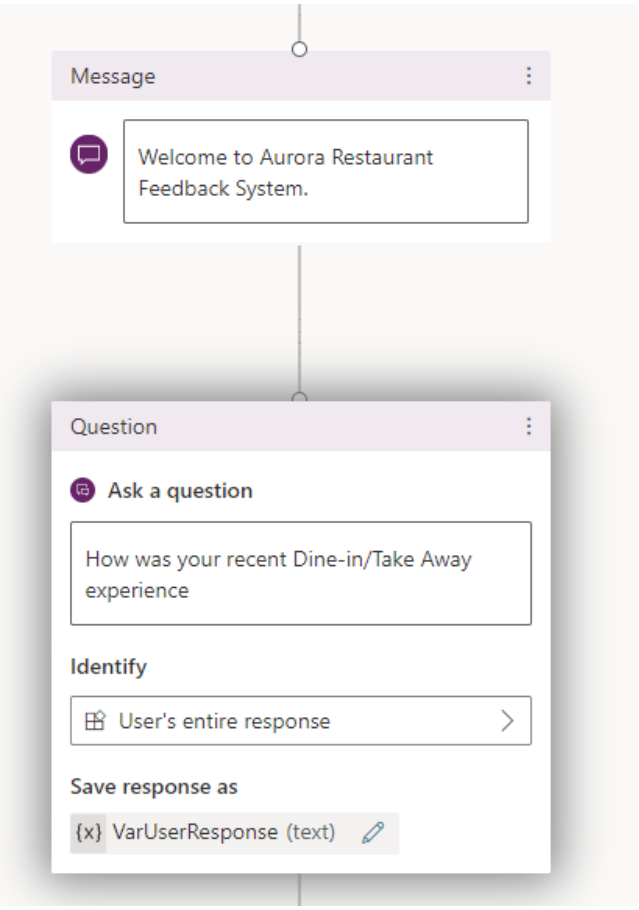
We have a requirement where we have a Restaurant app that accepts the feedback from end user about the food and services. We will expand the basic chat implementation to accept the feedback and do a sentiment analysis using AI builder to categorize the negative feedbacks so that instantaneous actions can be taken on those feedbacks by the management.

Implementation

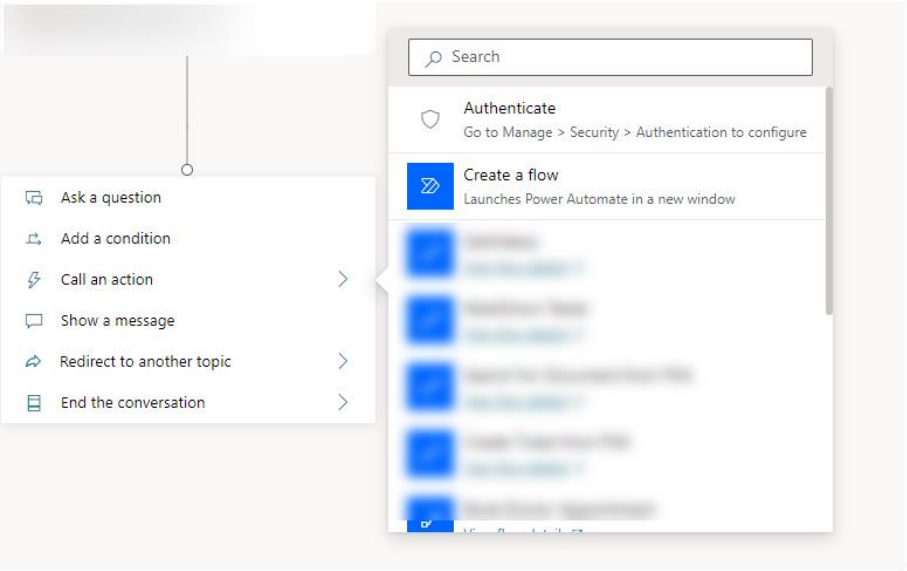
We will create a SharePoint List with Title and Feedback Sentiment columns to house the details from the user conversation. Let's create a basic bot and create a new blank topic to which we will add few trigger words.



We will then greet the user and ask for the feedback so that we can capture it for further analysis

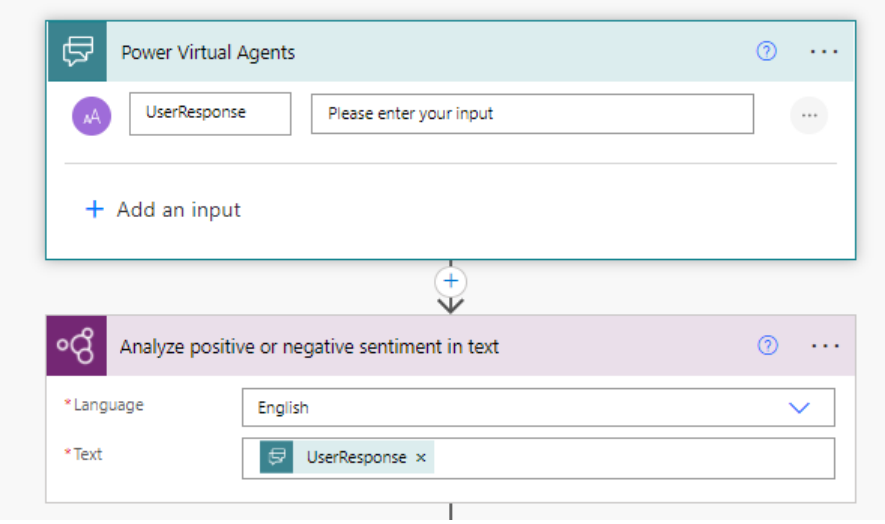


Now, lets create a Power Automate by clicking on Call an Action which will open the Power Automate window.

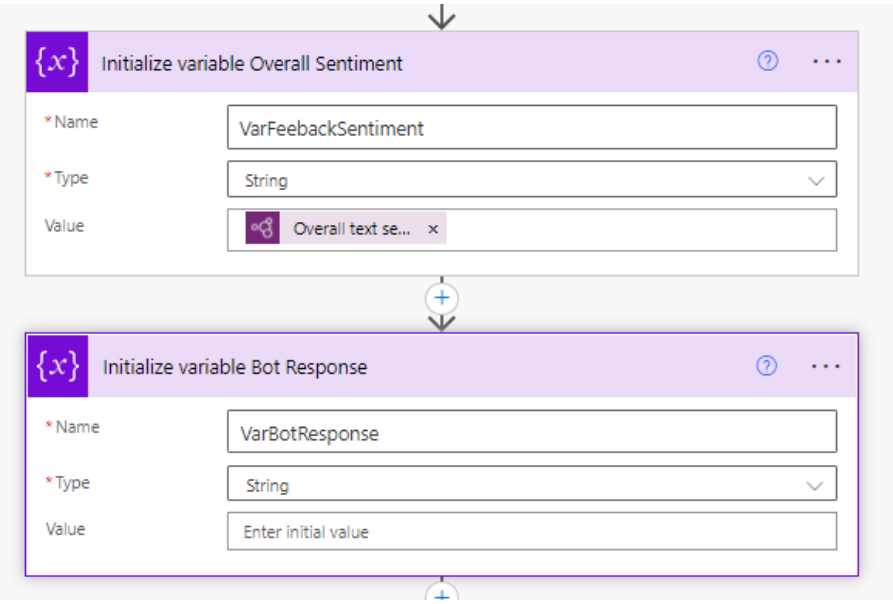


Create Power Automate

Let's add an input parameter in the Power Automate so that we can accept the user response for analysis. Followed by that, we will add the AI Builder action – Analyse positive or negative sentiment in text to which we will pass the user feedback as the input.

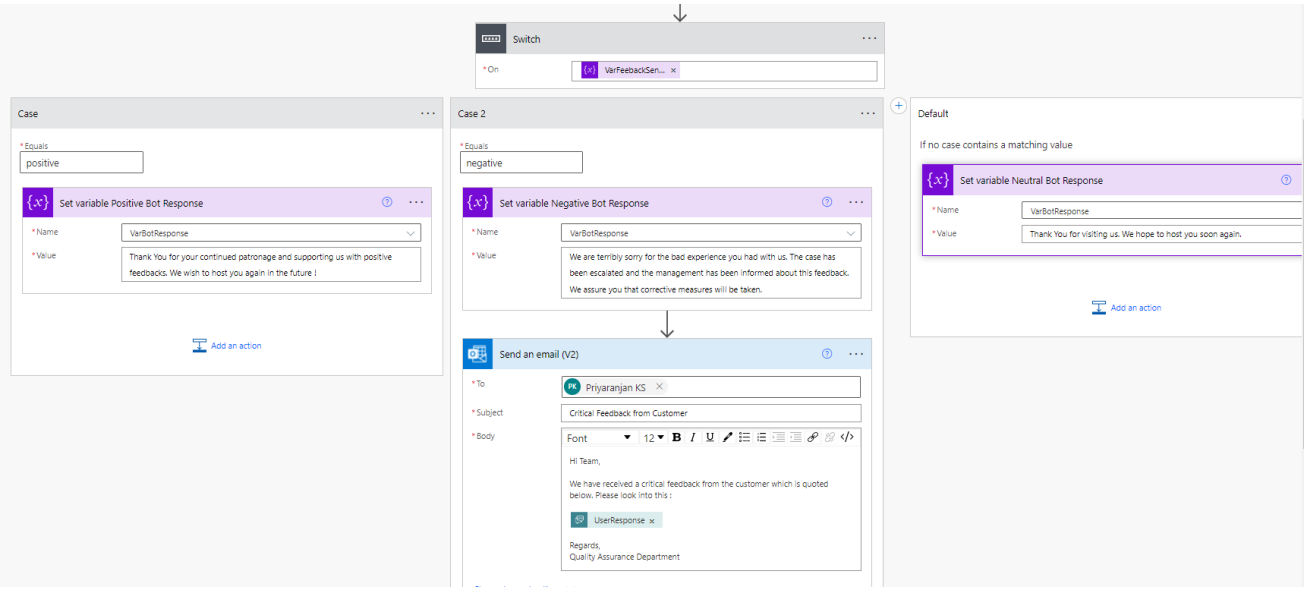


We will declare 2 variables, one to store the overall sentiment of the feedback and the next to store the response for the respective sentiment.

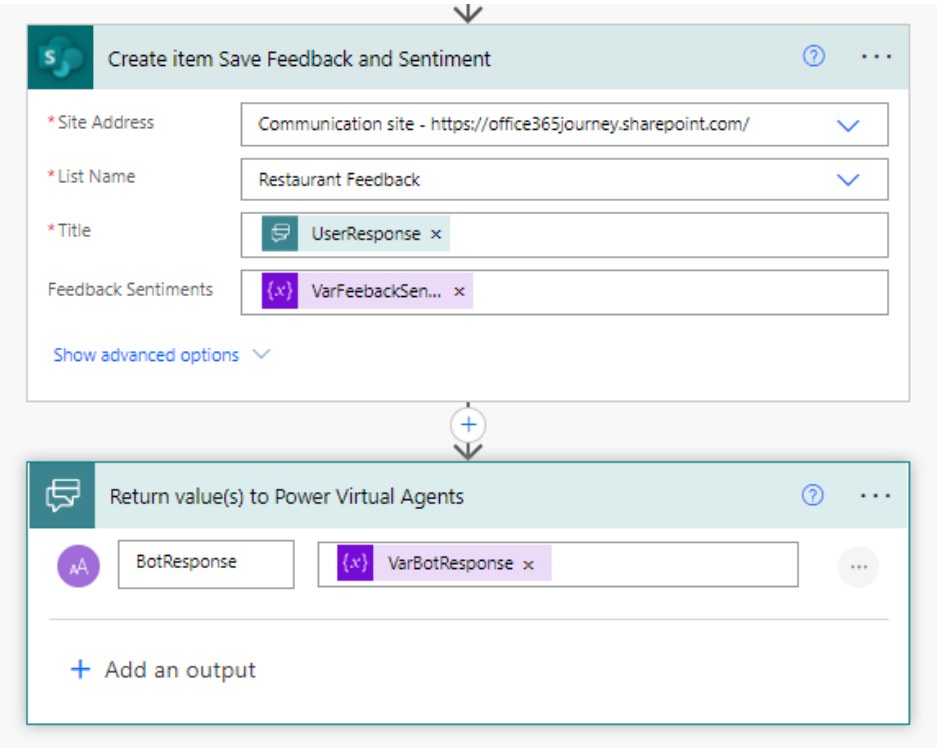


We will then add a Switch statement On the Feedback sentiment variable and add 3 case branches

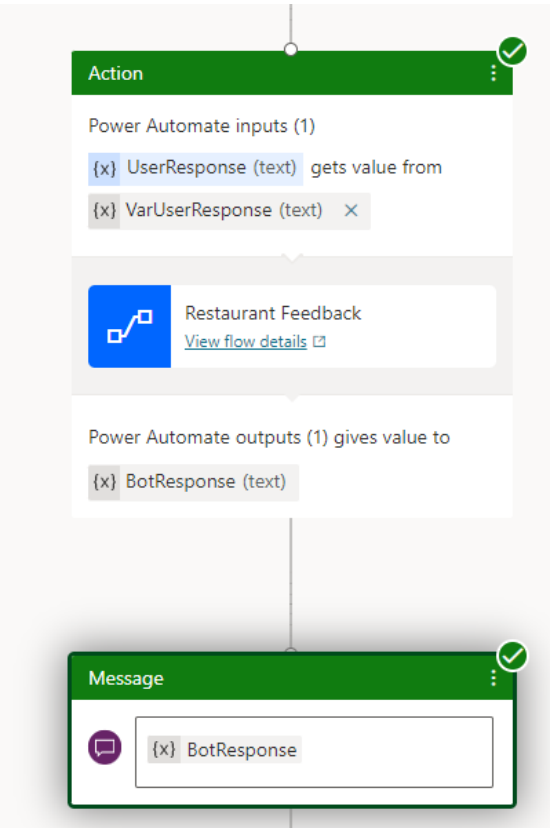
In the first branch, if the feedback is positive, we will assign a Positive Thank You response that the bot will say in return. If the feedback is negative, we will add an apologetic response for the bot and sent a mail to the management escalating the issue. As the default case, the sentiment is neutral, and we share a neutral response back to the customer.



Finally, we will save the record to the SharePoint List and return the Bot Response to the Power Virtual Agent.

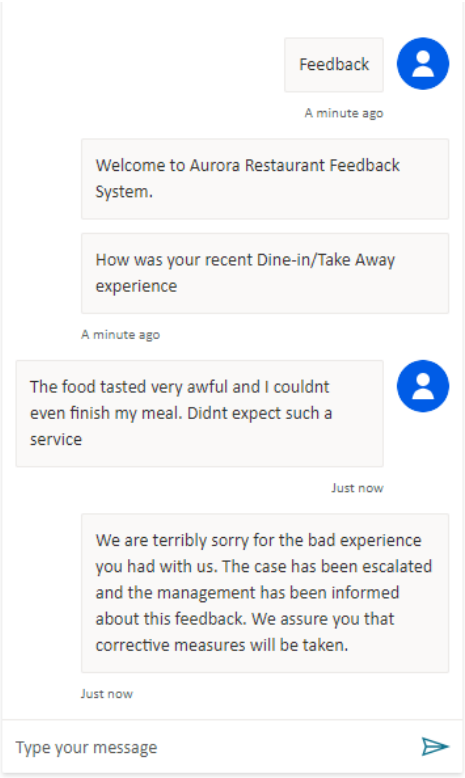


Heading back to PVA, we will add the recently created flow and pass in the User Response and get back the bot response as below:

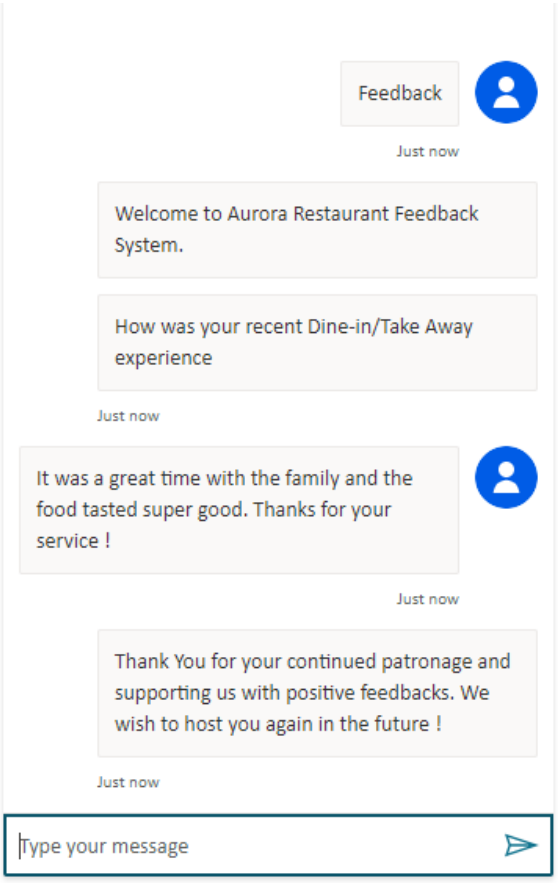


Test the Bot

Let's test the bot implementation by providing a negative feedback. The AI builder analyses the response and categorizes it as negative and shares the appropriate apologetic response back to the user



Now let’s try out a positive feedback and we receive back the excited positive response from the bot



Summary

Thus, we saw how to plug in the AI Builder with Power Virtual Agents to analyse the user conversation using sentiment analysis and respond back accordingly to the user.

Integrate Power Virtual Agents with Bot Framework Composer and Display Adaptive Cards

Introduction

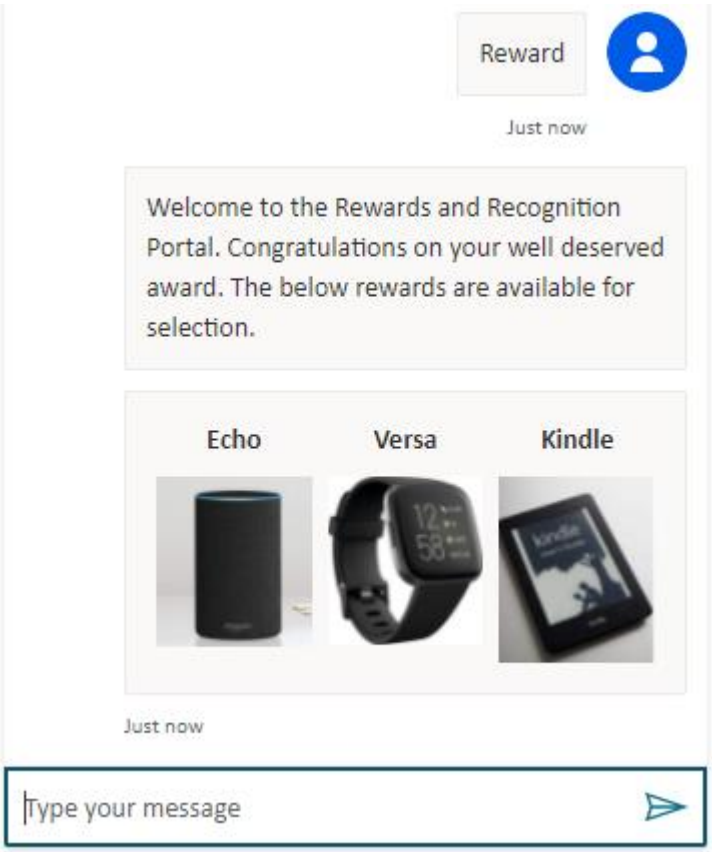
Power Virtual Agents come with expanding possibilities and at the same time limitations when it comes to using complex actions and dialogs. However we can overcome this by using Bot Framework composer and extend the user experience by adding adaptive cards bringing whole lot more possibilities to Power Virtual Agents.

The custom dialogs that is created using Bot Framework Composer are deployed together with Power Virtual Agents bot content and can be used natively within the bot creation experience of the PVA which do not require any additional Azure hosting.

Lets see how we can make use of Adaptive cards created using Bot Framework composer within Power Virtual Agents

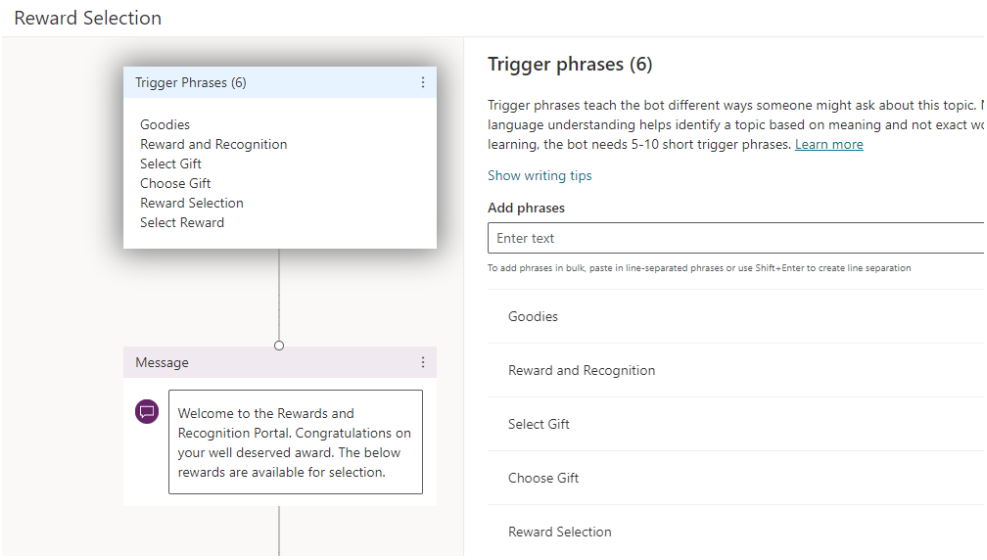
Scenario

We are creating a Rewards Selection Bot that lets users choose from a selection of the gifts. So as to show the image of the gifts to the user, we can make use of adaptive cards and create a table like representation of images as shown below:

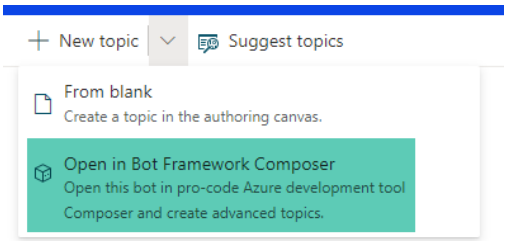


Implementation

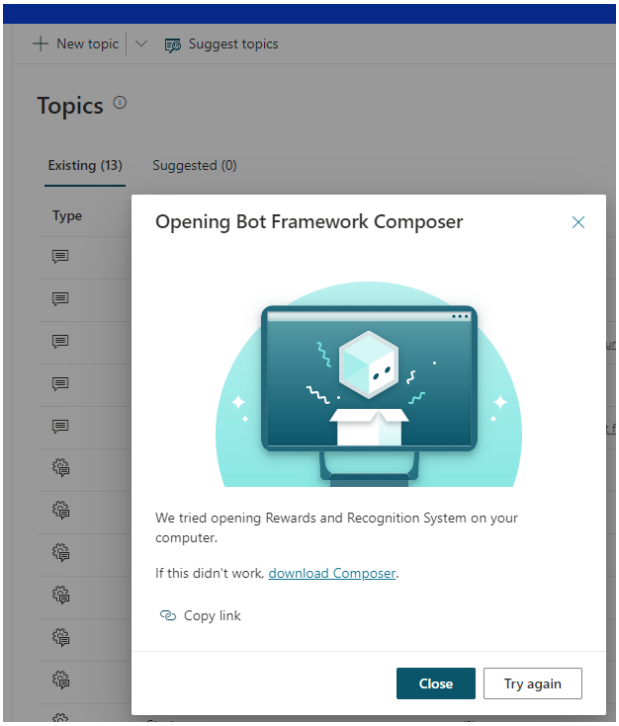
Let's get started with the Bot creation by creating a new bot and adding few trigger words that will invoke the bot conversation.



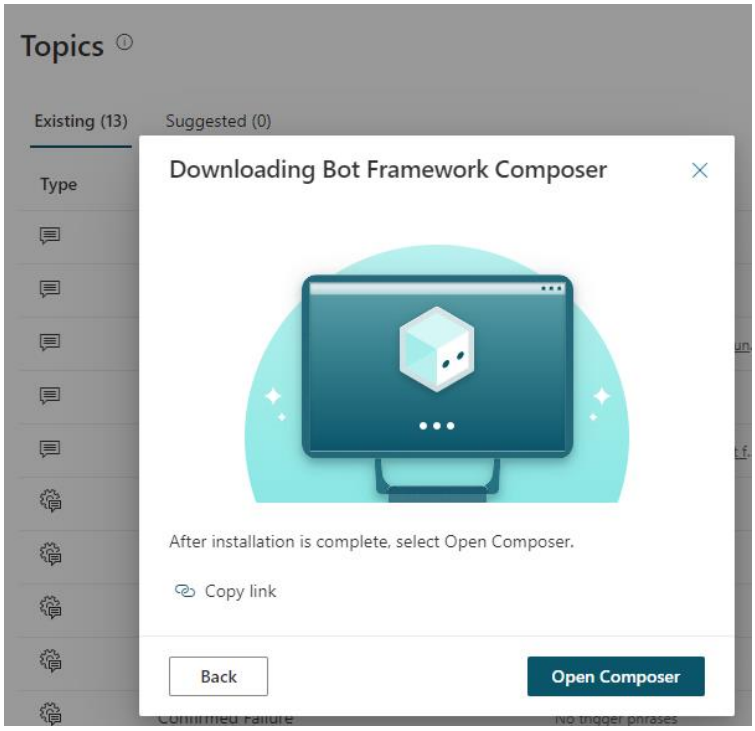
We have also added a welcome message and show the available images as an adaptive card. However, to show the Adaptive card we will create a new Topic using Bot Framework Composer and redirect the conversation to the newly created topic.



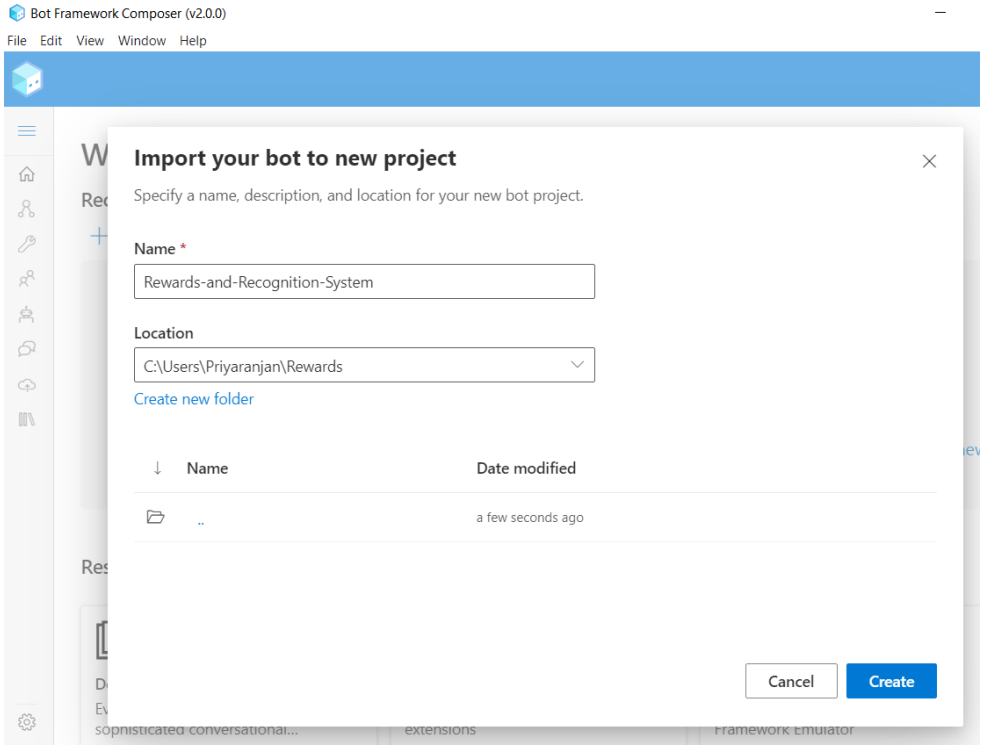
Click on Open in Bot Framework Composer, in case you are working with the composer for the first time, you will get the below prompt



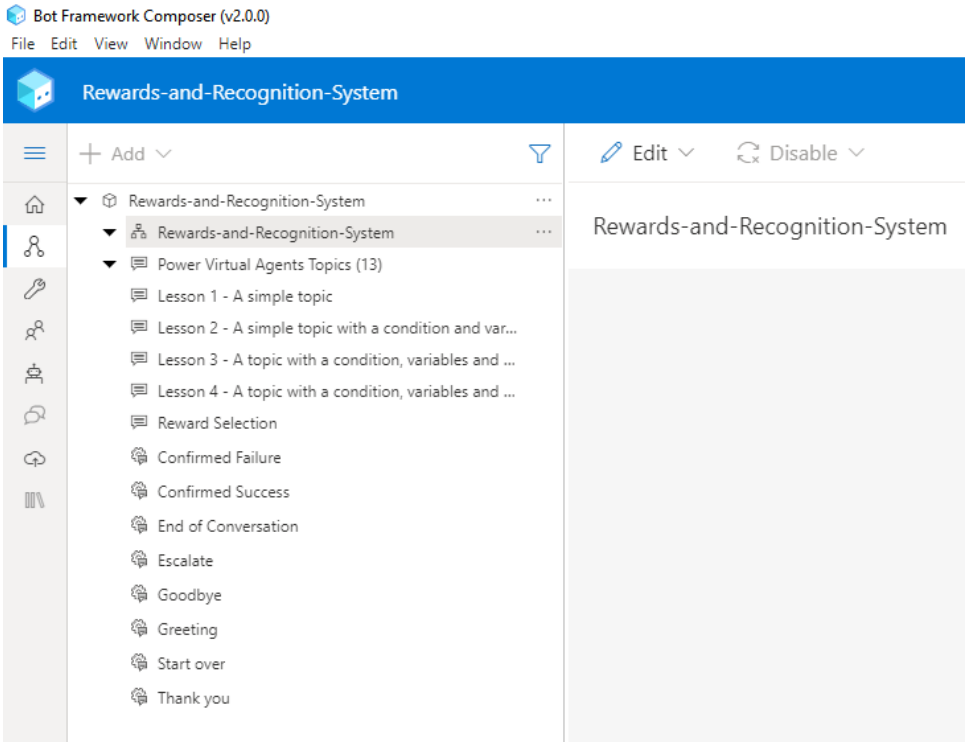
Click on download Composer to install it in the desktop



Once the installation is completed, click on Open Composer which will open the Bot Framework composer. It will let you import the existing Power Virtual Agent Bot as a new project to the Composer.

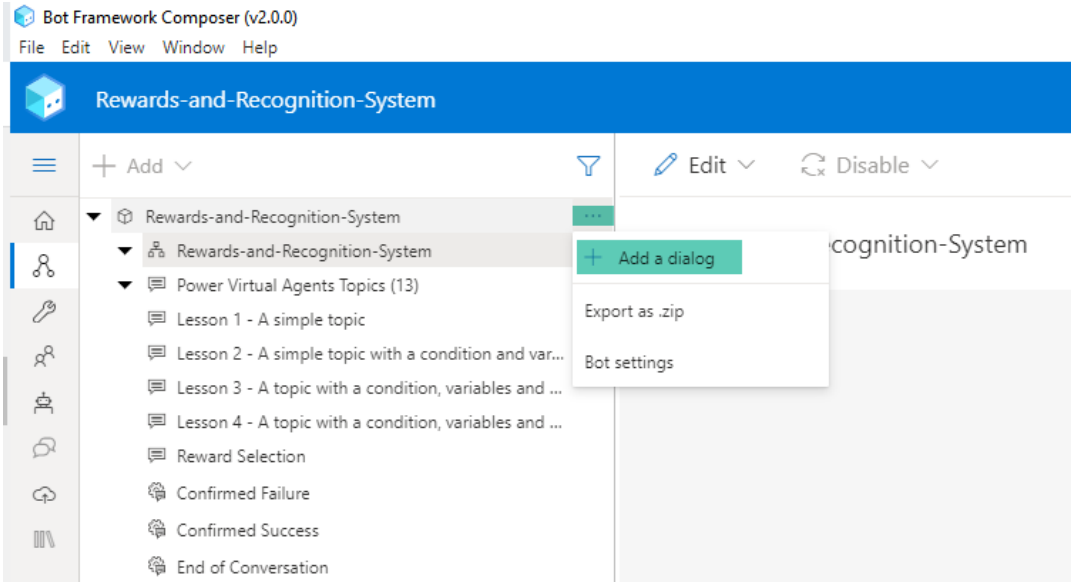


In the left pane we can see the existing topics. We will not be able to edit it from composer. Clicking on it will take us to the Power Virtual Agents for editing.

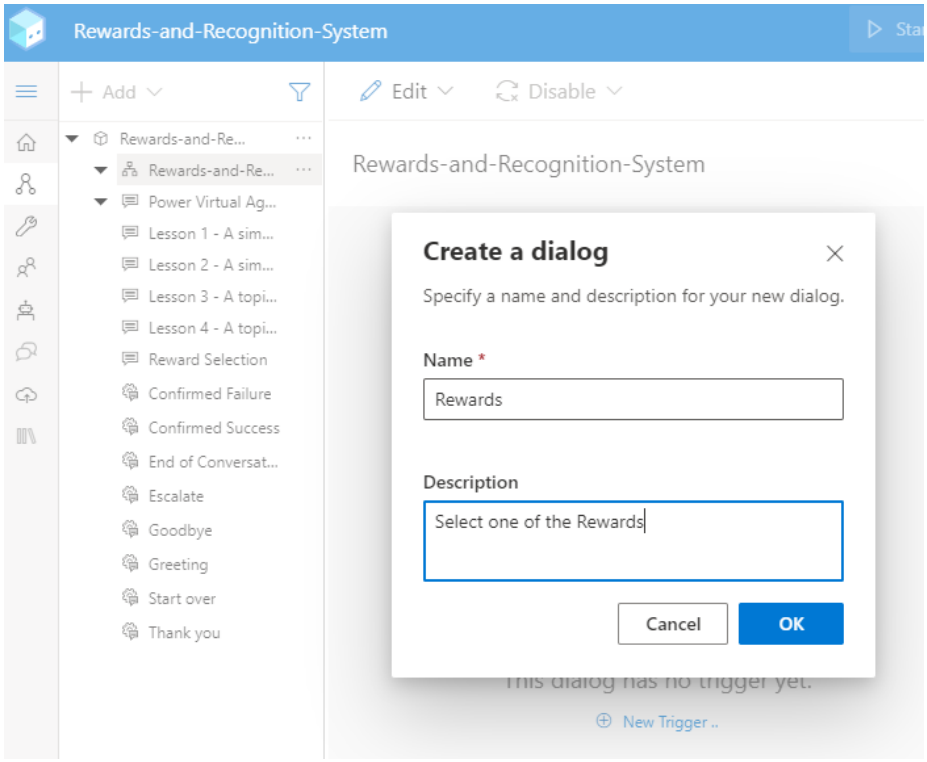


Create a Dialog

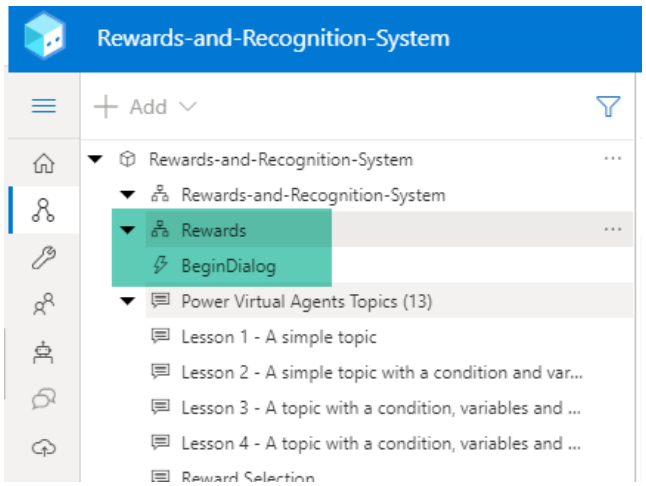
As the first step click on the 3 dots of the root project and click on add a dialog



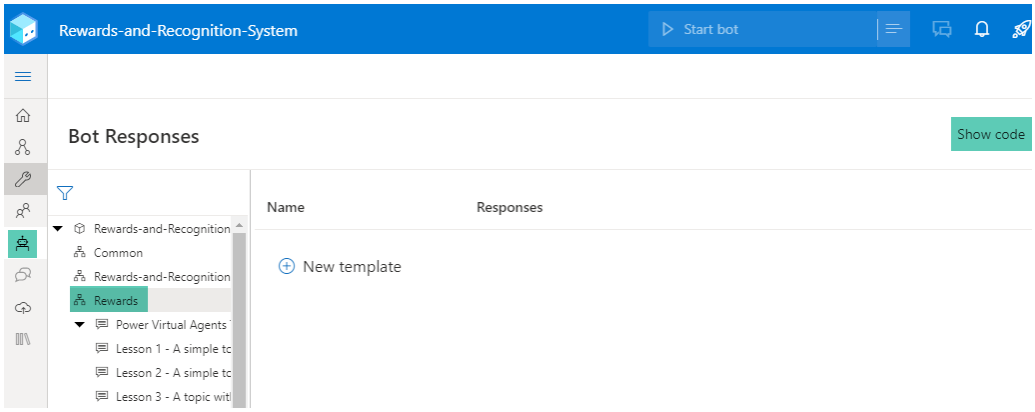
Specify the name and click on OK.



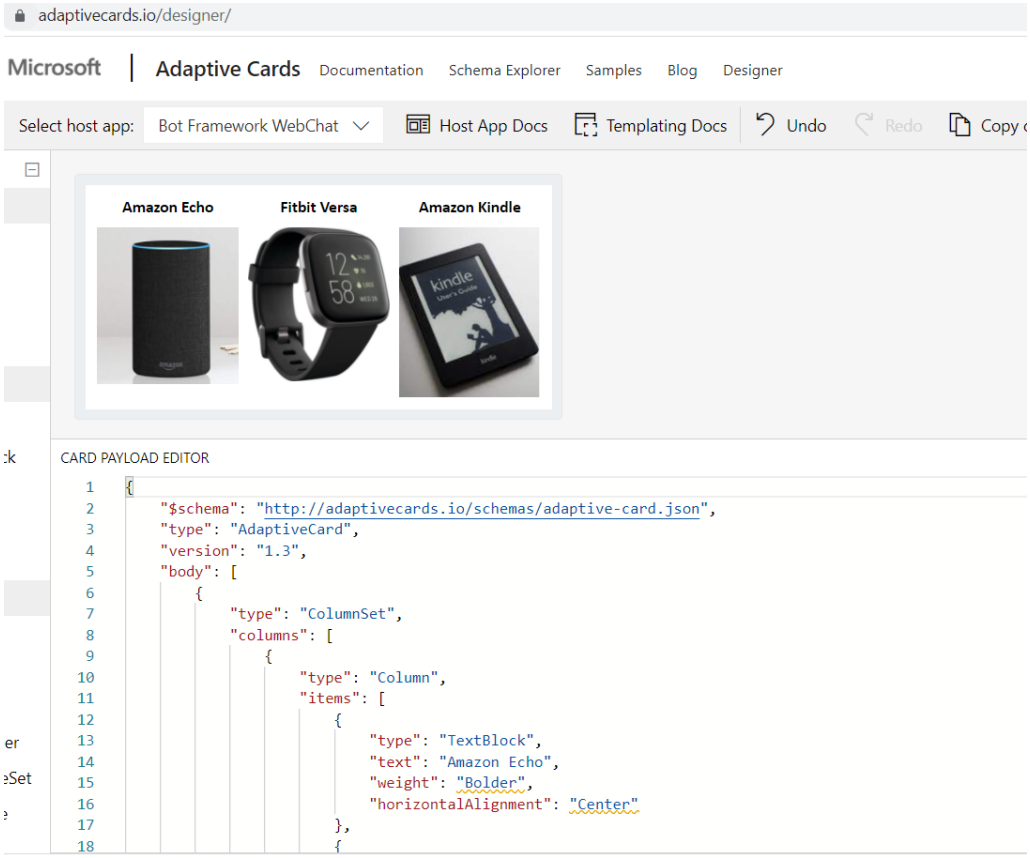
It will add a new Rewards Dialog to the project



Select Rewards and Click on the Bot icon in the Left pane and click on Show code to add Templates which will be used for displaying adaptive card schema using the [adaptive card designer](#) .



We have created the needed Adaptive Card in the adaptive card designer as below



Let's copy the schema to the composer as :

Here # <TemplateName()> is the syntax to define a template and a template can either have a single text value which is represented as

TemplateName ()

-Value

In case of multiline values, it is represented with the syntax

TemplateName ()

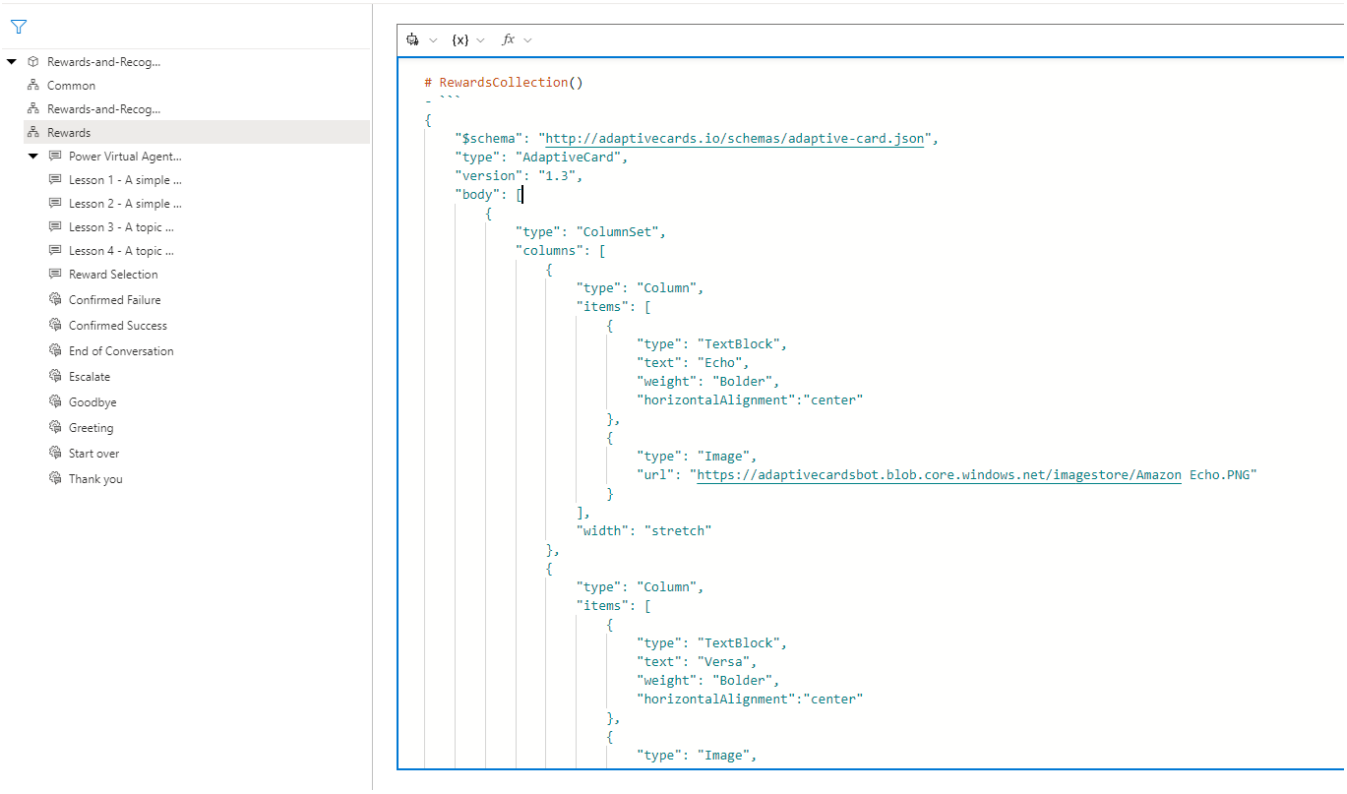
- ```

<Multi Line Text>

```

```
RewardsCollection()
- ```
{
    "$schema": "http://adaptivecards.io/schemas/adaptive-card.json",
    "type": "AdaptiveCard",
    "version": "1.3",
    "body": [
```
```

```
        {
            "type": "ColumnSet",
            "columns": [
                {
                    "type": "Column",
                    "items": [
                        {
                            "type": "TextBlock",
                            "text": "Echo",
                            "weight": "Bolder",
                            "horizontalAlignment":"center"
                        },
                        {
                            "type": "Image",
                            "url": "https://adaptivecardsbot.blob.core.windows
.net/imagestore/Amazon Echo.PNG"
                        }
                    ],
                    "width": "stretch"
                },
                {
                    "type": "Column",
                    "items": [
                        {
                            "type": "TextBlock",
                            "text": "Versa",
                            "weight": "Bolder",
                            "horizontalAlignment":"center"
                        },
                        {
                            "type": "Image",
                            "url": "https://adaptivecardsbot.blob.core.windows
.net/imagestore/Fitbit.PNG"
                        }
                    ],
                    "width": "stretch"
                },
                {
                    "type": "Column",
                    "items": [
                        {
                            "type": "TextBlock",
                            "text": "Kindle",
                            "weight": "Bolder",
                            "horizontalAlignment":"center"
                        },
                        {
                            "type": "Image",
```

```
                            "url": "https://adaptivecardsbot.blob.core.windows
.net/imagestore/Kindle.PNG"
                        }
                    ],
                    "width": "stretch"
                }
            ]
        }
    ]
}
```

Bot Responses



The adaptive card schema uses the columnset to show the Image which is stored in blob storage in 3 columns

RewardsCollection()

```
{
    "$schema": "http://adaptivecards.io/schemas/adaptive-card.json",
    "type": "AdaptiveCard",
    "version": "1.3",
    "body": [
        {
            "type": "ColumnSet",
            "columns": [
                {
                    "type": "Column",
                    "items": [
                        {
                            "type": "TextBlock",
                            "text": "Echo",
                            "weight": "Bolder",
                            "horizontalAlignment":"center"
                        },
                        {
                            "type": "Image",
                            "url": "https://adaptivecardsbot.blob.core.windows.net/imagestore/Amazon Echo.PNG"
                        }
                    ],
                    "width": "stretch"
                },
                {
                    "type": "Column",
                    "items": [
                        {
                            "type": "TextBlock",
                            "text": "Versa",
                            "weight": "Bolder",
                            "horizontalAlignment":"center"
                        },
                        {
                            "type": "Image",
                            "url": "https://adaptivecardsbot.blob.core.windows.net/imagestore/Fitbit.PNG"
                        }
                    ],
                    "width": "stretch"
                },
                {
                    "type": "Column",
                    "items": [
                        {
                            "type": "TextBlock",
                            "text": "Kindle",
                            "weight": "Bolder",
                            "horizontalAlignment":"center"
                        },
                        {
                            "type": "Image",
                            "url": "https://adaptivecardsbot.blob.core.windows.net/imagestore/Kindle.PNG"
                        }
                    ],
                    "width": "stretch"
                }
            ]
        }
    ]
}
```

ColumnSet to display as 3 columns

Echo Text and Image

FitBit Tex and Image

Kindle Text and Image

Following this, lets add the Activity that will display this Adaptive Card in the Bot Framework dialog to the same window in Composer

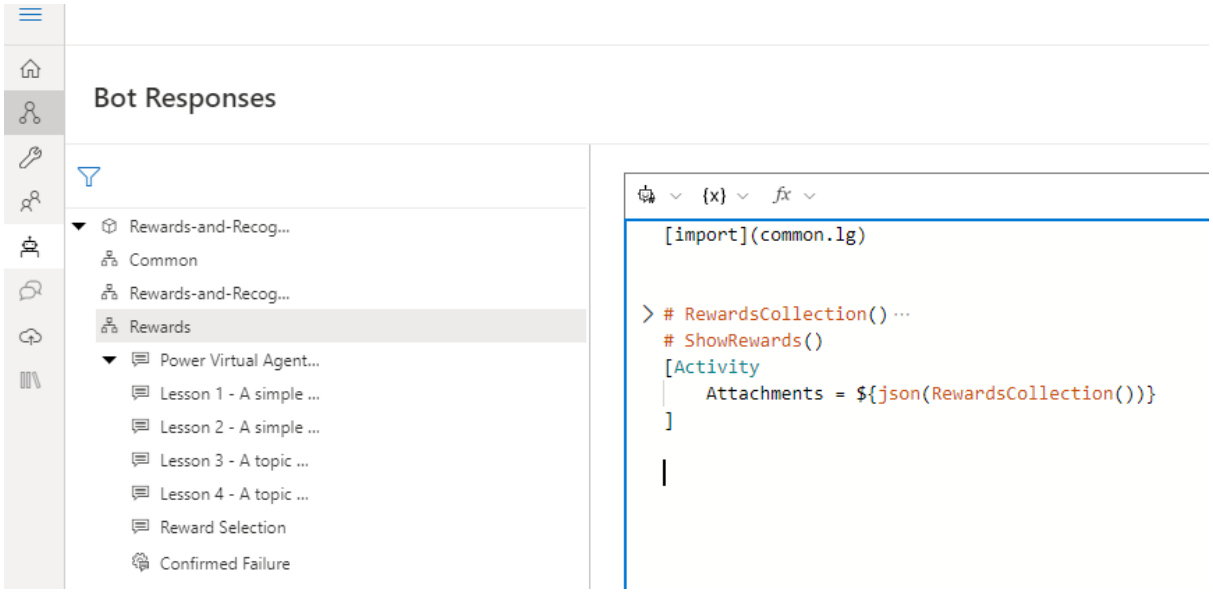
ShowRewards()

[Activity

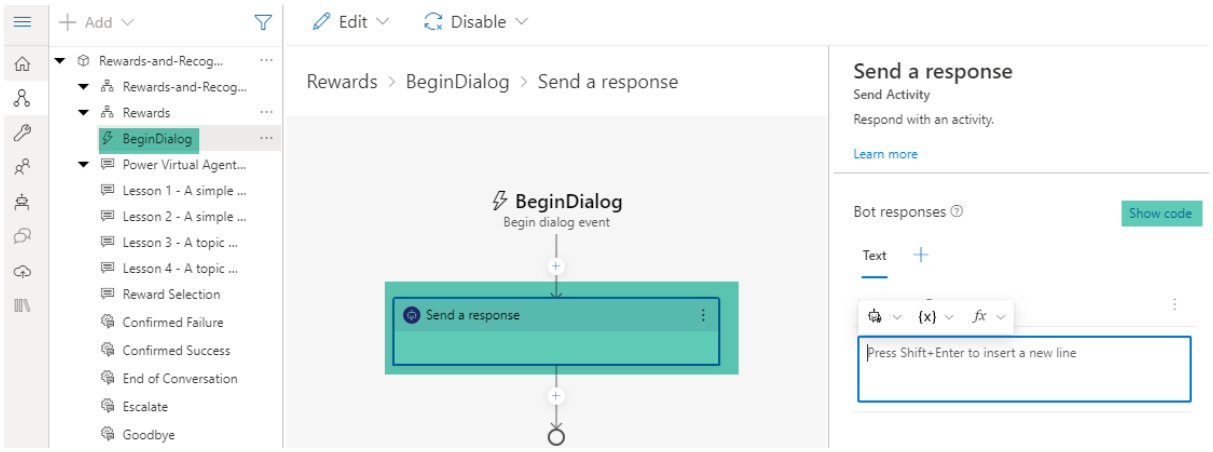
Attachments = ${json(RewardsCollection())}

]

Here we are calling the Previously defined Adaptive Card JSON and showing it in the Dialog

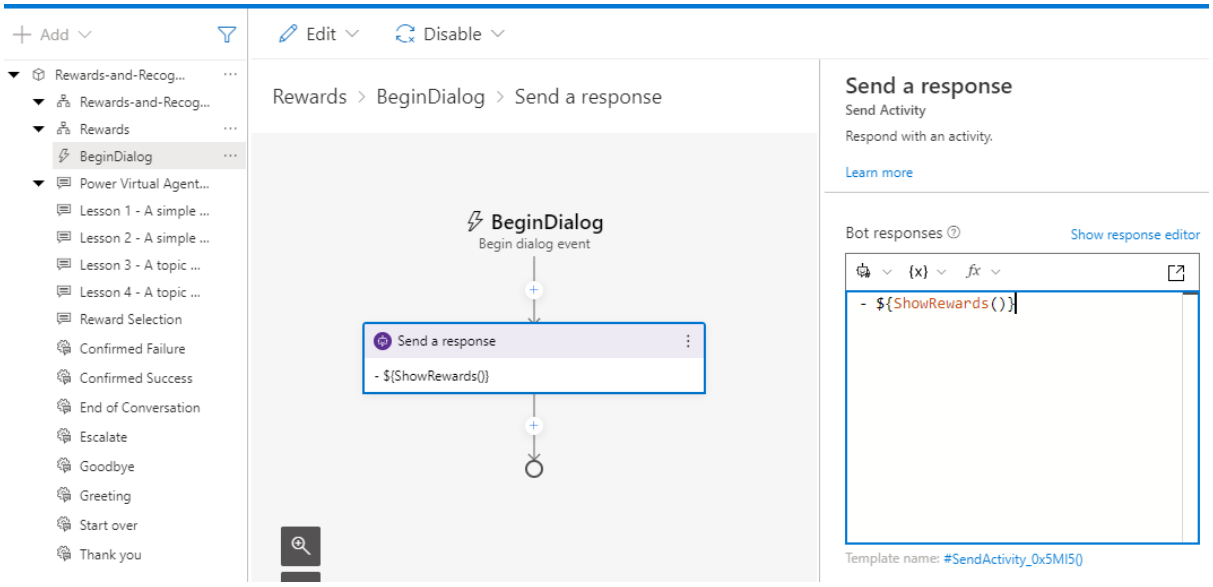


Now let's go back to the Create Tab and select the rewards dialog. Select the BeginDialog trigger and add the Send a response action to the dialog.



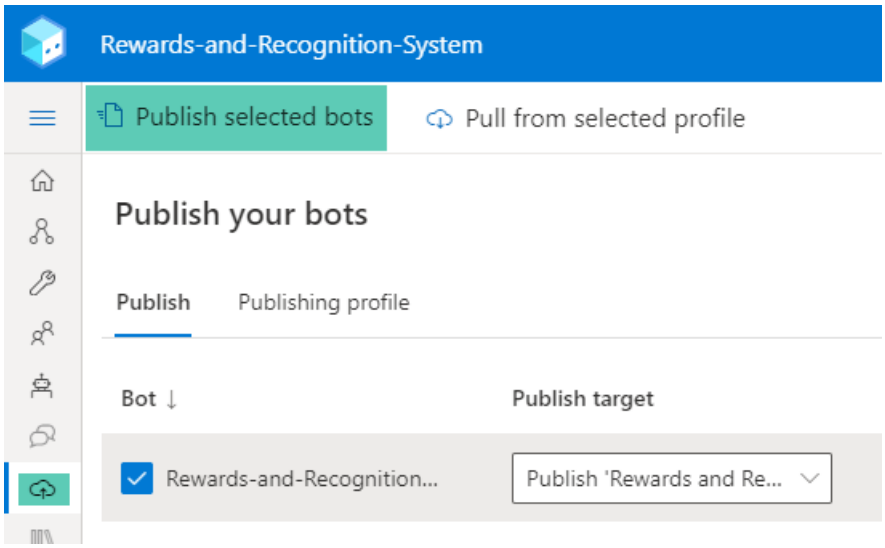
Selecting the send a response will open the Right Property Pane , Click on Show code and add the below expression to it so that it will call the Adaptive card and show it.

- ${ShowRewards()}

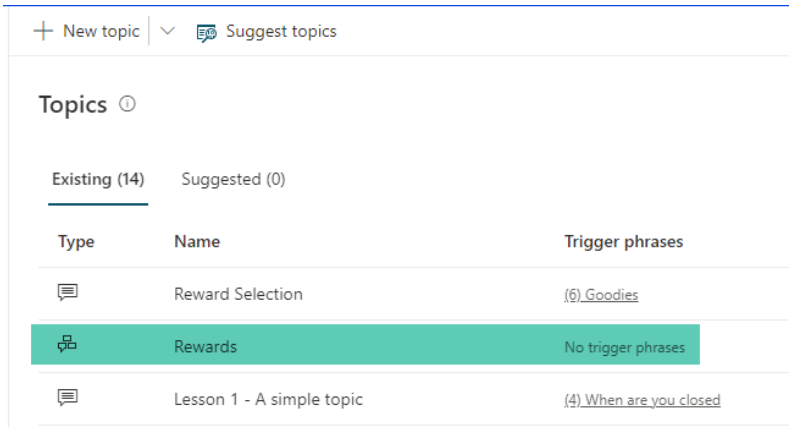


Publish the Bot

Now let's go ahead and publish the bot so that the Dialog that we have created becomes available as a topic in Power Virtual Agents.

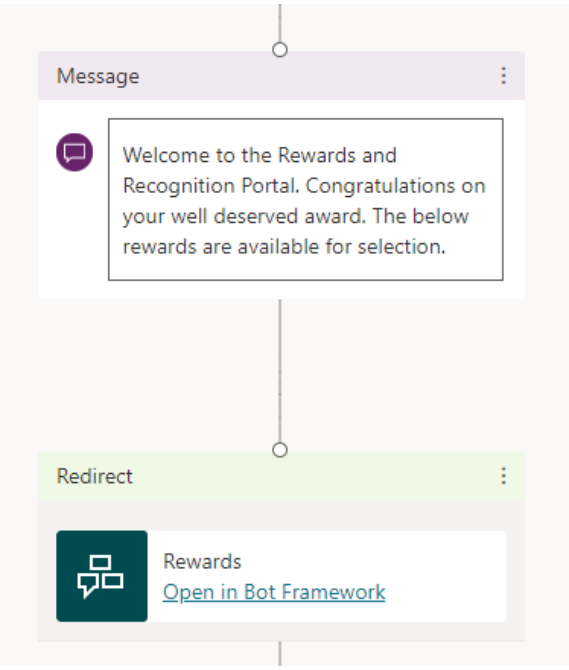


Heading back to Power Virtual Agents, we can see the recently created dialog in the list of topics.

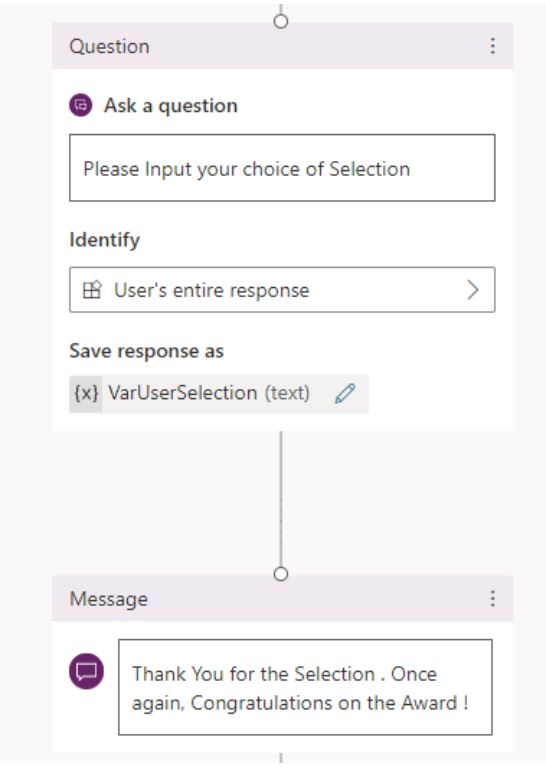


Redirect to Adaptive Card Display Topic

Let's add a redirection from the Topic that triggers the conversation to the newly created topic so that we display the images as adaptive cards.

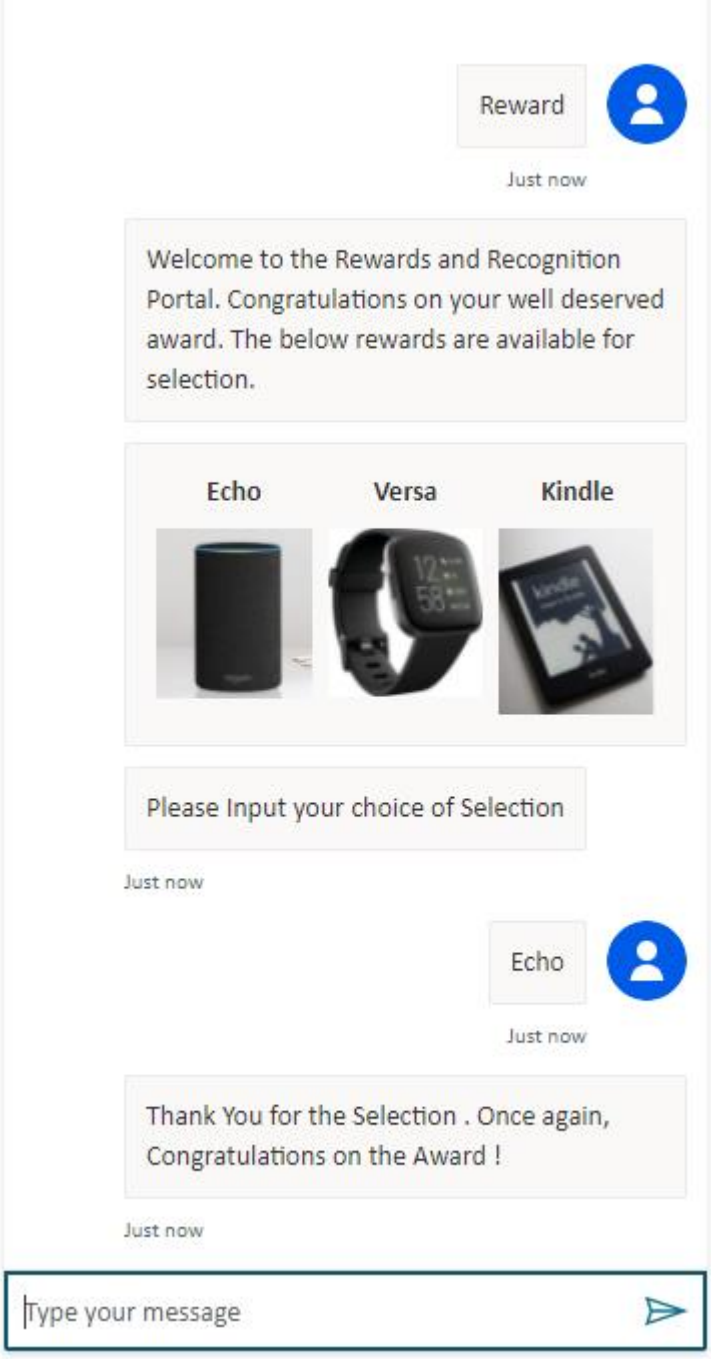


Once the Adaptive cards are displayed using the new topic, the control will return to the initial topic. To complete the flow, lets ask the user for his selection of gift and close the conversation with a Message.



Test the Bot

Now let's test the bot end to end by triggering the conversation. We can see that from the initial topic, it redirects to the Topic created using Composer and it displays the Images using an adaptive card and return backs to the initial topic to further collect the details.



Summary

Thus, we saw how to plug in an adaptive card to show images to the end users which is created and published using Bot Framework Composer.

Implement Power Virtual Agent Hand Over to Human Agent using Omnichannel

Introduction

We have seen how to automate the interaction with Power Virtual Agents using Topics and how to drive the conversation based on the user inputs. However, there can be a situation where the Topics that we have defined is not covering the problem statement that the user is facing or he may have a query that is not covered within the existing Topics. In this scenario, we a human intervention is required.

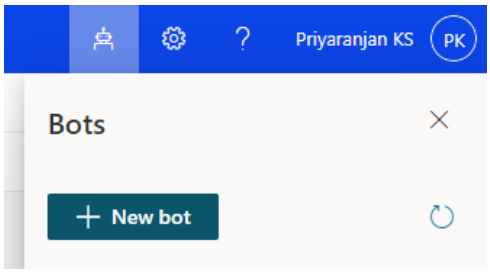
In this article, we will see how to transfer the conversation from a bot to a Live Agent who is a Human user to take up further communication with the End user.

Scenario

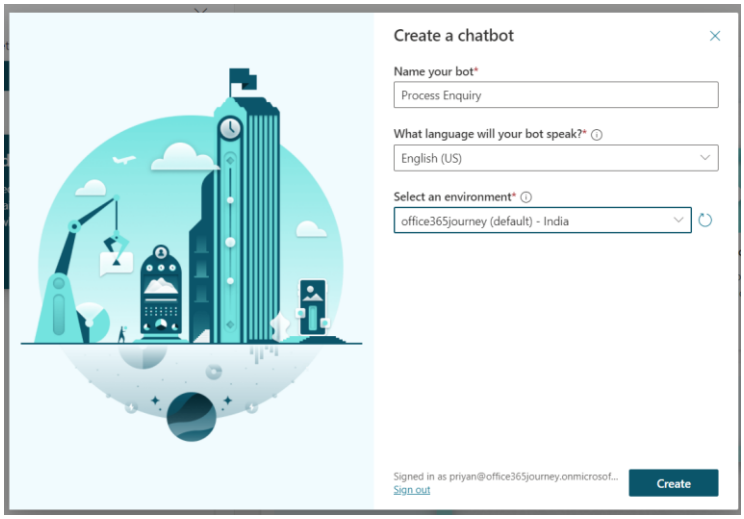
We will consider a scenario where we will build a simple bot that provides the Organization links to internal tools, process documents and Project Repositories. However, there can be a situation where the requested information is not present in the Power Virtual Agent topic to provide a response. In such a case, we will try to implement handing over the conversation to a human agent who can assist in sharing the requested information.

Implementation

Let's head over to https://web.powerva.microsoft.com/ and click on the Bot symbol on the top right corner which lets you create a new bot



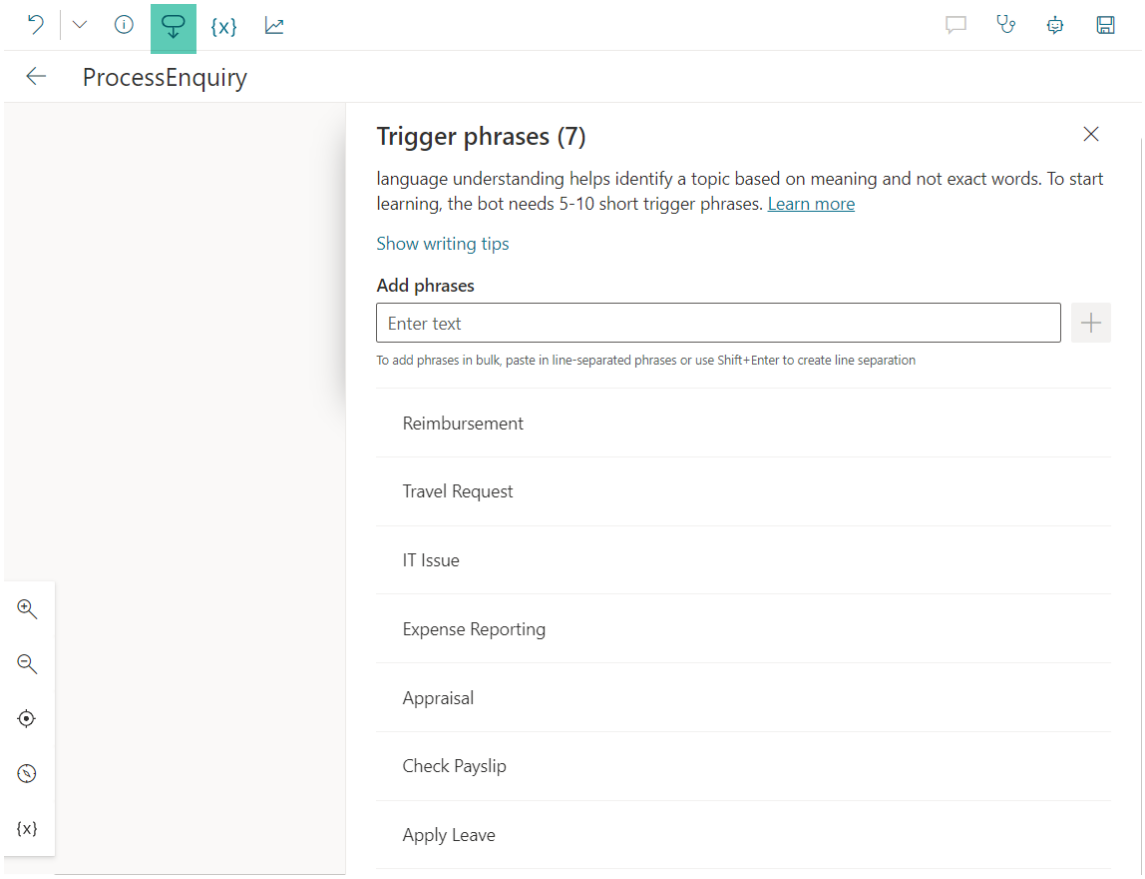
Specify the name and language for use in the bot as well as the environment where it should be provisioned.



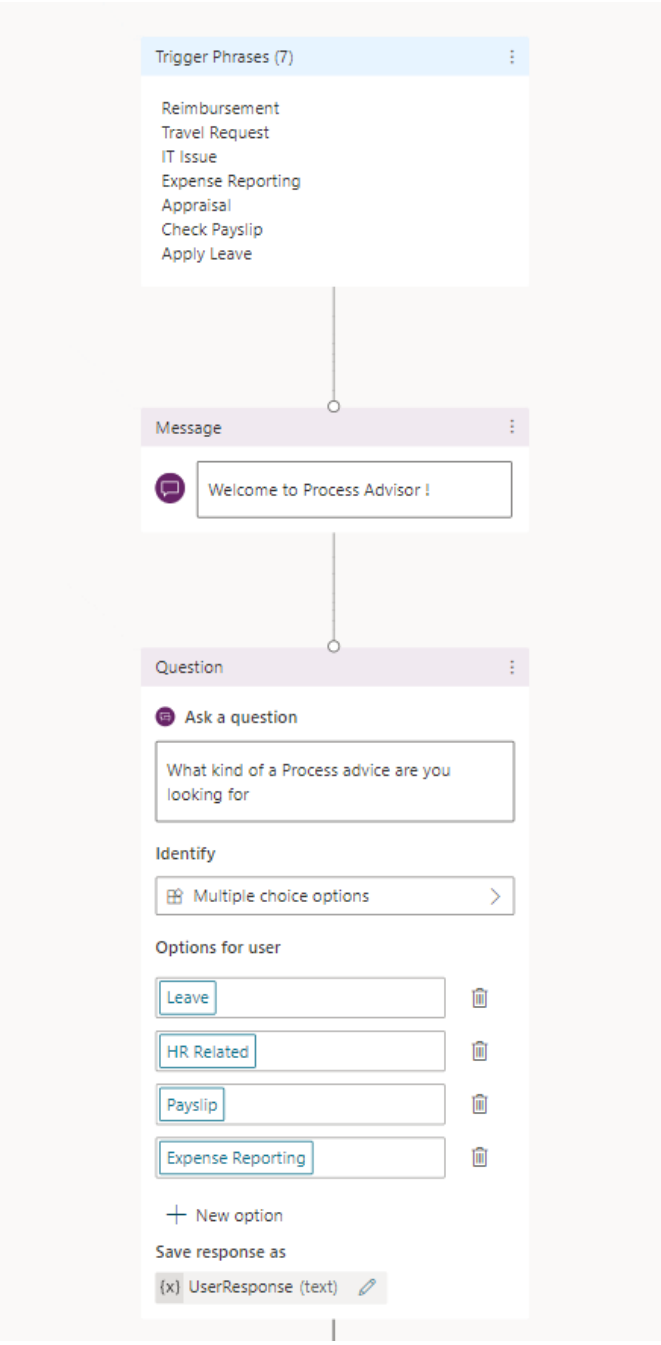
It will create a basic bot where we can add the customizations needed for our requirement. We can see the section Topics which by default lists multiple conversation topic listed out. A topic defines how a bot conversation will be initiated and how the bot would respond to user interactions.

Click on New Topic and Select From blank to create a Topic and we will name it ProcessEnquiry.

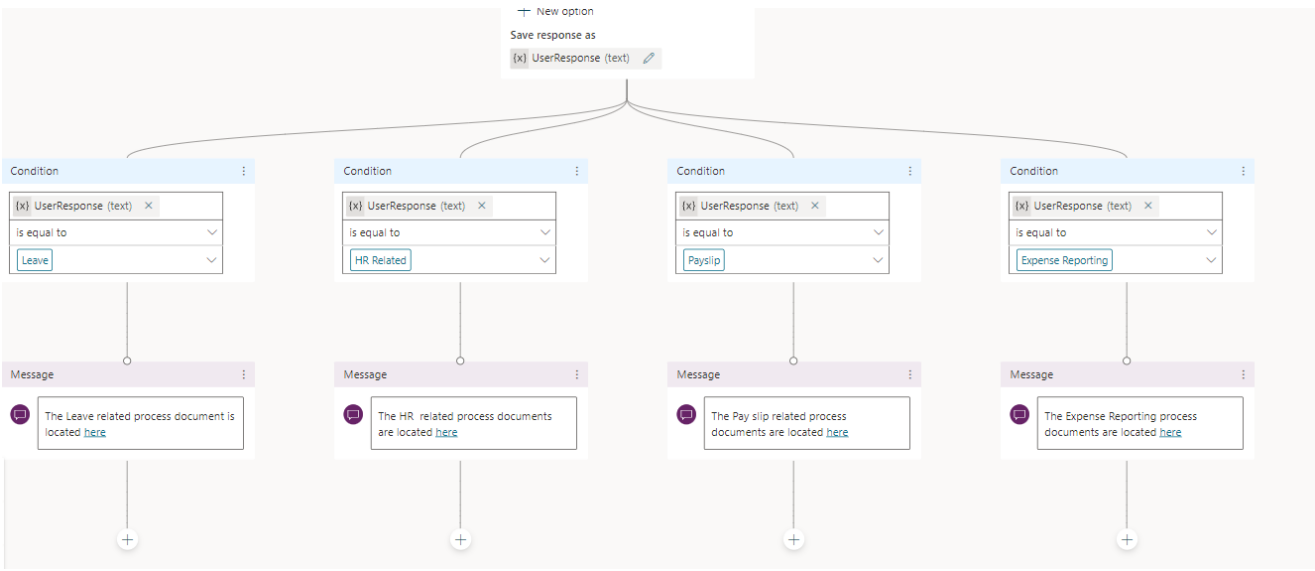
To trigger the topic, so that the control flows to that specific topic, we will define few words called Trigger Phrases that will transfer the control to these topics. The trigger phrases can be single keywords or a group of words and to cover a broad spectrum of possible trigger conditions, It is good to mention 5-10 different and still related phrases. Click on Trigger phrases that will open a right pane where we can add the trigger words.



We will then add a welcome message followed by the question to the user to understand what kind of information he/she is looking for

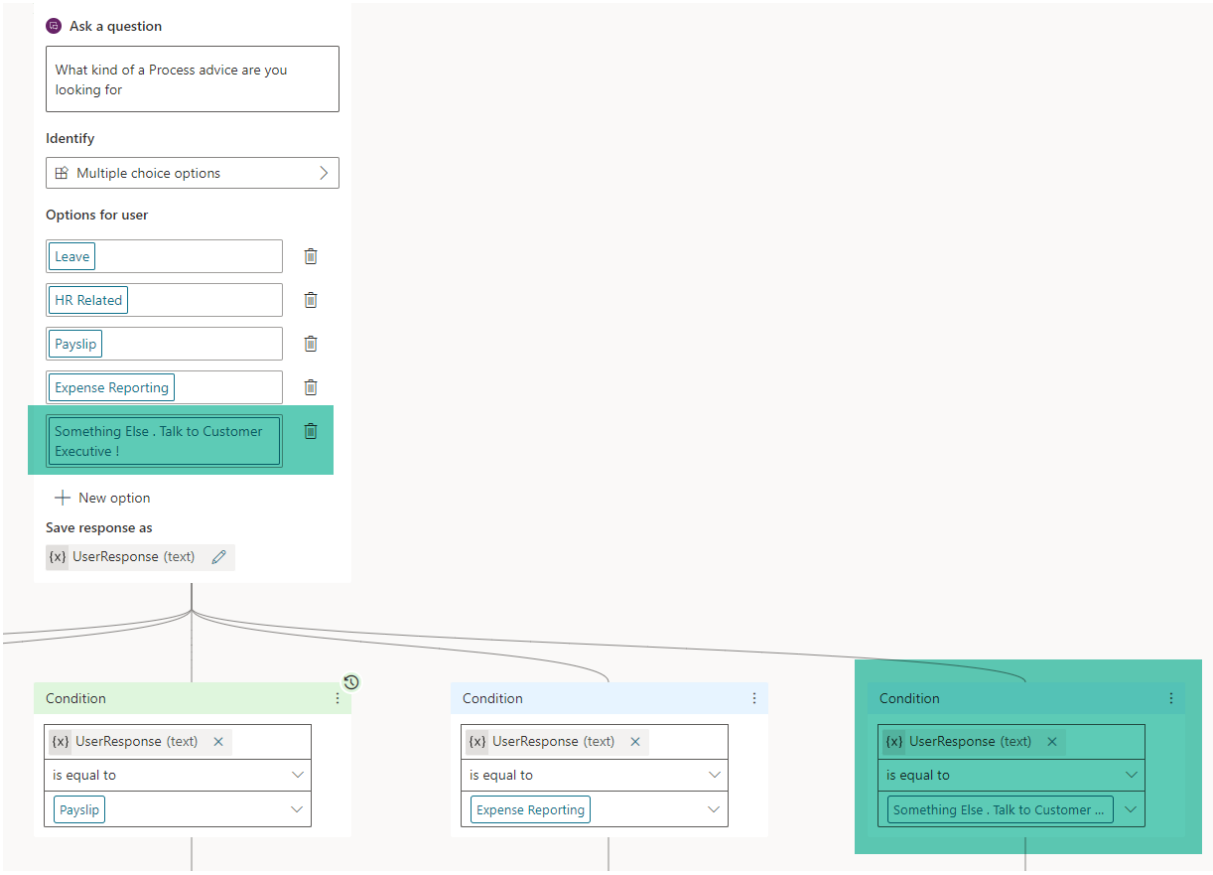


Based on the user input, we will branch the conversation to 4 branches which will show the respective process document links so that the user can navigate to it to get the needed information.

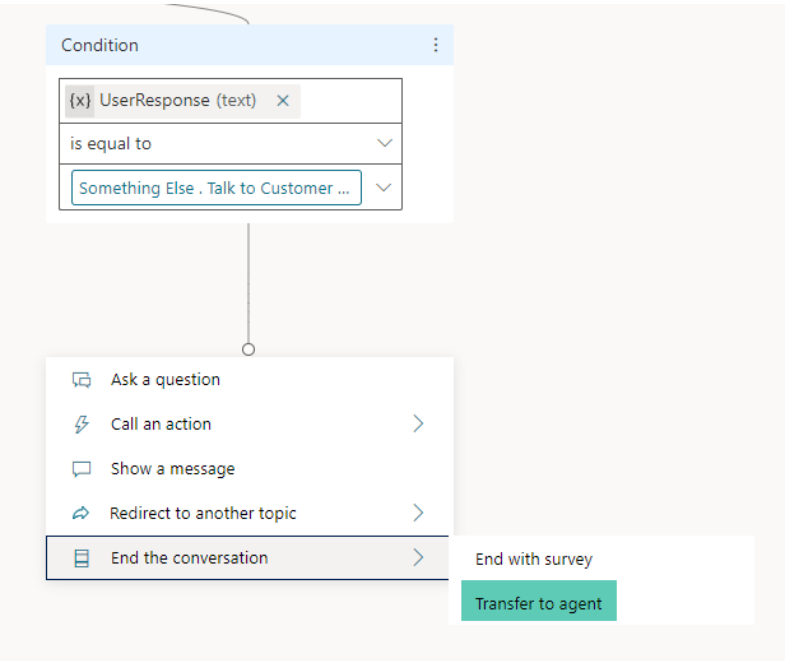


However, if the user has a query not related to the provided options above, he should still have an option to get the needed information for which we will include a new functionality to transfer the conversation to a live agent.

Let's add a new Option to the initial question we were asking the user that gives the flexibility for the user to select it which will take the control to the branch that will hand off the conversation from the bot to the live agent.

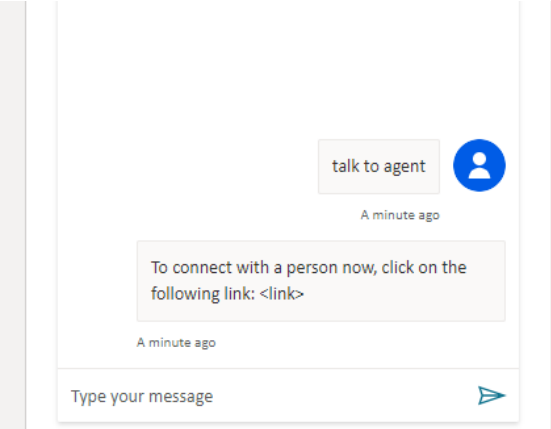


In the Hand off to Live Agent branch, we will add the node to transfer conversation to the agent.

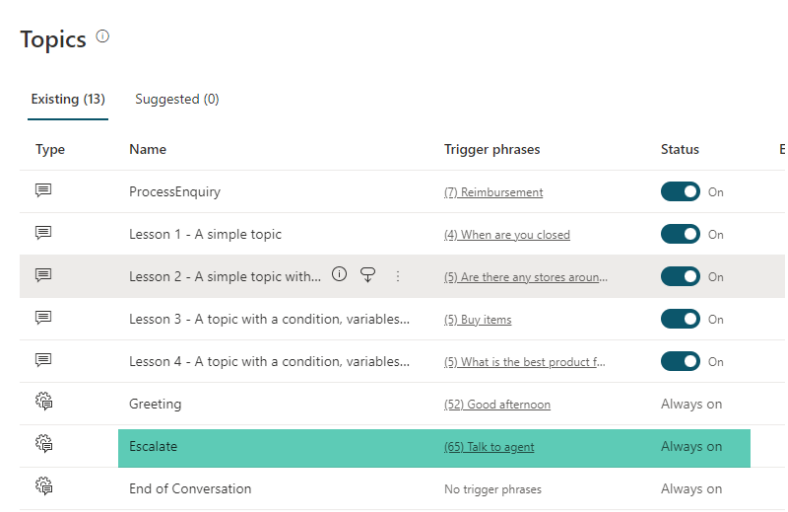


Modifying Default Transfer to Agent Trigger

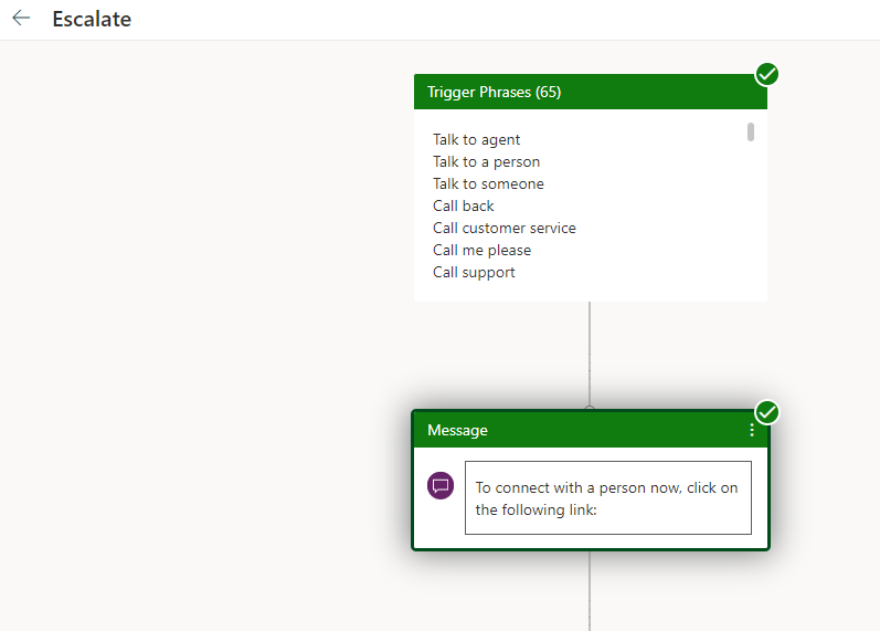
Power Virtual Agent by default has a topic that responds to the trigger words like “Talk to agent” that gives the below standard response.



We can view this Topic from the collection which is named as Escalate.

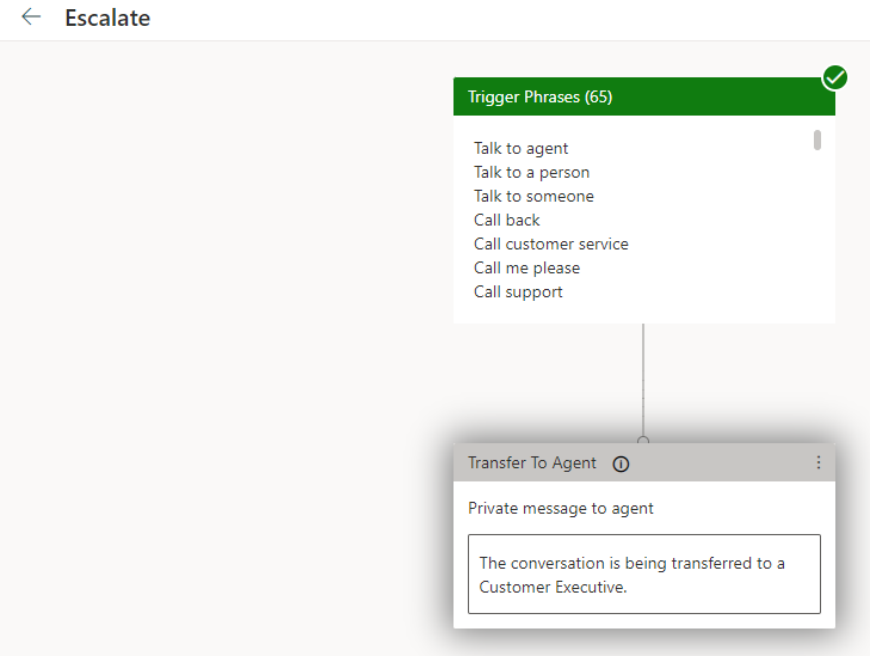


As we can see, this topic does not in reality transfer to an agent. It just outputs the message without providing the actual hand off link



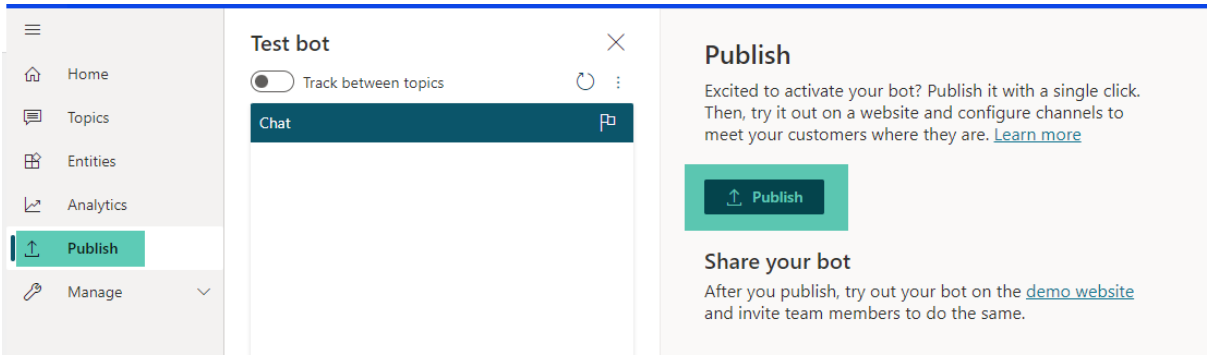
We handled the Transfer to agent scenario that occurs when a user clicks on the Transfer to agent option asked by the Bot. However, if the user types in a trigger word to transfer the conversation to a live agent, he will be seeing this meaningless message without any link. To avoid confusion for the end user, who may reach this topic by typing in the trigger word, lets delete this message and add a Transfer to Agent Node.

This way for any human interactions, we have a consistent flow that takes it to the Omnichannel that we are going to configure.

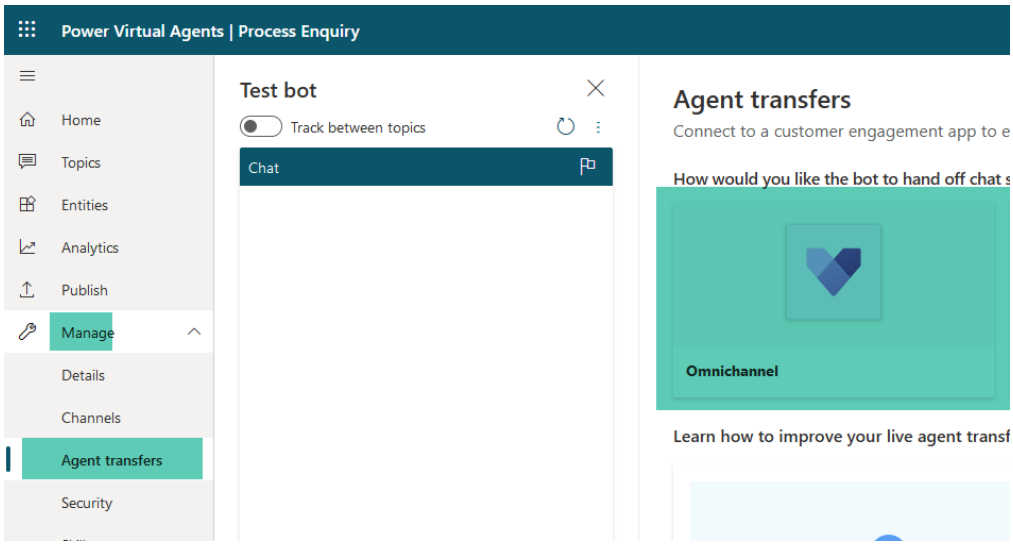


Configure Transfer to Agent Node

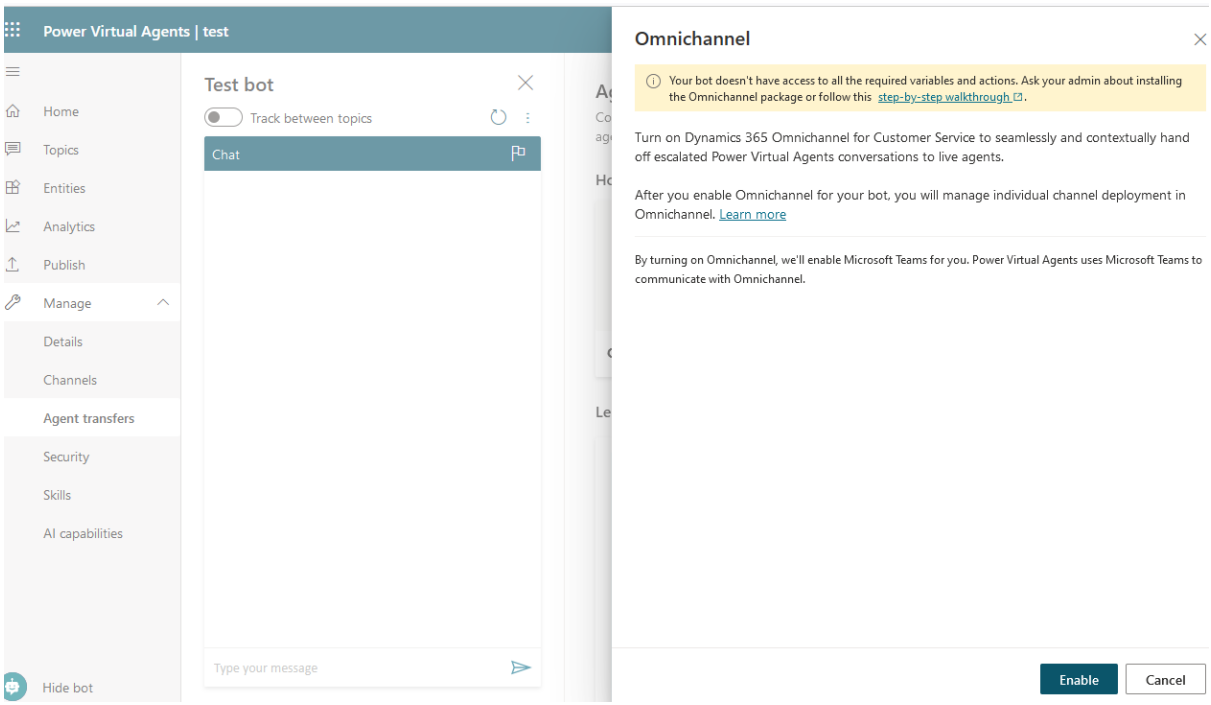
Before making further changes, lets save the Topic and Publish the bot once.



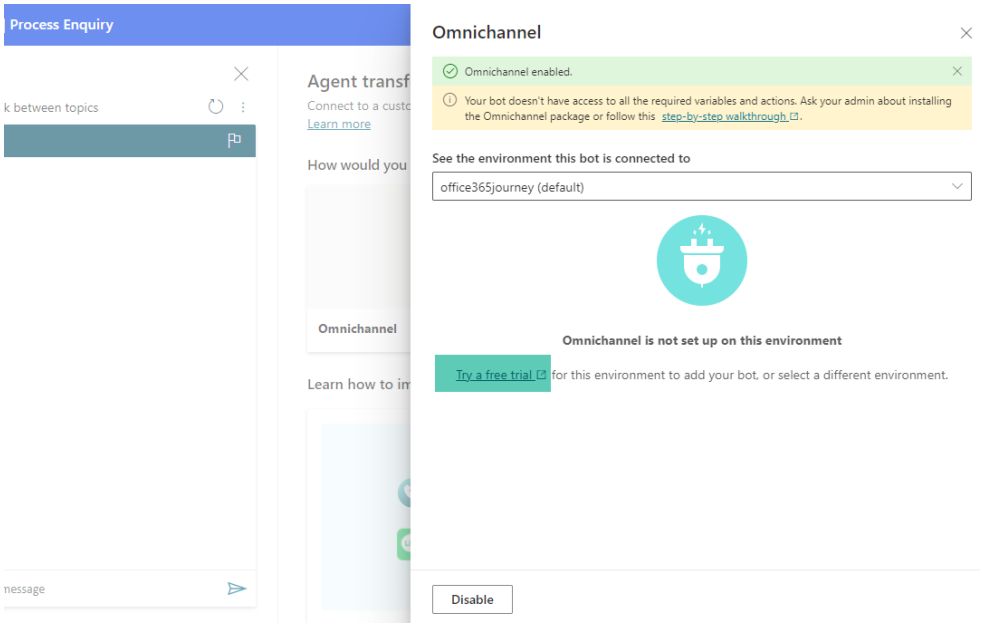
Now lets configure the Agent transfer by selecting Agent Transfers -> Omnichannel from the left pane.



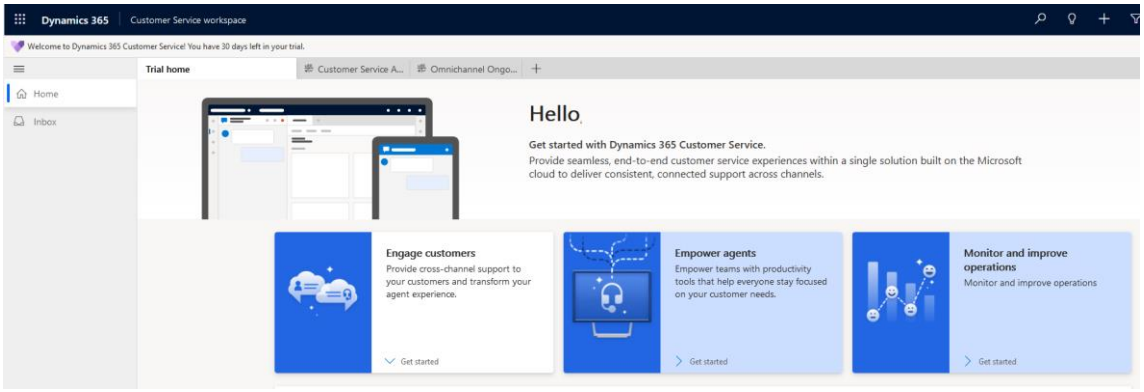
Select Enable to turn on the Omnichannel experience.



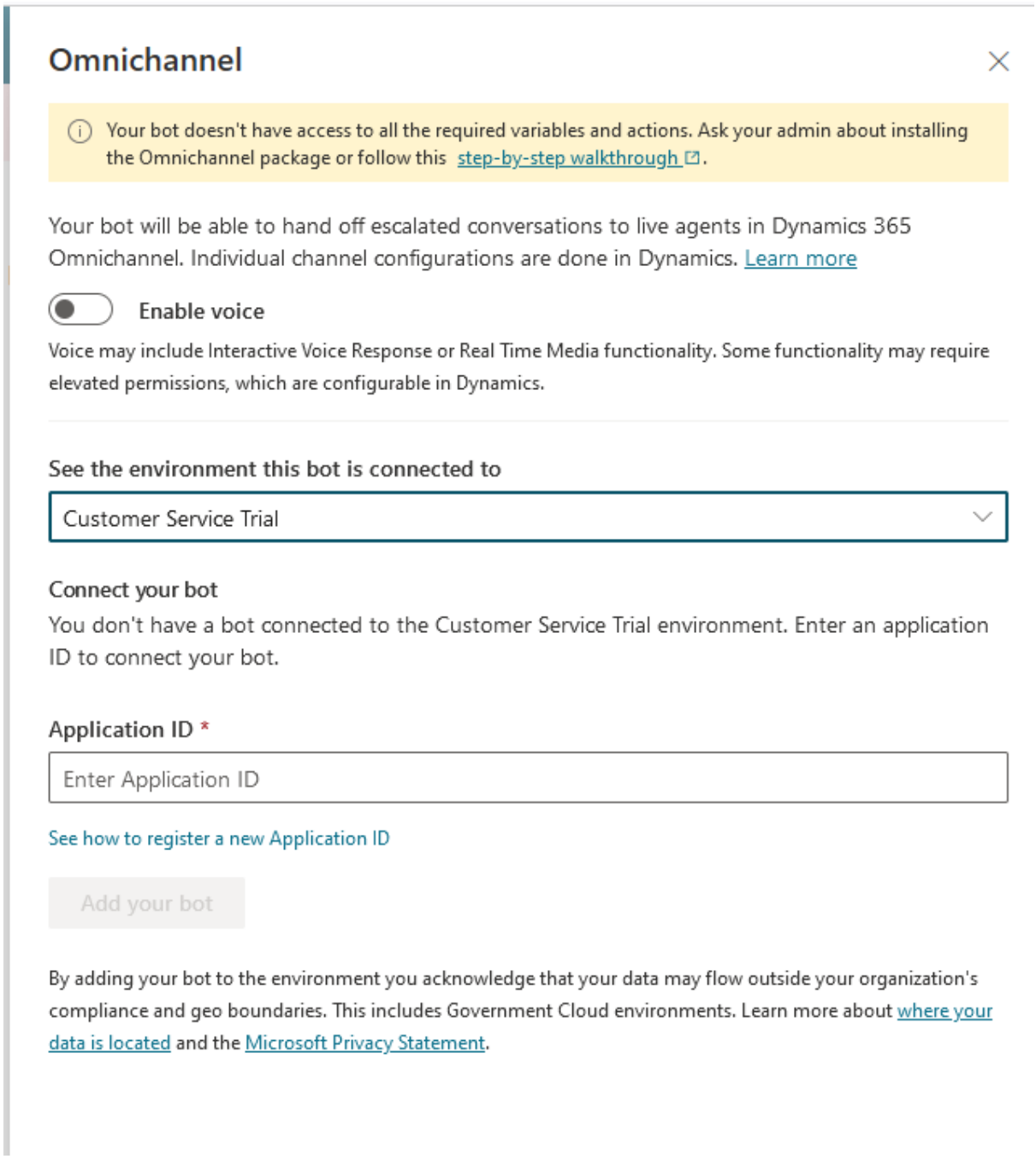
You would need a Dyanmics 365 Customer Service environment for the Omnichannel to work. If available, it will be listed in the environment drop down, else create a free trial by clicking on the link.



Enter the work email id and subscribe for a 30 day trial if you are activating Dyanmics 365 Customer Service for the first time. On Successful subscription and provisioning, we can see Dynamics 365 Customer Service in action



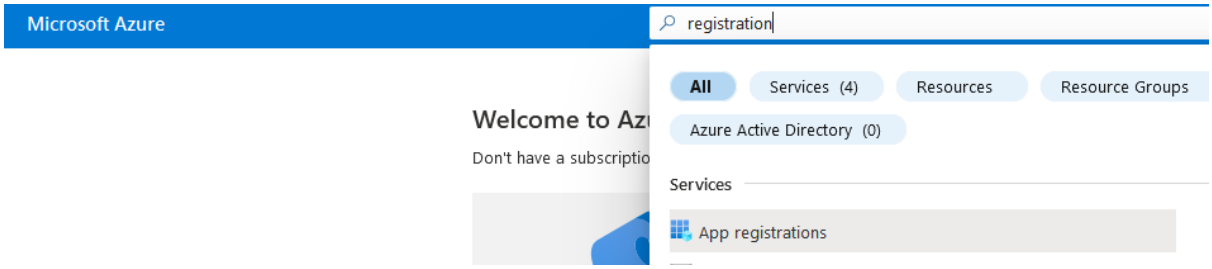
Now we can head back to the omnichannel pane where we can select the recently provisioned environment.



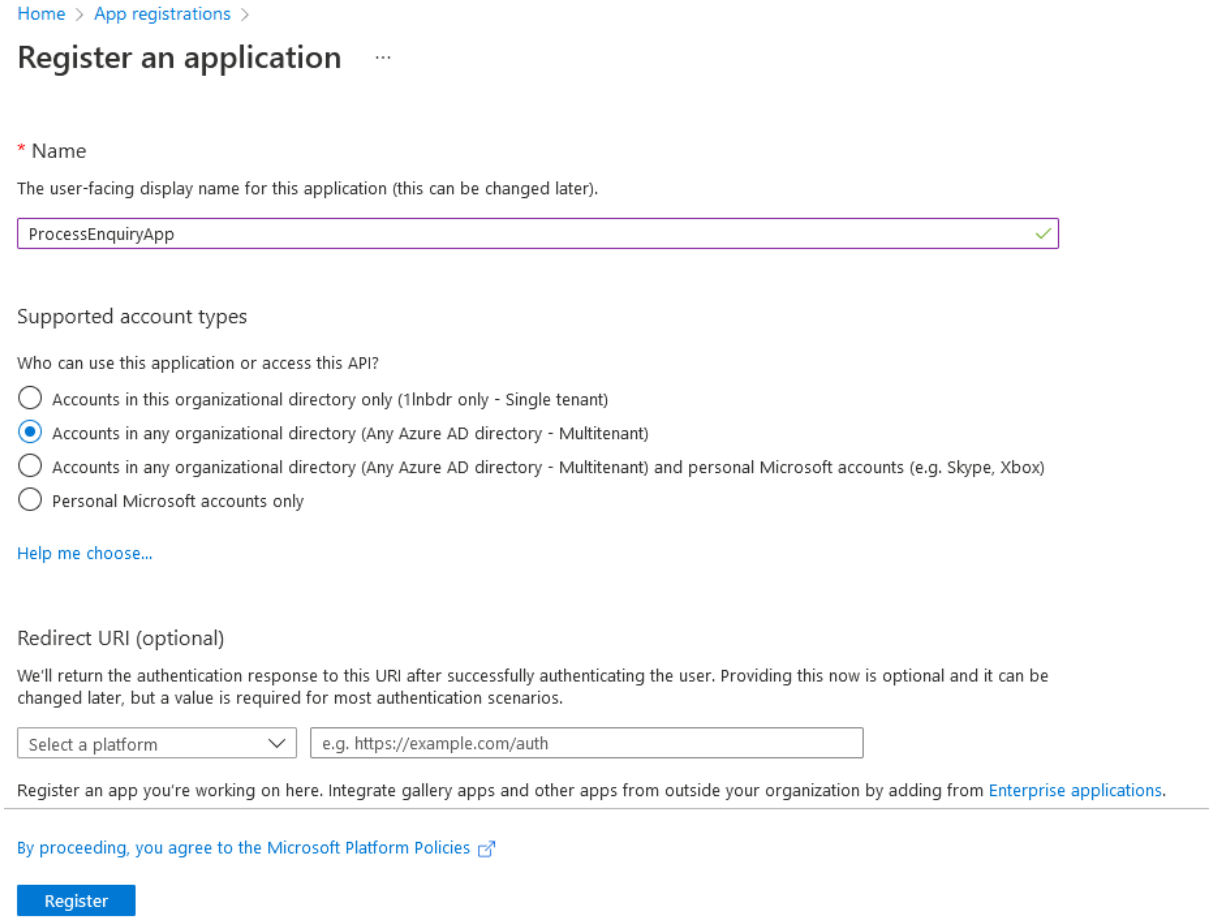
We can also connect the bot to the customer service environment by entering the Application ID which we will generate by registering in Azure.

Registering Application in Azure AD

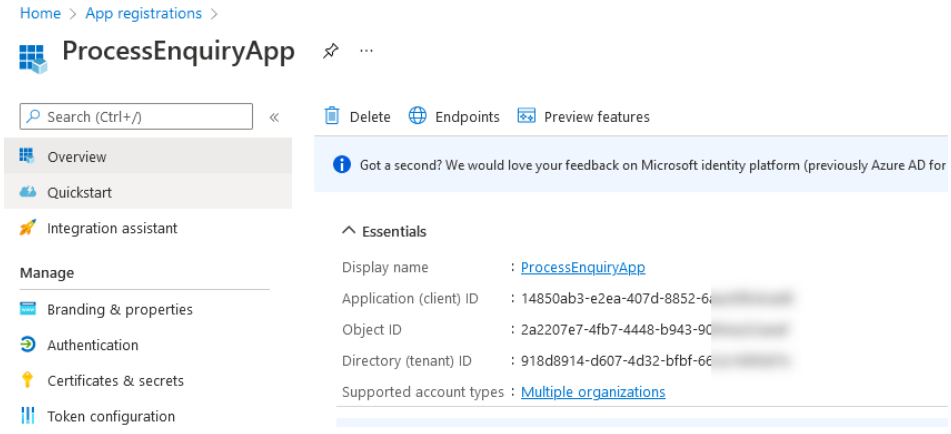
To generate the Application ID and to connect the bot with Customer Services lets register an application by heading over to portal.azure.com and search for App Registrations.



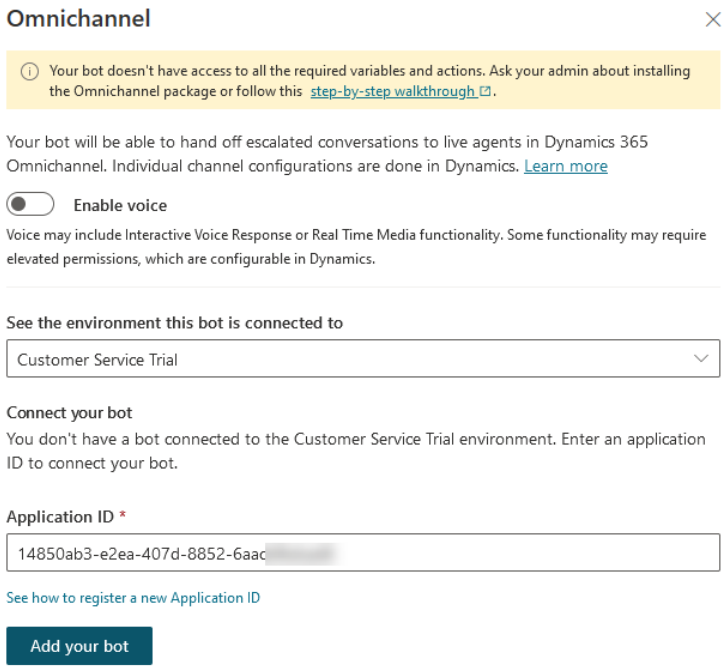
Click on New Registration and specify a name for the application and click on Register



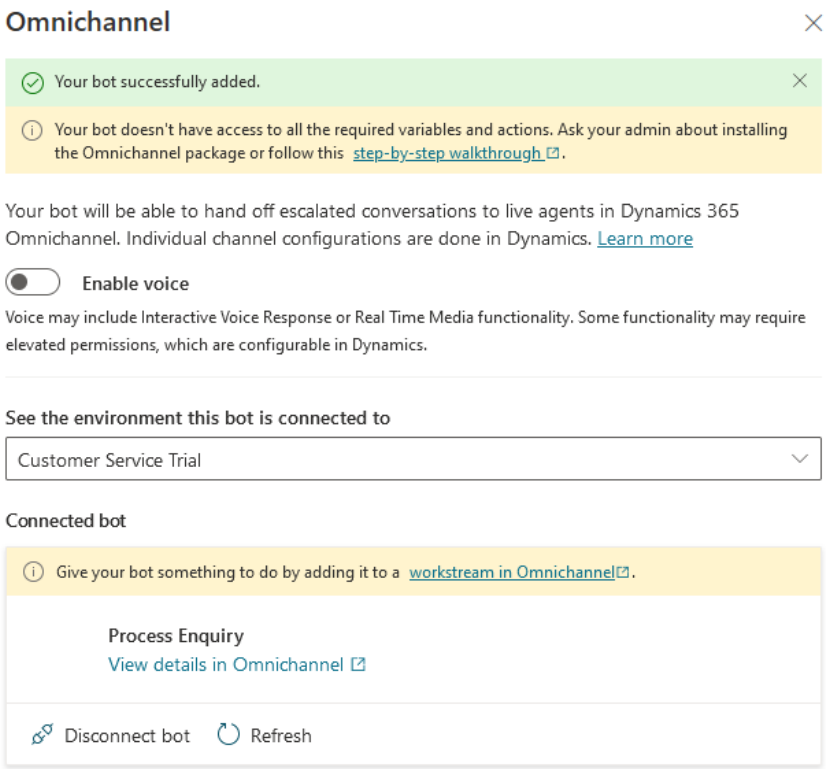
Once the Application is registered, select the Application ID from the Overview tab.



We will now paste this in the omnichannel pane. This will enable the Add your bot button. Click on it.



The bot has been successfully added



Setting up Omnichannel for Customer Service

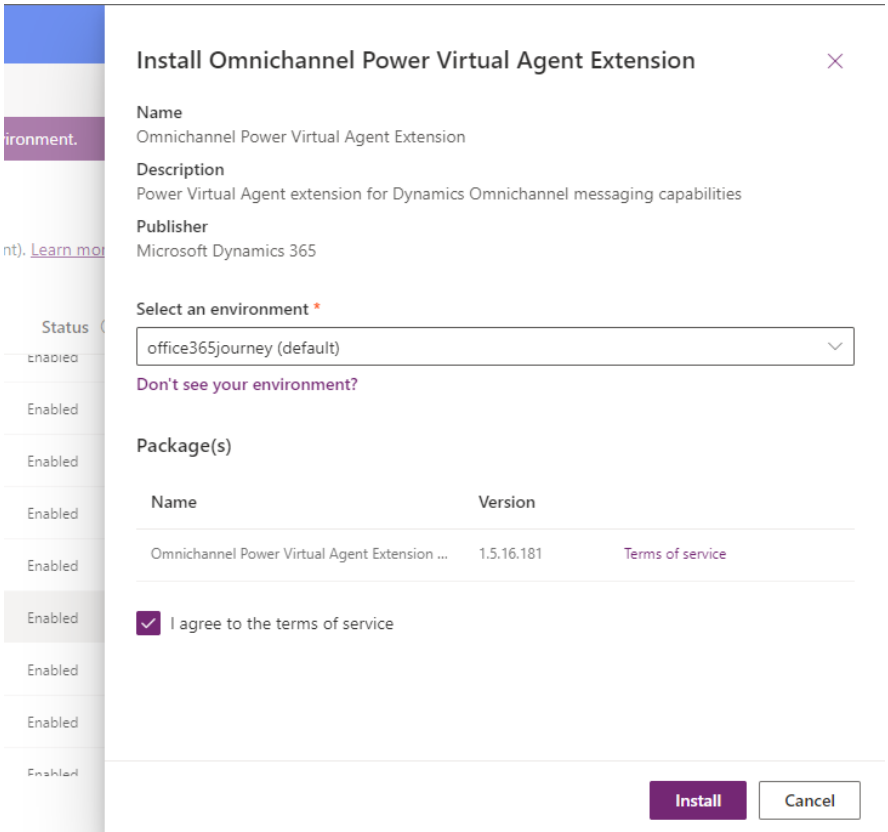
The yellow banner on top indicates missing features. For our bot to hand off a conversation to your omnichannel interface, we have to install the appropriate extension solutions for Dynamics 365 Customer Service omnichannel integration and Power Virtual Agents.

- If we only want text-based (messaging) hand off capabilities, we must install the following extension:
    - Omnichannel Power Virtual Agents Extension
- If we want both text-based (messaging) and voice-based hand off capabilities, we must install each of the following extensions in the following order:
    - Power Virtual Agents telephony extension
    - Omnichannel Power Virtual Agents extension
    - Omnichannel voice Power Virtual Agents extension

As we are going to hand over only chat conversations, we will just install the Omnichannel Power Virtual Agents Extension. Head over to Power Platform Admin centre and select the environment where omnichannel has to be configured. From Resources-> Dynamics 365 Apps , select Omnichannel Power Virtual Agent Extension.

Click on Install which will open a right pane where you would be asked to agree to the terms of service.

Specify the environment in which you have the bot created and click on Install.



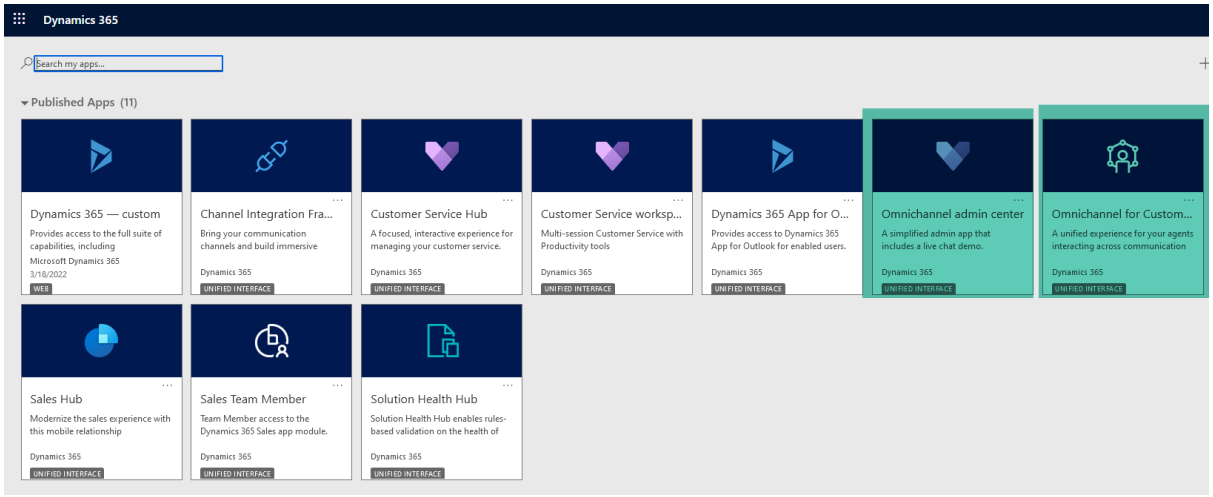
This will complete installation of the extension package

Now that we have completed the connection of the bot with the omnichannel, lets head over to Dynamics 365 to configure the Workstream and Queue which will handle the bot conversation handover.

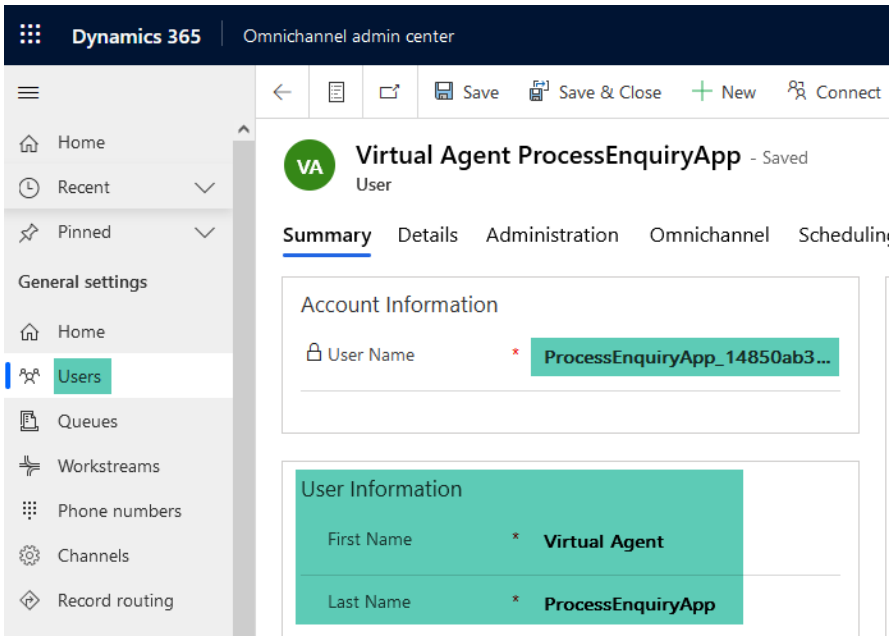
Configure the Omnichannel Hand Over

https://<tenant>.crm.dynamics.com/apps will take you to the Dynamics 365 Unified Interface page from which we can navigate to the Omnichannel Admin centre and Omnichannel for customer serivce with which we will work in the next steps.

First, let's head over to the Omnichannel admin centre.



In the users section we can see the Bot added as a User.



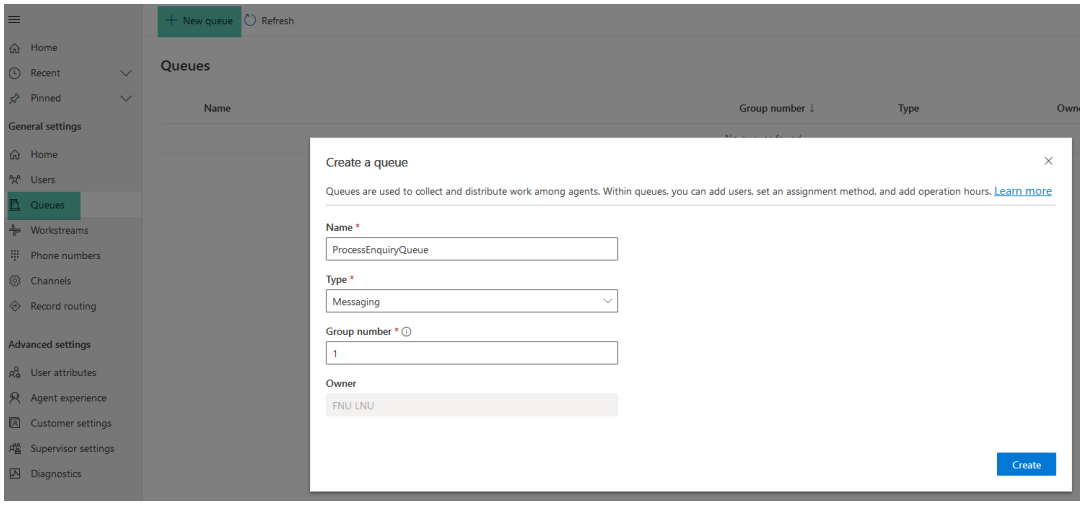
So as to get started with the configurations of the Omnichannel we will do 3 actions

- Create a Queue
- Create a Workstream
- Create a RuleSet

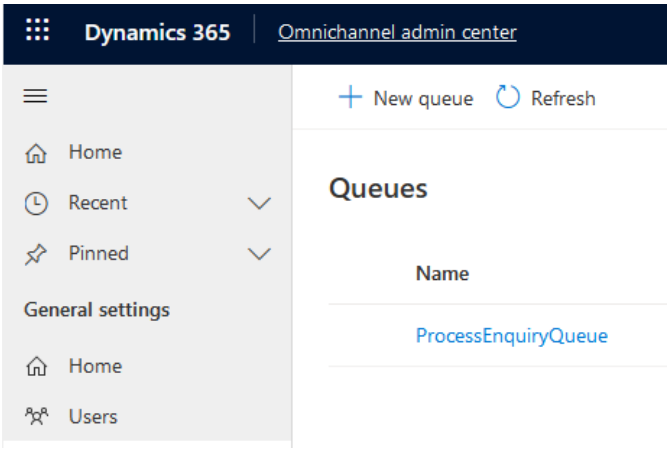
Create the Queue

To hold the incoming conversations that are handed over, we will create a Queue and the human agent as well as the bot that we have created to the queue.

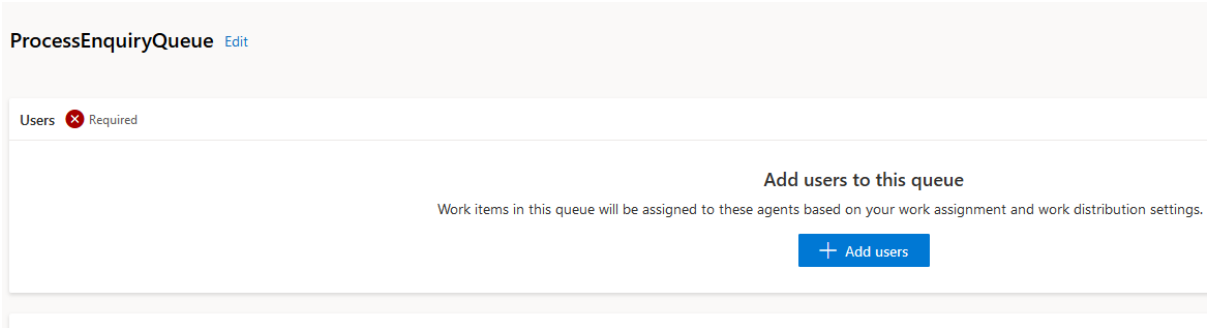
From the Queue tab in the left pane, click on Queue and specify the name and the type of the queue to be created. We can create a Messaging/Record/Voice queues. For this demo, we will create a messaging handover queue.



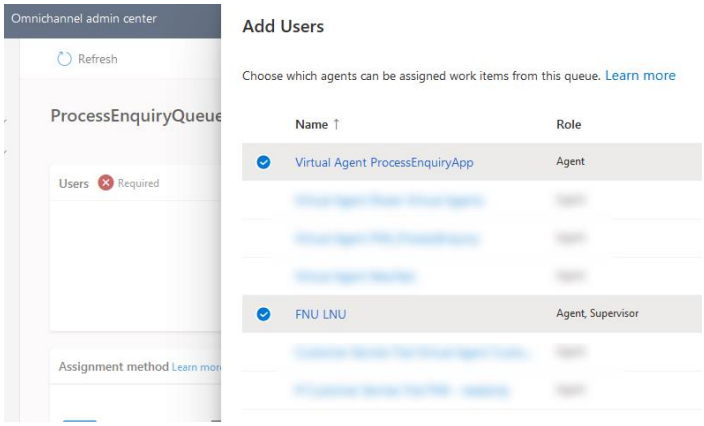
Clicking on Create will provision the Messaging Queue



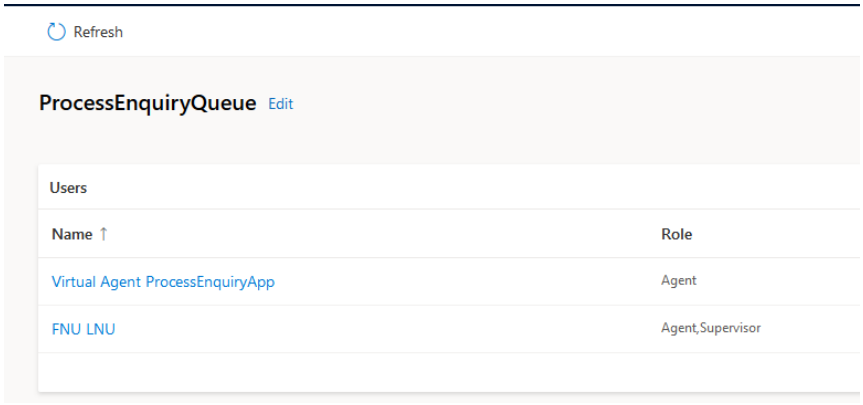
Thus the queue is created and now we need to add the Human Agent as well as the Power Virtual Agent to the Queue



Click on Add users to add the agents to the queue.

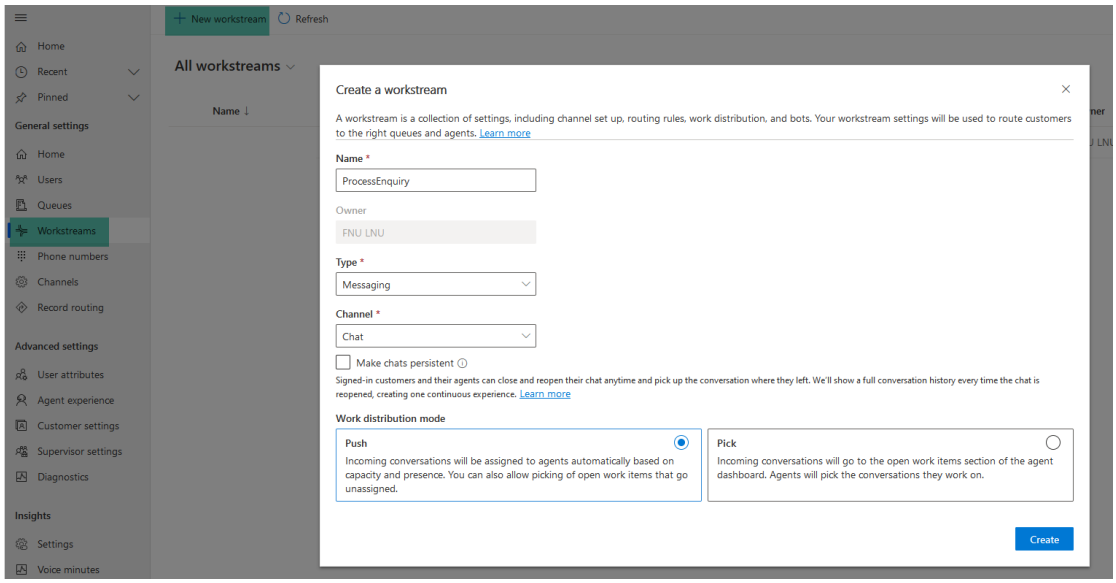


Thus we have completed the setting up of the queue.

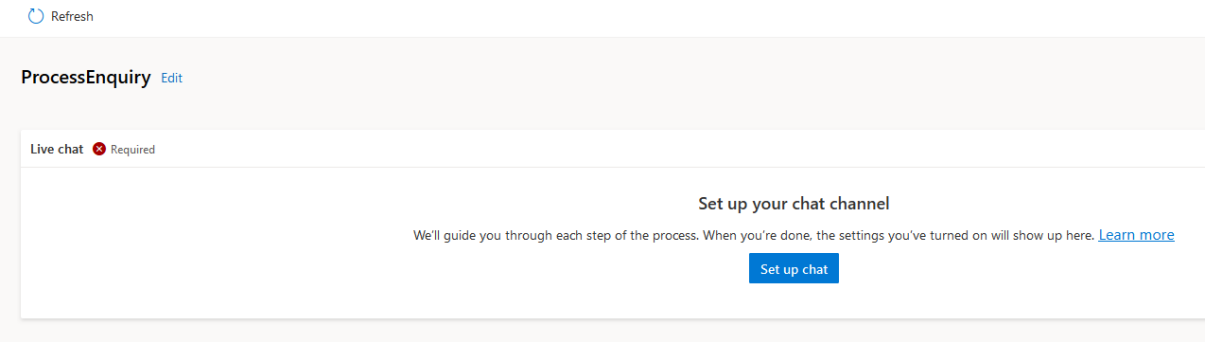


Create Workstream

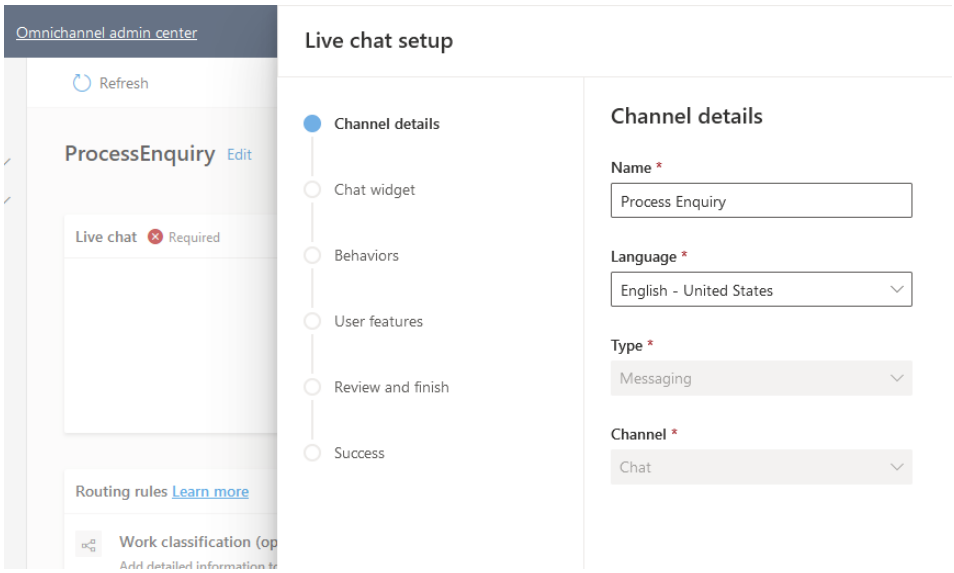
Now let's create a Workstream. A workstream acts as a container to route, and assign work items. From the Workstreams tab in the left pane, click on New Workstream to create a Messaging Type Workstream. In the Channel, we will select Chat as we will be handling chat based customer service. It provides the option to setup Voice, Facebook, WhatsApp etc channels as well.



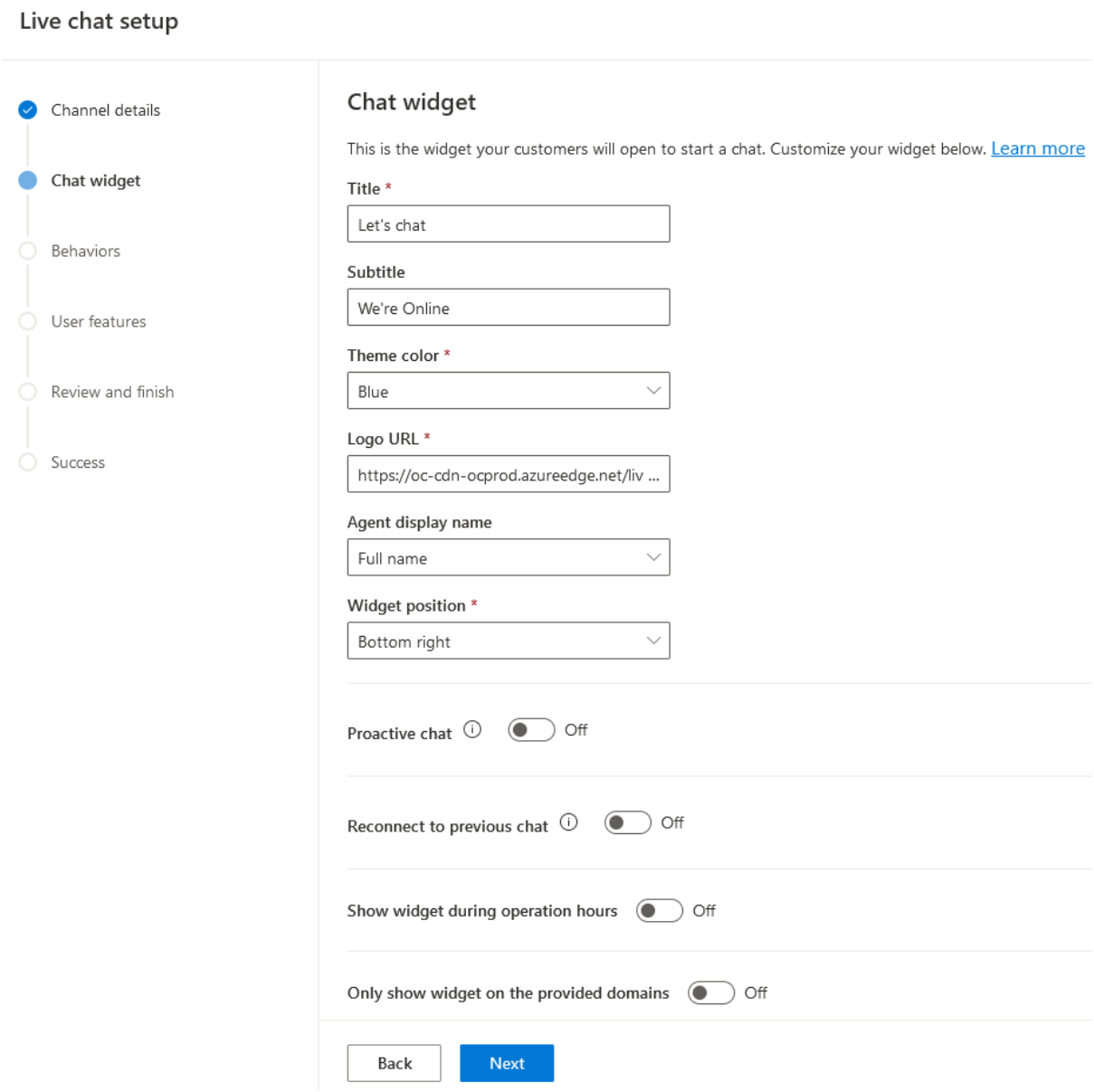
Now that the Workstream is created and clicking on the newly created workstream will open up the below page where we can configure the chat that we will be using for interacting with the users.



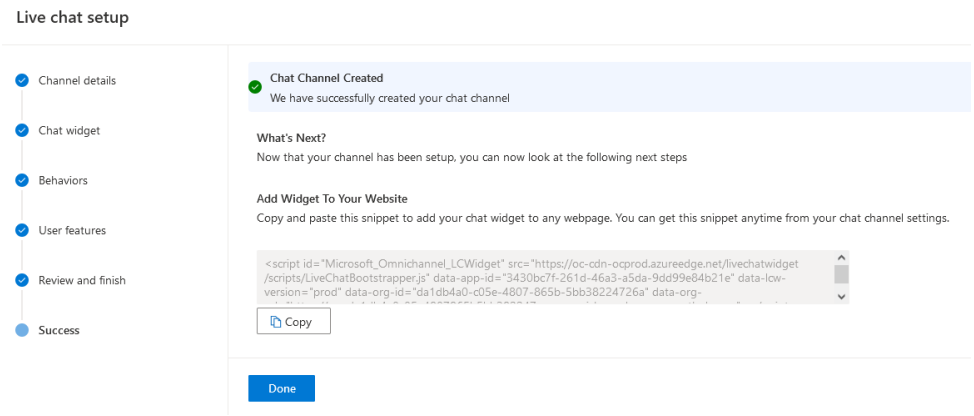
Click on setup chat and give a name to the Chat Widget.



Subsequent pages lets you define the chat widget behavior and features. For time being we will keep the defaults and proceed to the next pages.

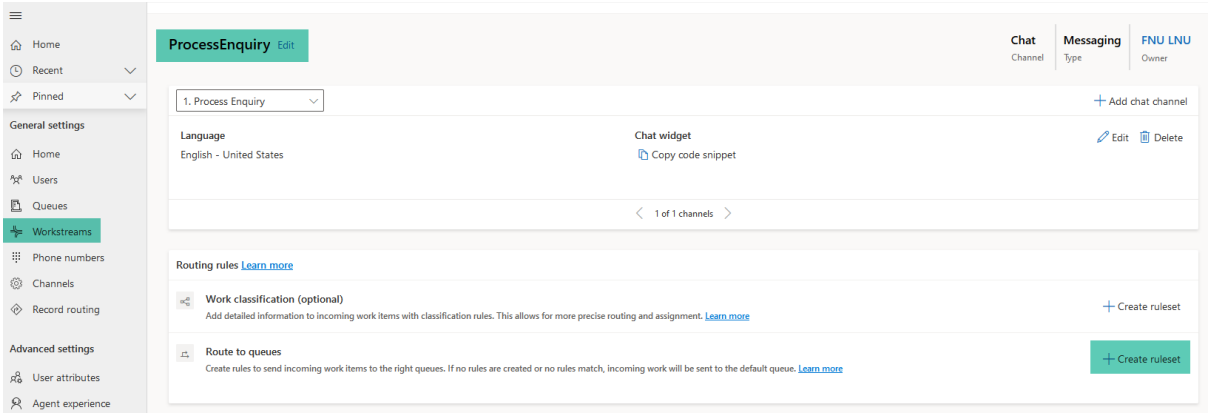


Thus we have finished the setting up of the Chat Channel within the Workstream. It will also give us a HTML script tag that can be used to plug and play the chat widget in any webpage. Copy the script and we will test it out in a while.

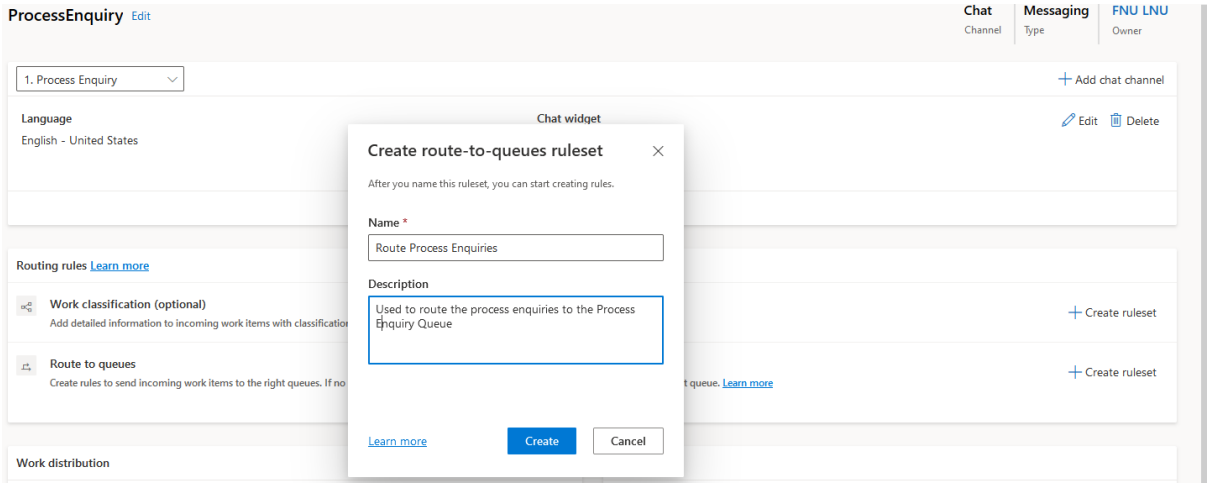


Add Ruleset

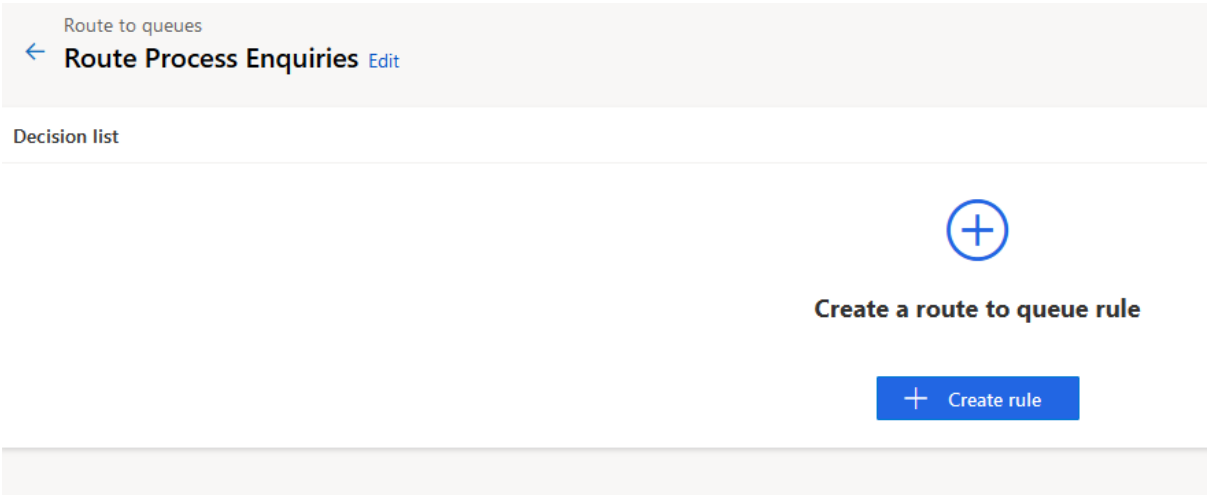
We have to make few more configurations in the workstream, let route the requests to a queue so that whenever a user initiates the chat, it goes to the queue for service.



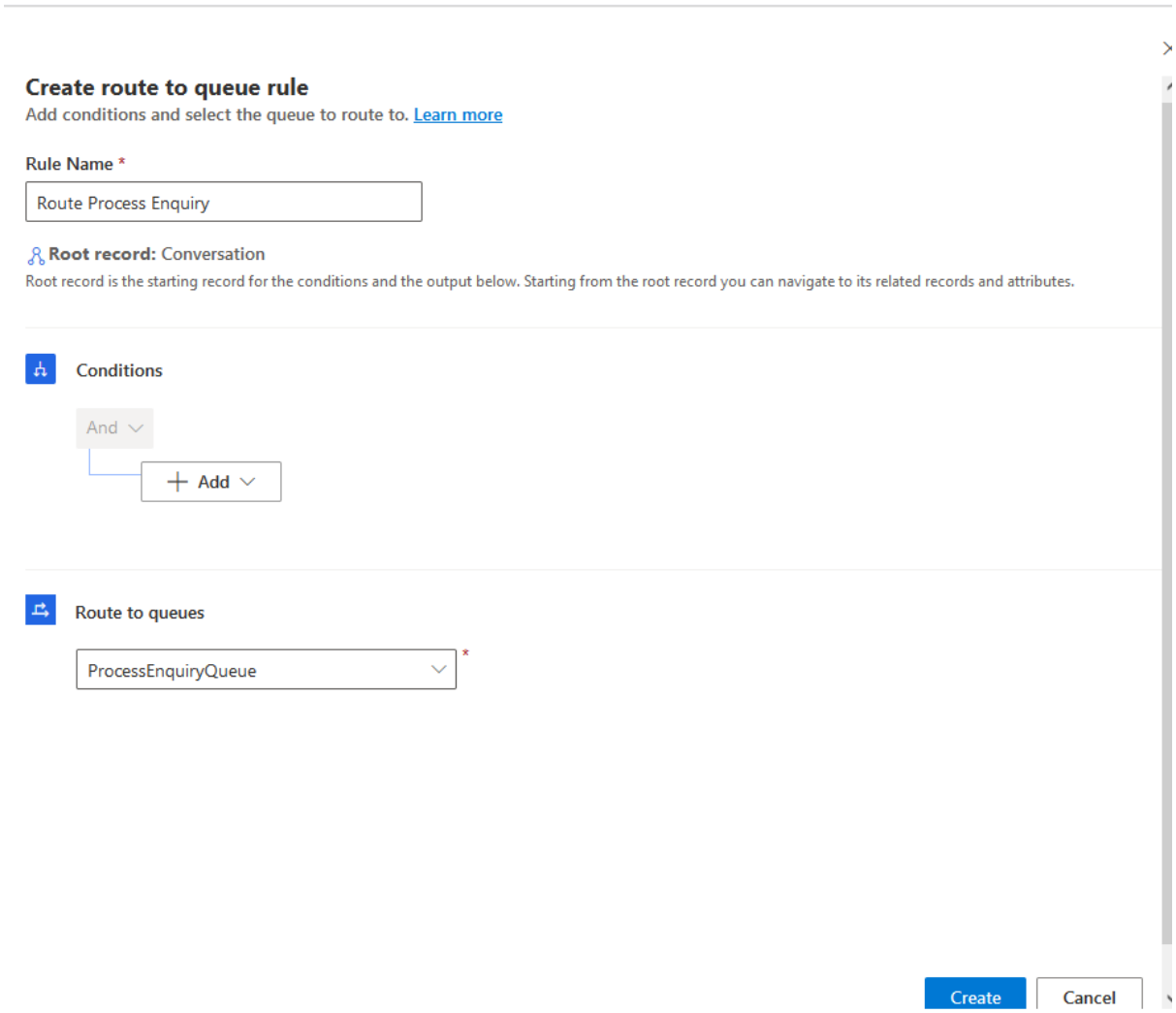
Click on Create ruleset to define the routing. Give the ruleset a name and description



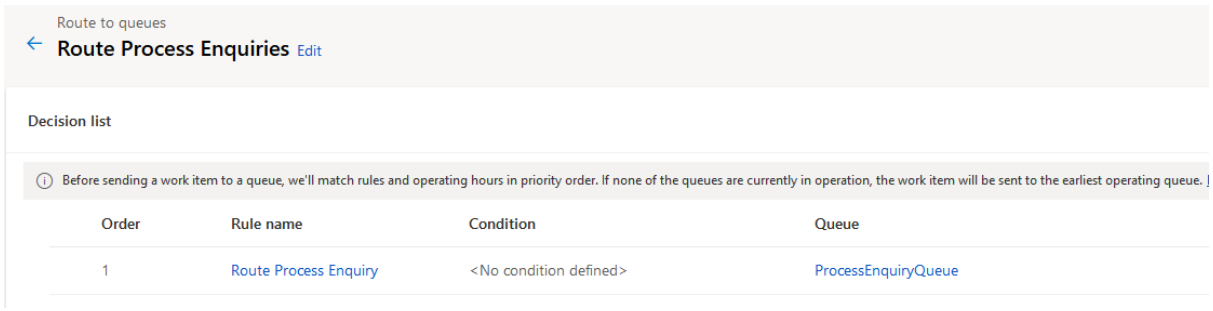
Click on Create to provision the ruleset. Now we can add the routing rule inside the ruleset



We can specify conditional routing to control the routing to different queues based on the conditions. For now, we will do a simple routing to the Process Enquiry Queue that we had created previously so that all conversational requests go to it.

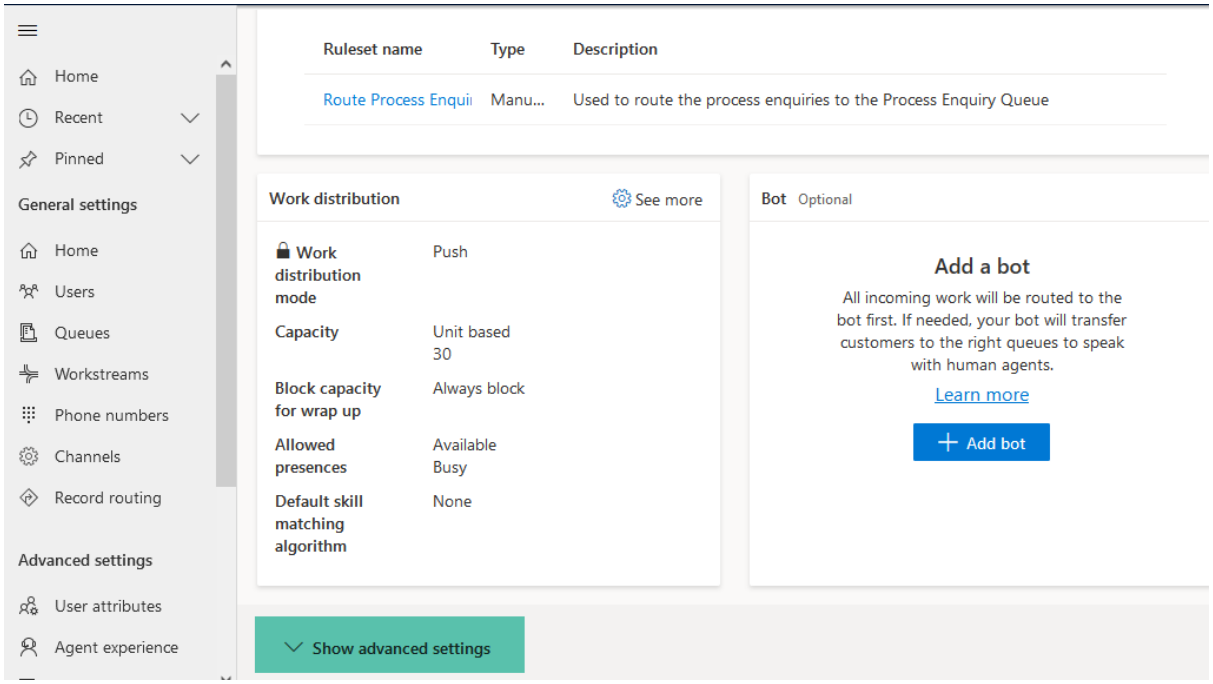


The ruleset is created

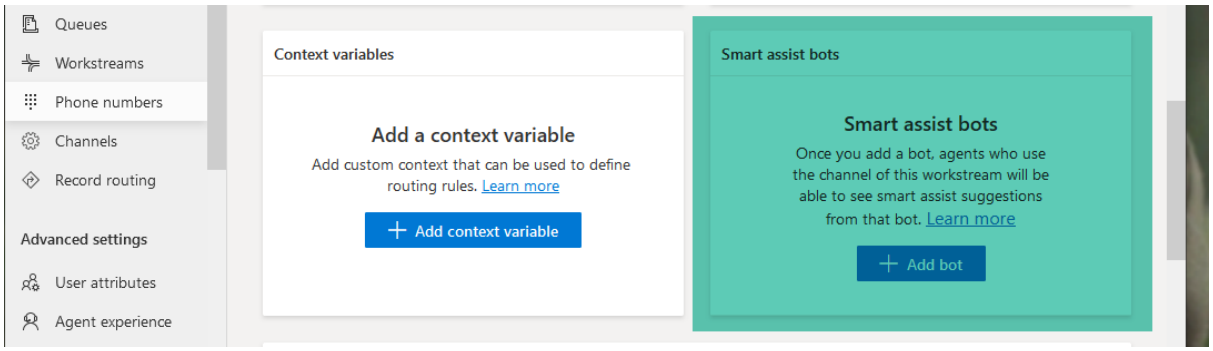


Add Bot to the Worksteam

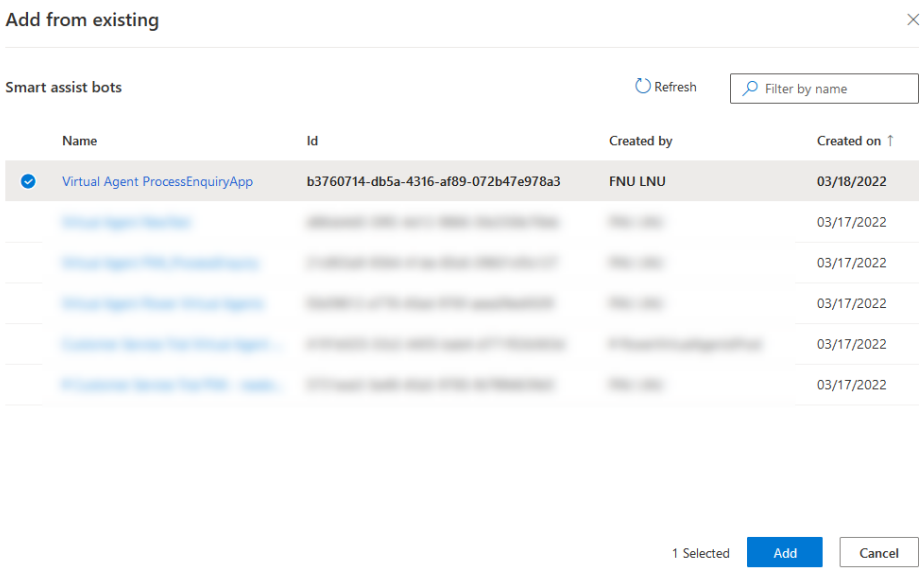
Lets head back to the workstream and add the Power Virtual Agent that we had created to the Workstream so that it will handle the conversations before handing it over to a human agent. In the Workstream, at the bottom, click on Show advanced settings.



Click on Add bot to select the Power Virtual Agent that we had connected to the Omnichannel.

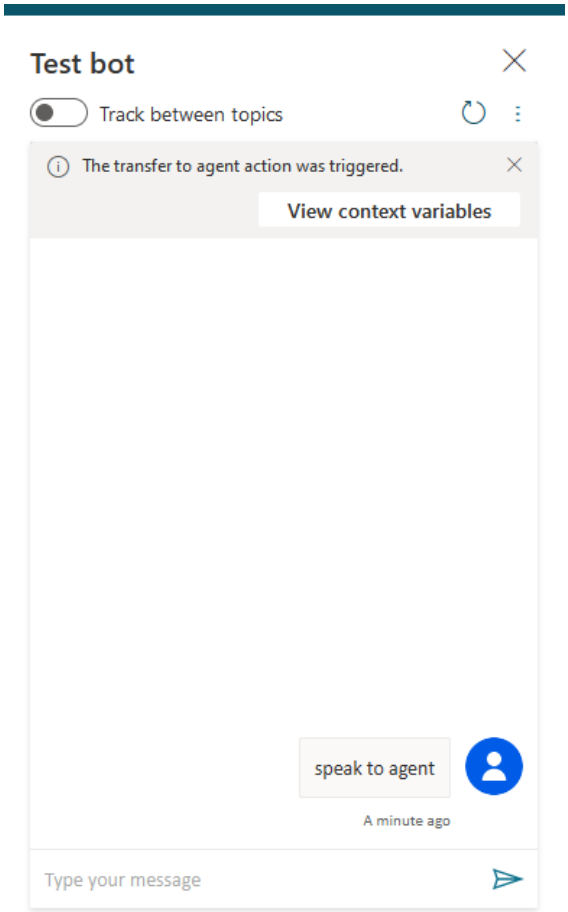


It will list the available bots. Select the bot that we would like to add to this workstream and Click on Add.

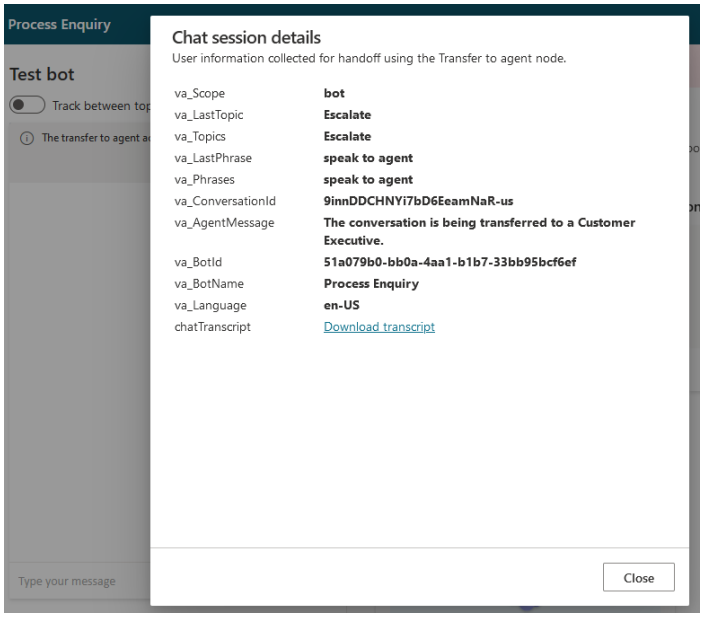


Add Context Variable

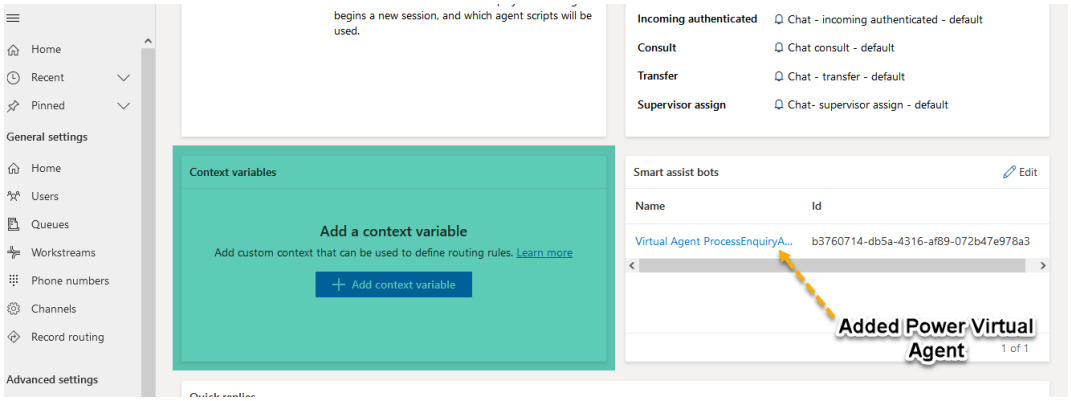
While testing the Power Virtual Agent and handing over the conversation to the agent. It will pass a set of variables from the bot which we can view by clicking on View context variables



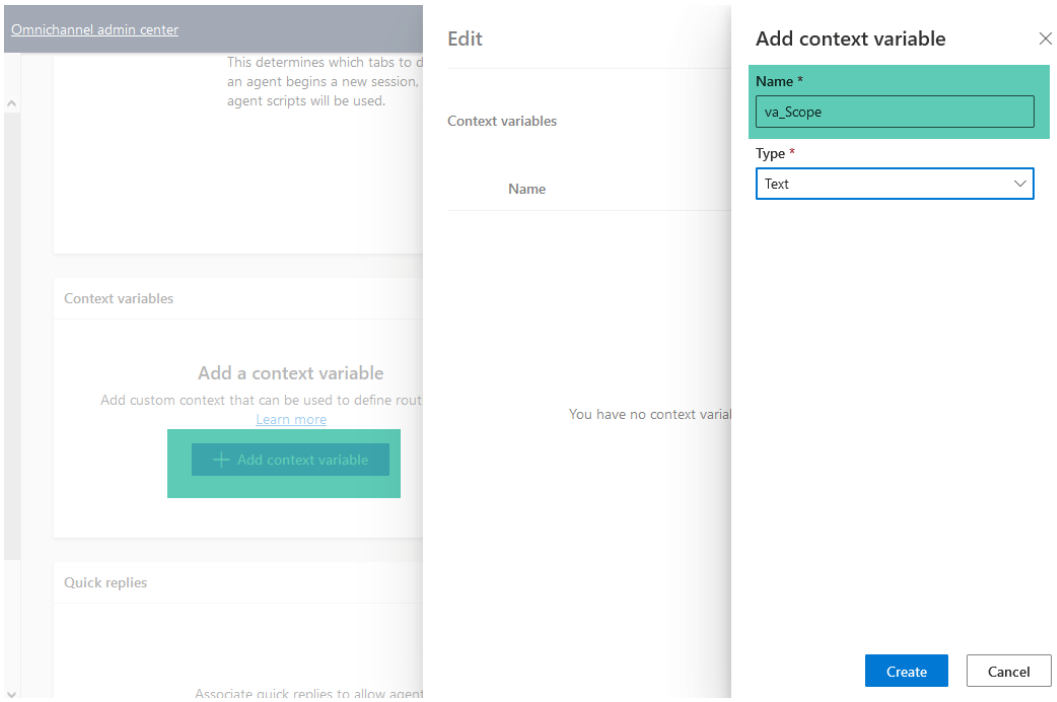
Clicking on View Context variables shows the details of the variables



We can access these variables by defining them in the Context Variables section in the WorkStream



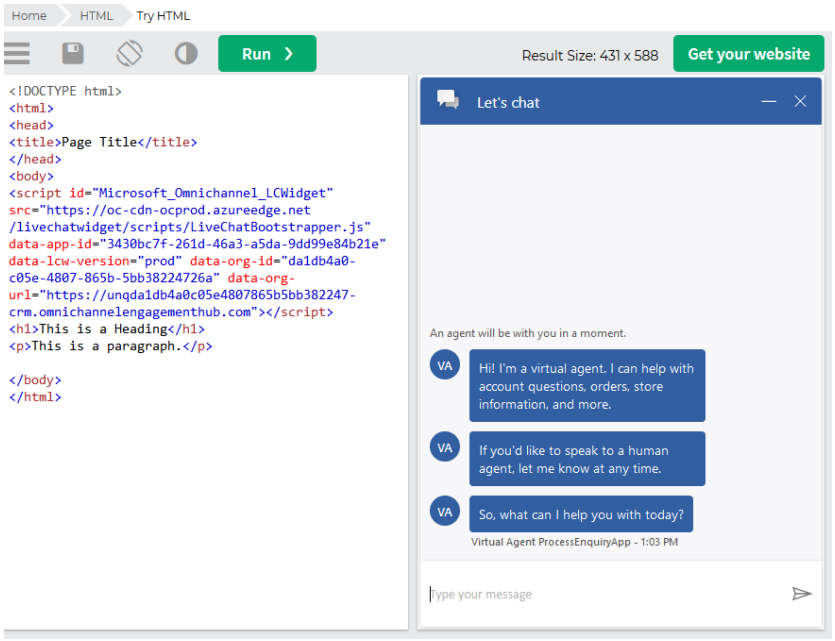
Click on Add Context variable to add the variables by the same name as provided by the bot and click on Create.



Thus we have completed the configurations within the Workstream .

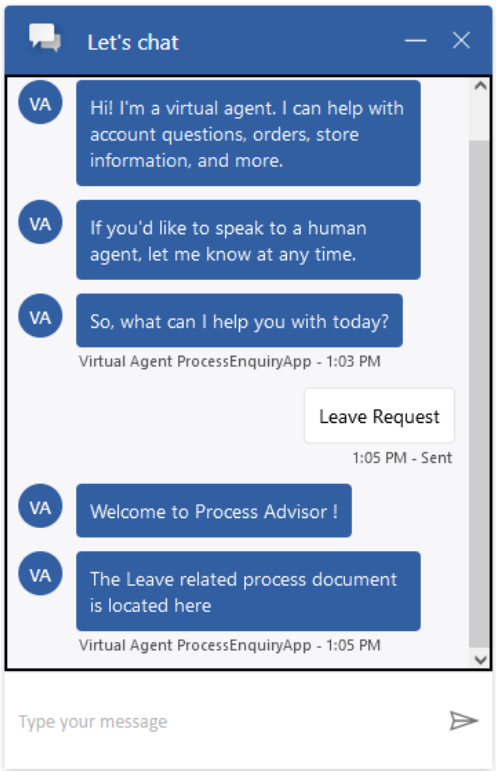
Test the implementation

Now lets see how the Power Virtual Agent and Omnichannel Agent hand over works. We will copy the chat widget script to a webpage to test this. A readily available online HTML editor like W3schools could be a good testing ground.

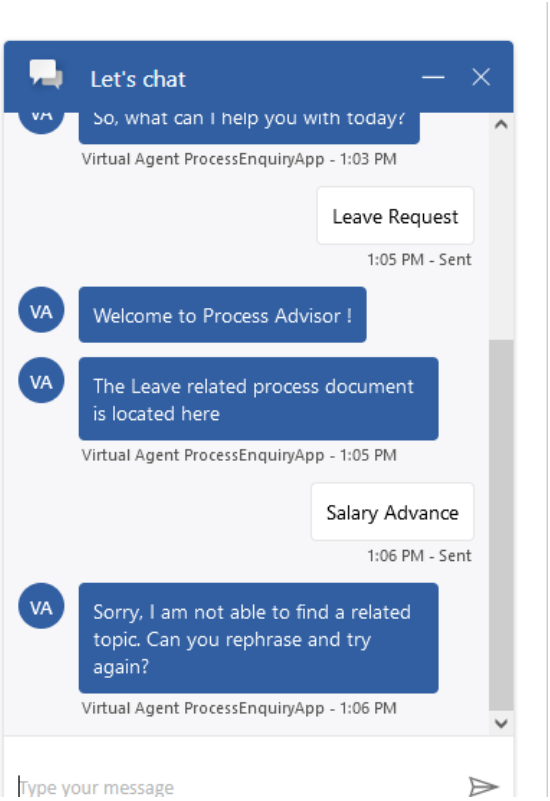


Adding the Chat Widget Script to the body of the html and running it will open up the Omnichannel where we have the Power Virtual Agent connected to it.

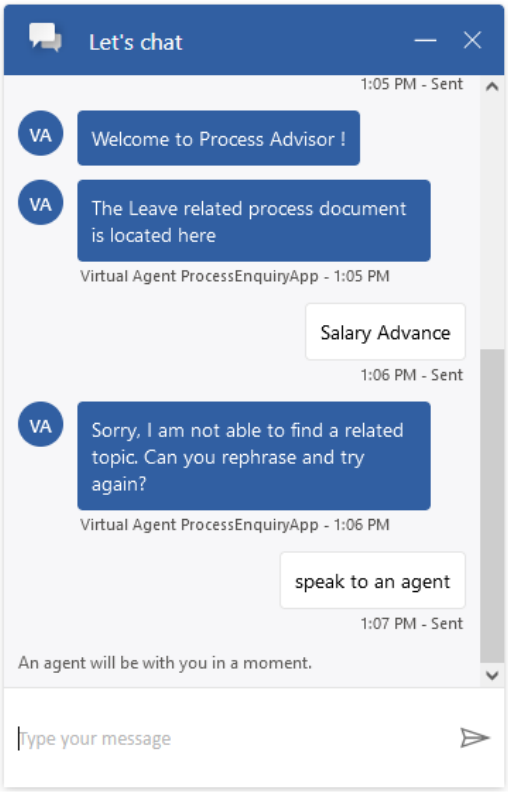
Lets type in a trigger word that will invoke the Power Virtual Agent branch for showing the leave documents



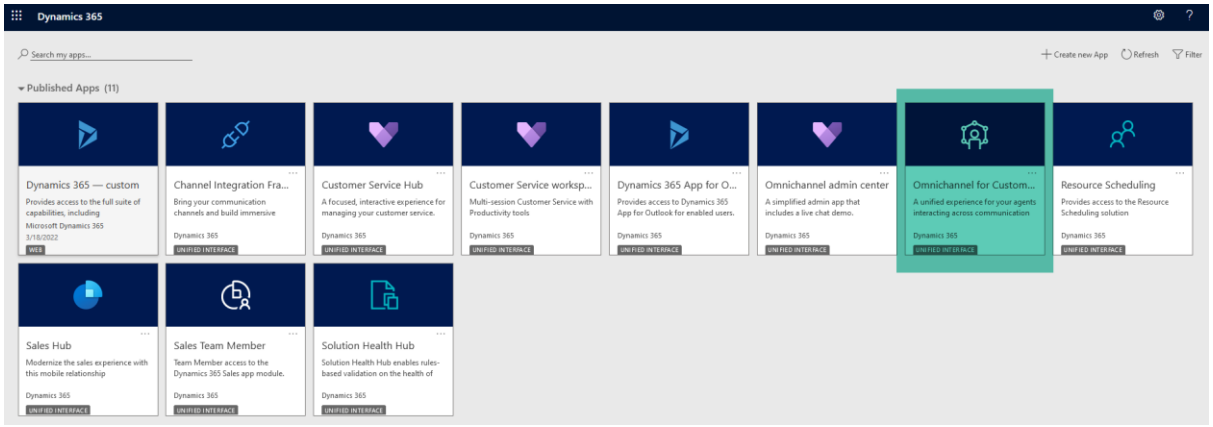
Now lets type in a query which our existing topics cannot handle.



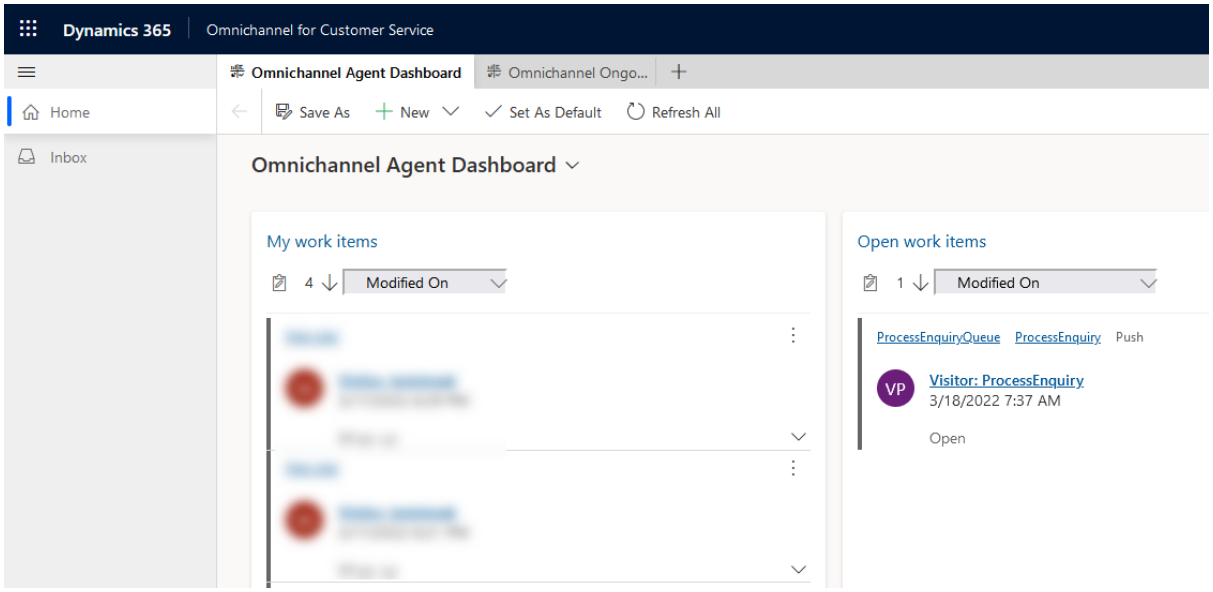
It will ask us to rephrase the query. So lets try to speak with a human agent. It has handed over the request to the Omnichanel agent



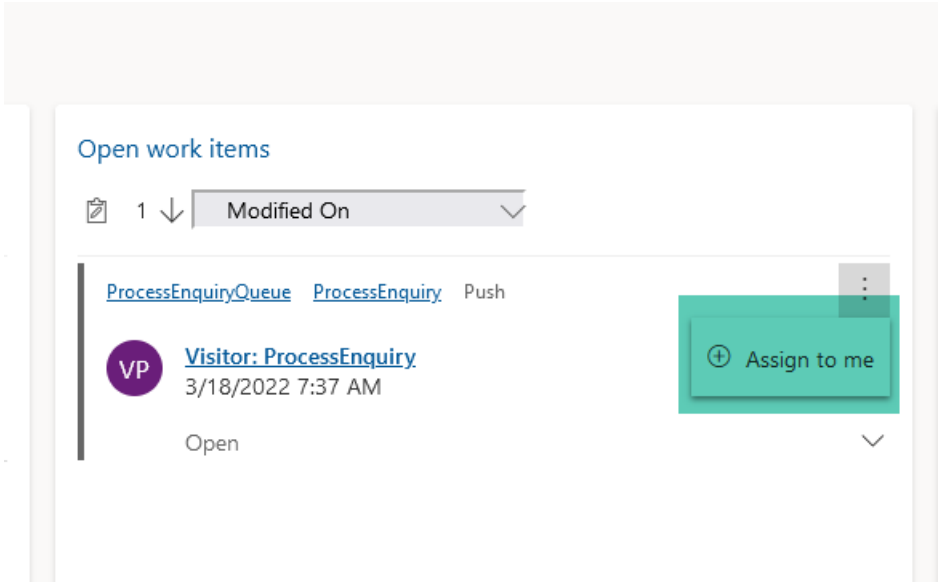
Now lets head over to the Omnichannel agent dashboard by going to the https://<tenant>.crm.dynamics.com/apps and selecting Omnichannel for customer service



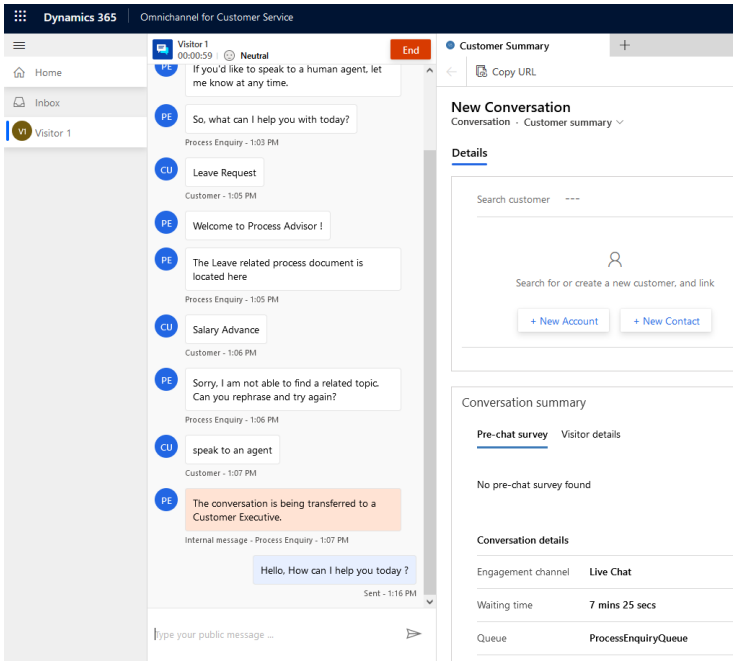
We can see that in the Open work items, it has come up.



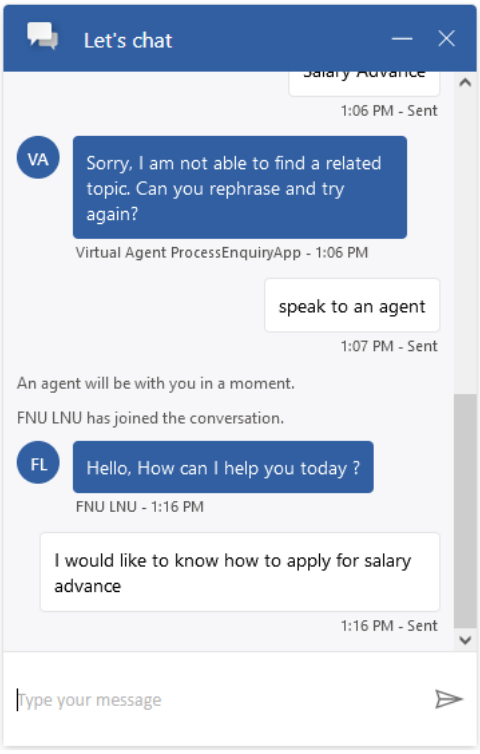
The agent can based on his availability, assign it to himself



This will open the chat between the agent and the end customer.



It will instantly show the agent name who has joined the chat and the Power Virtual Agent has successfully handed over the conversation to the Human agent and they can converse and take it forward.



Summary

Thus, we saw how to setup a handover between Power Virtual Agent and Omnichannel in Dynamics 365 so that when the end user is not able to find the needed details in the topics defined, the conversation can be handed over to the human agent.