Wellington Global Power Platform Bootcamp 2023

# End to end scenario

We are implementing an end-to-end service that takes a conversation recording in WAV passes it to a Azure service to transcribe it and store the returned sentences in Dataverse.

For each of the sentences, we will run an AI Builder model to detect the sentiment and for the general conversation, we will run ChatGPT to detect sentiment and reason for the sentiment.

Finally, we will display the sentiment results in a Canvas app.

## What is out of scope

Capturing the recording and formatting correctly to work with Azure cognitive services.

Creating an Azure function that accepts a link to the recording and transcribes it.

The resources above will be provided.

# Instance details

## App URL

<https://gppb23.crm6.dynamics.com/main.aspx?appid=9798ad91-d2ad-ed11-83ff-0022489804cd>

## Usernames

[abhay@SamplePages.onmicrosoft.com](mailto:abhay@SamplePages.onmicrosoft.com)

[ak@SamplePages.onmicrosoft.com](mailto:ak@SamplePages.onmicrosoft.com)

[linn@samplepages.onmicrosoft.com](mailto:linn@samplepages.onmicrosoft.com)

[mac@SamplePages.onmicrosoft.com](mailto:mac@SamplePages.onmicrosoft.com)

[user1@samplepages.onmicrosoft.com](mailto:user1@samplepages.onmicrosoft.com)

…

[user18@samplepages.onmicrosoft.com](mailto:user18@samplepages.onmicrosoft.com)

**password** Gppbnz23

# Session 1 – Create a data model and a model driven app in Dataverse

**Rami**

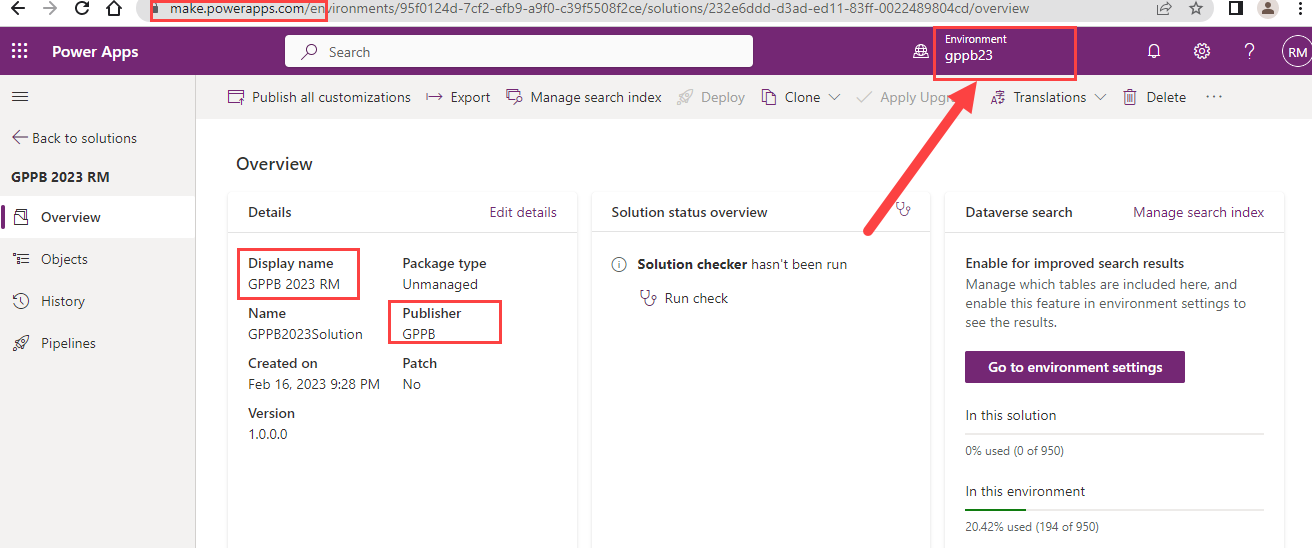
## Goals

1. Create a new solution
2. Create a data structure
3. Create a model-driven app to display the results



## Solution

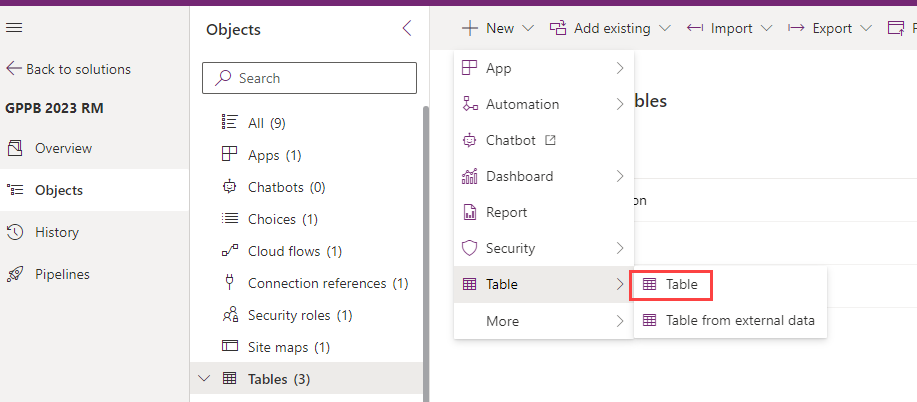
* Navigate to make.powerapps.com
* Login with one of the user accounts
* Ensure you are using gppb23 not the default app
* Create a new solution with your initials (e.g. gppb23\_rm) select gppb as the publisher



## Data Model

Build the schema for the following tables

* Conversation
* Sentence with a lookup to conversations



### Create new columns



### Conversation

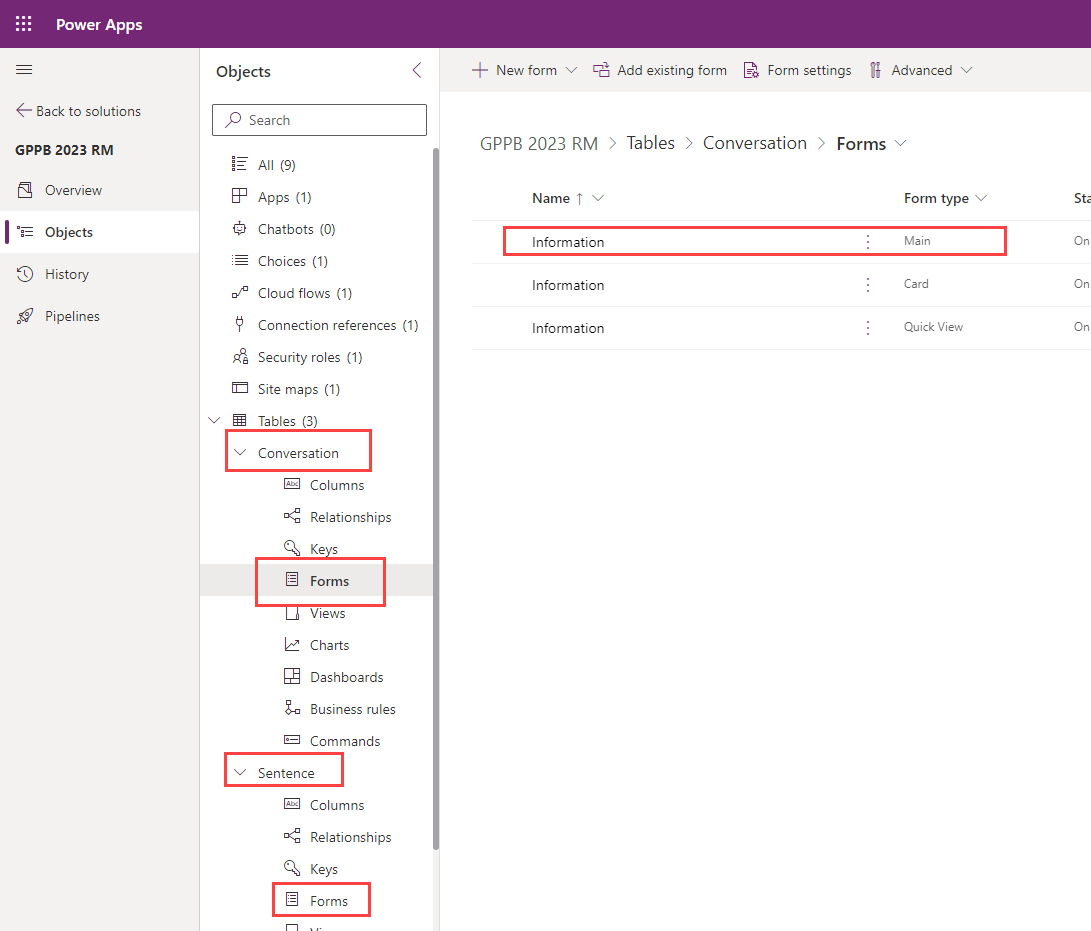
|  |  |
| --- | --- |
| Field Name | Data Type |
| Start Time | Date Time |
| Recording | File (32MB) |
| Name | Text |
| Sentiment | Global Drop Down (positive, negative, neutral) |
| Sentiment Text | Text 100 |
| Sentiment Analysis | Text 4000 |

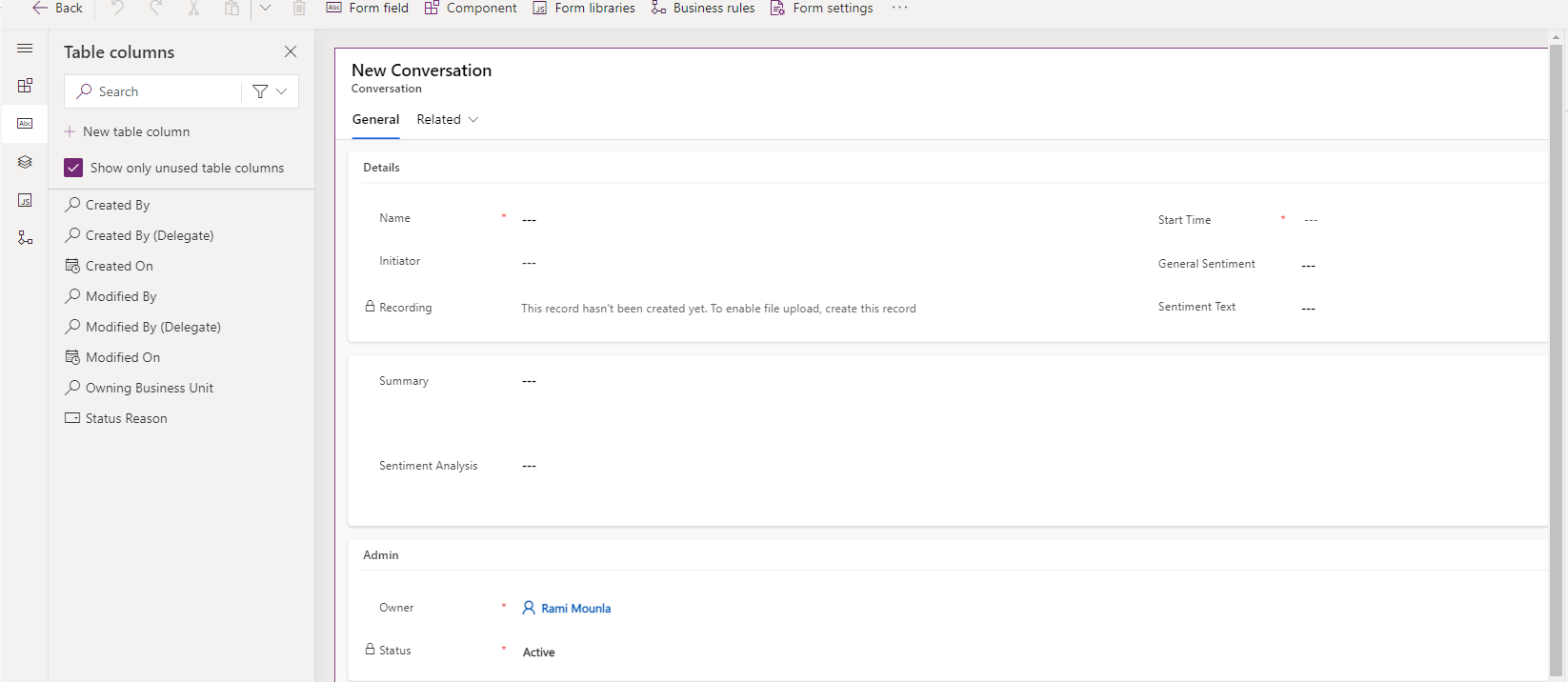
### Sentence

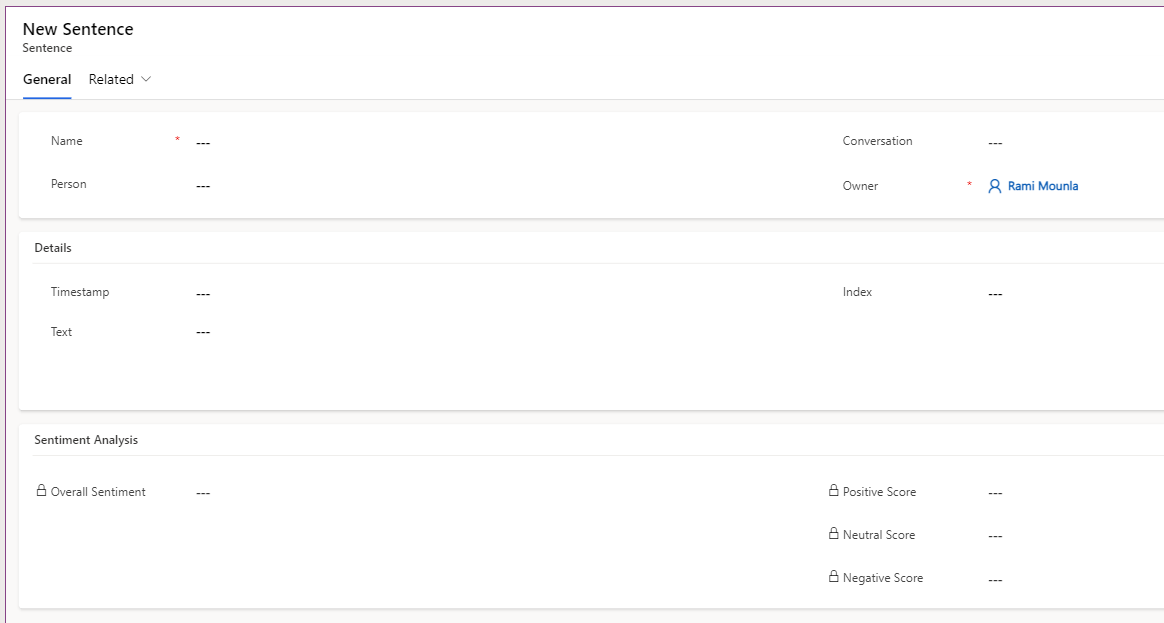
|  |  |
| --- | --- |
| Field Name | Data Type |
| Name | Text 100 |
| Text | Text 4000 |
| Sentiment | Global Drop Down (positive, negative, neutral) |
| Conversation | Lookup |
| Person | Text |
| StartTime | Number Int |
| Negative Score | Decimal |
| Neutral Score | Decimal |
| Positive Score | Decimal |
| Overall Sentiment | Text 100 |

### Configure forms and views

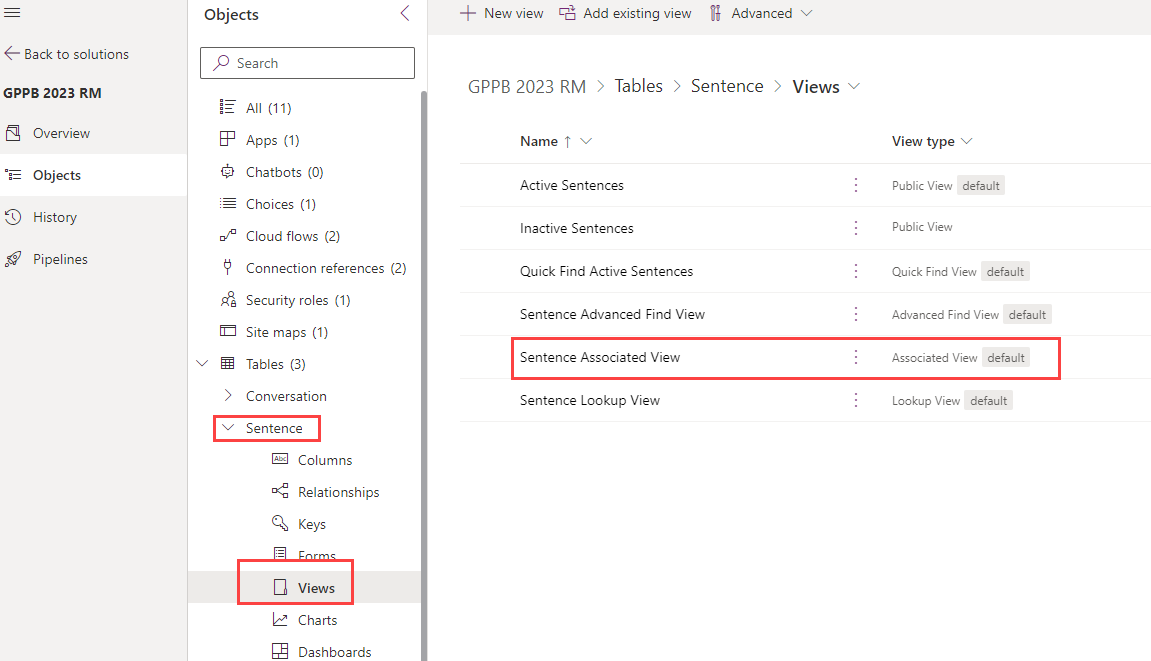
* Configure conversation and sentences forms

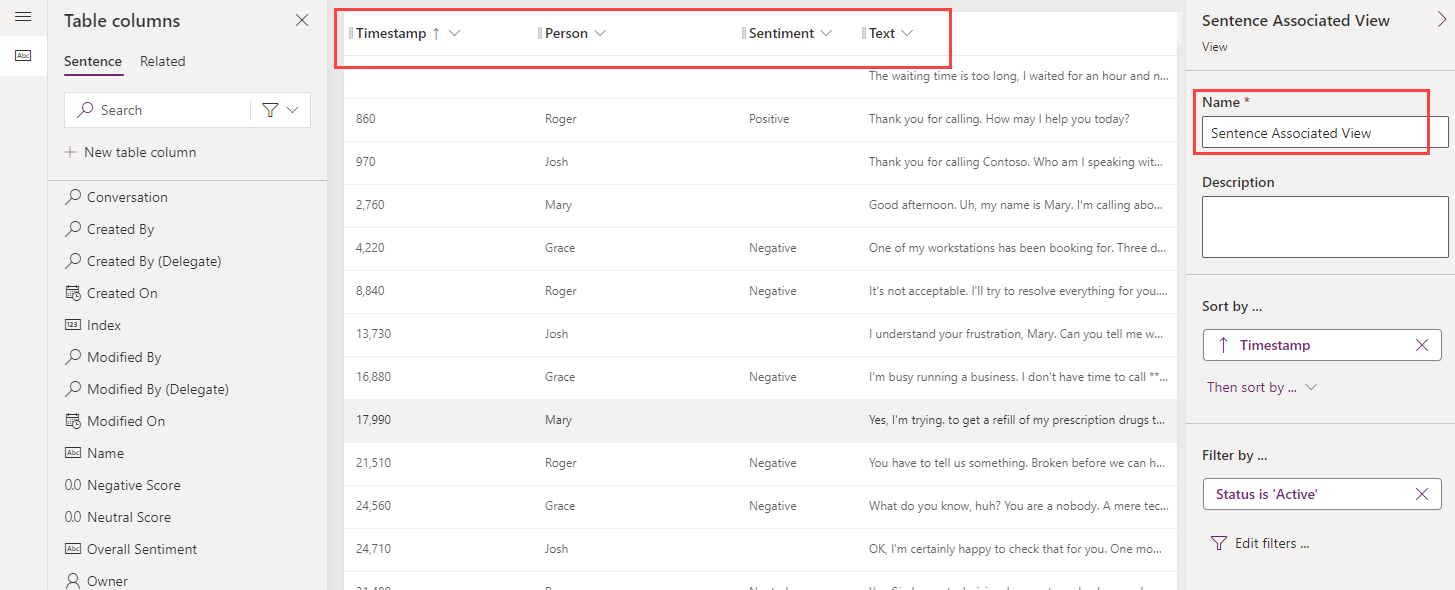






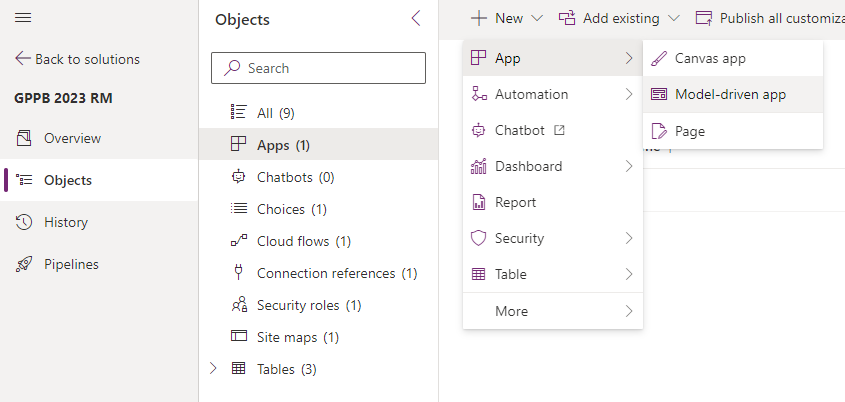
* Configure the sentences related view

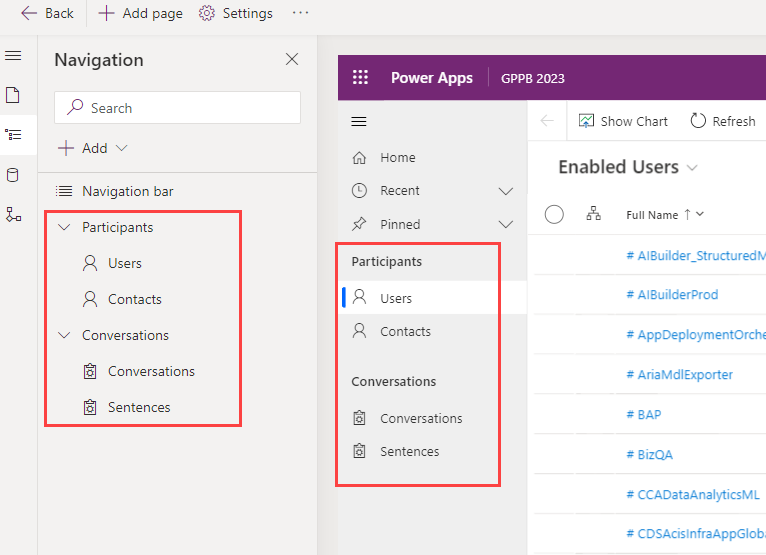




## Model Driven App

* Create the app with conversation and sentences
* Give the app a name with your initials





# Session 2 – Create an AI Builder model to detect sentiment

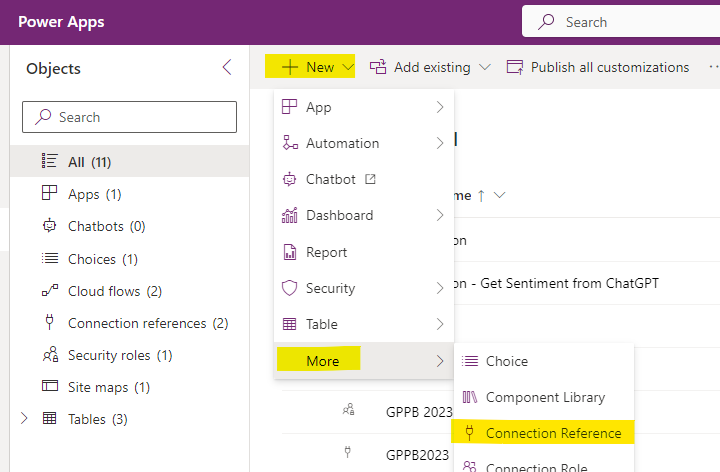
**Mary Ann Castro**

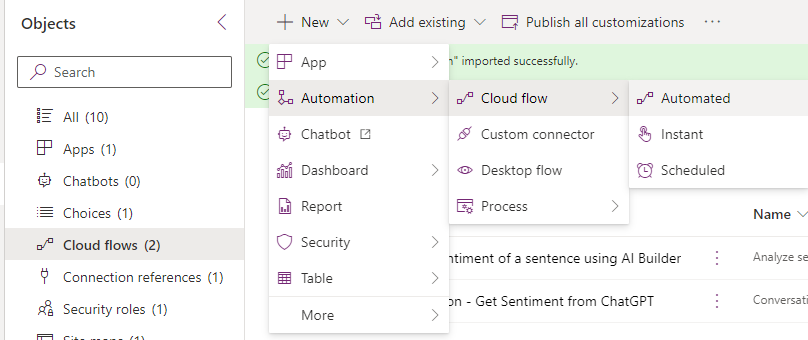
## AI Builder

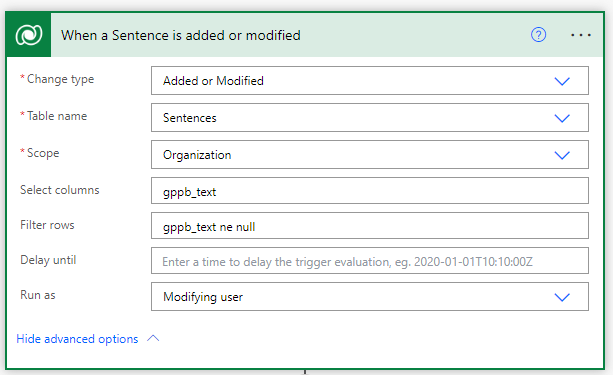
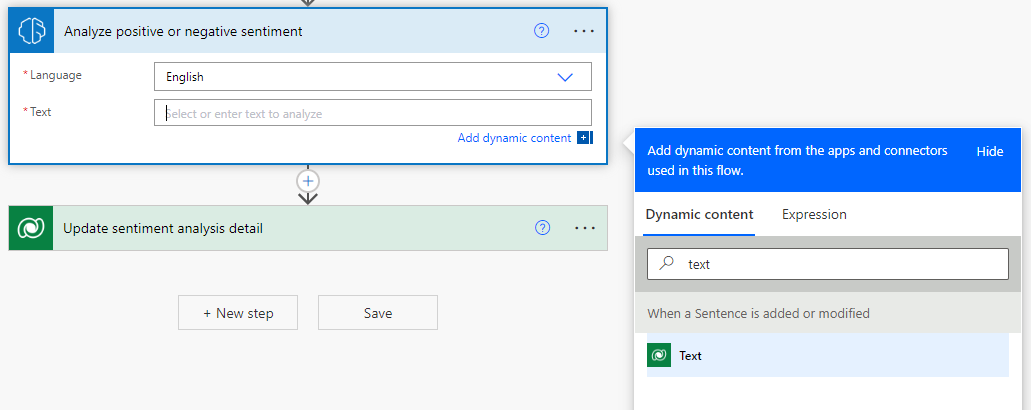
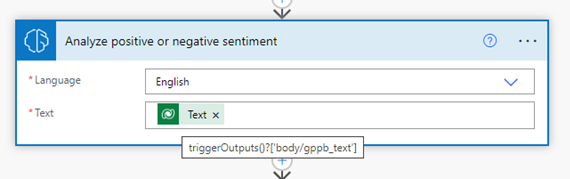
Go through the basics, create AI model that trigger on create of a sentence, passes the text and return the sentiment.

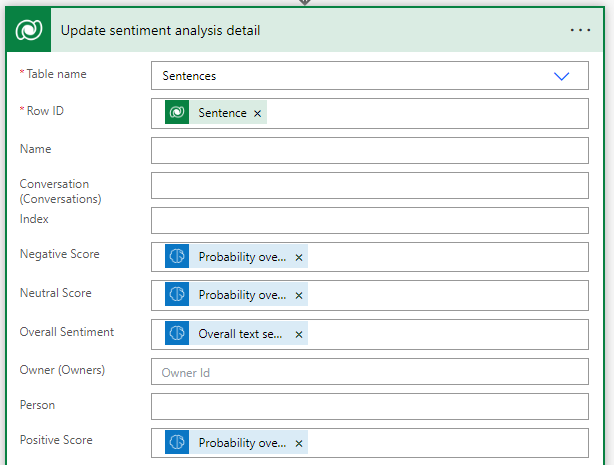
## Use the sentiment analysis prebuilt model in Power Automate

In this exercise we will create a cloud flow that will trigger on create or update of a Sentence record in Dataverse and call the AI Builder to analyse the sentiment of the text. The cloud flow will also write the sentiment analysis output back to Dataverse and update the Sentence record.

1. Login to <https://make.powerapps.com/> and navigate to Solutions you are working on.
2. Create a Connection Reference. Open the solution you are working on and click +New à More à Connection Reference   
   
3. Provide the Display name (i.e. GPPB 2023 – Microsoft Dataverse) and description and choose Microsoft Dataverse under Connector. Select an existing connection or select New connection to create one. Your connection reference will now be created to be used in the cloud flow that we will create in the next step.
4. Create the cloud flow. Select  +New again and go to Automation à Cloud Flow à Automated



1. Provide a flow name (i.e. Sentence - Analyze sentiment of a sentence using AI Builder) and choose the Dataverse trigger “When a row is added, modified or deleted”.
2. Add / update the trigger details as per screenshot below   
   
3. Click +New Step and look for AI Builder. Select “Analyze positive or negative sentiment in text” and add the “Text” column from the trigger output.   
      
   
4. Click +New Step and click Dataverse. Select “Update a row” and add the following output from “Analyze positive or negative sentiment” step as per screenshot below.

* Negative Score = Probability overall text is negative
* Neutral Score = Probability overall text is neutral
* Positive Score = Probability overall text is positive
* Overall Sentiment = Overall text sentiment

1. Save the flow and test.

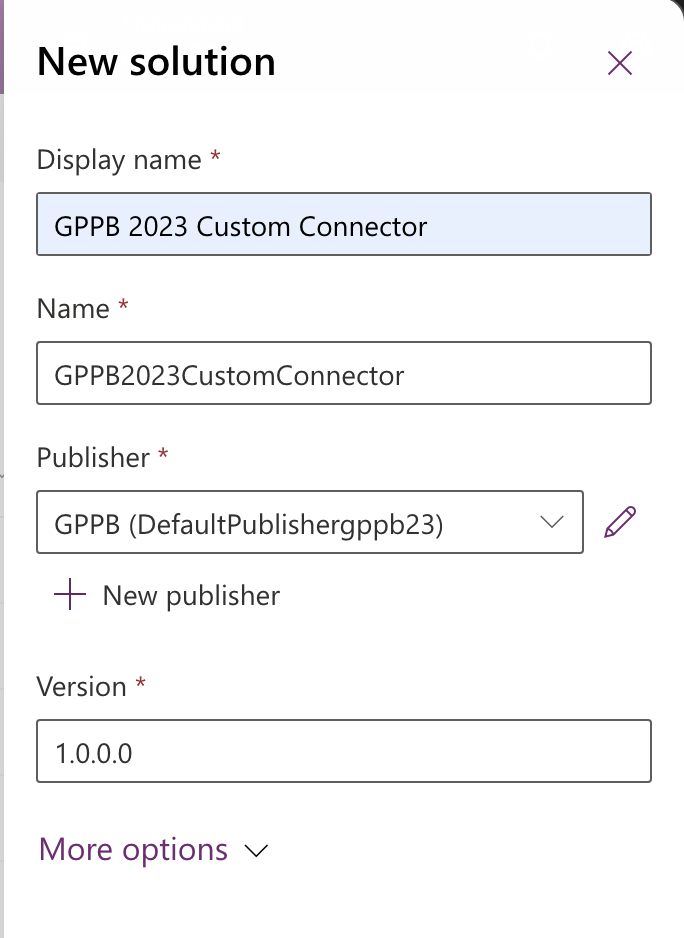
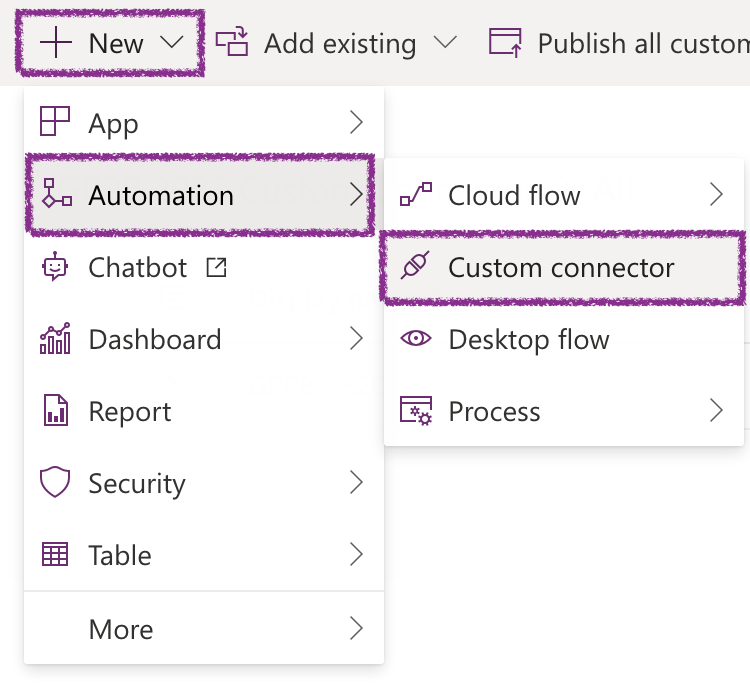
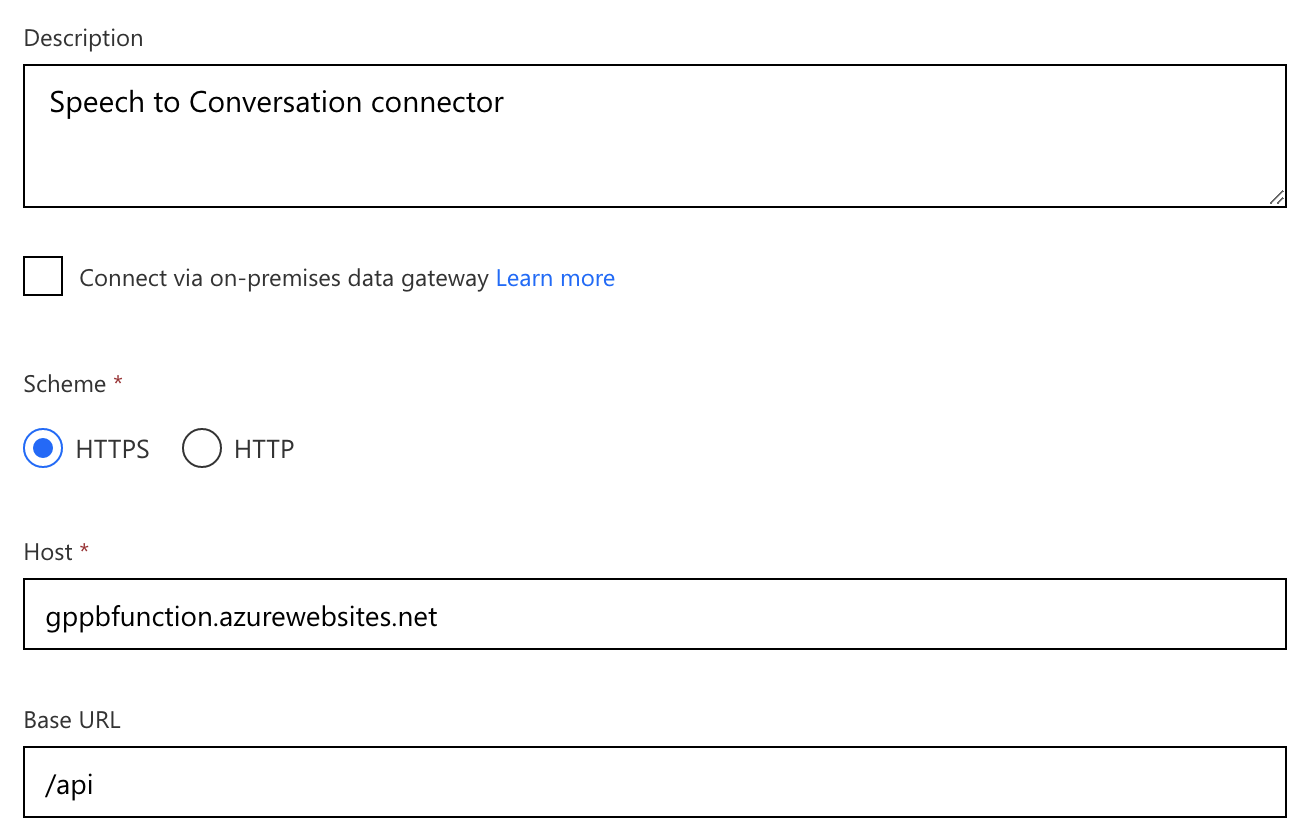
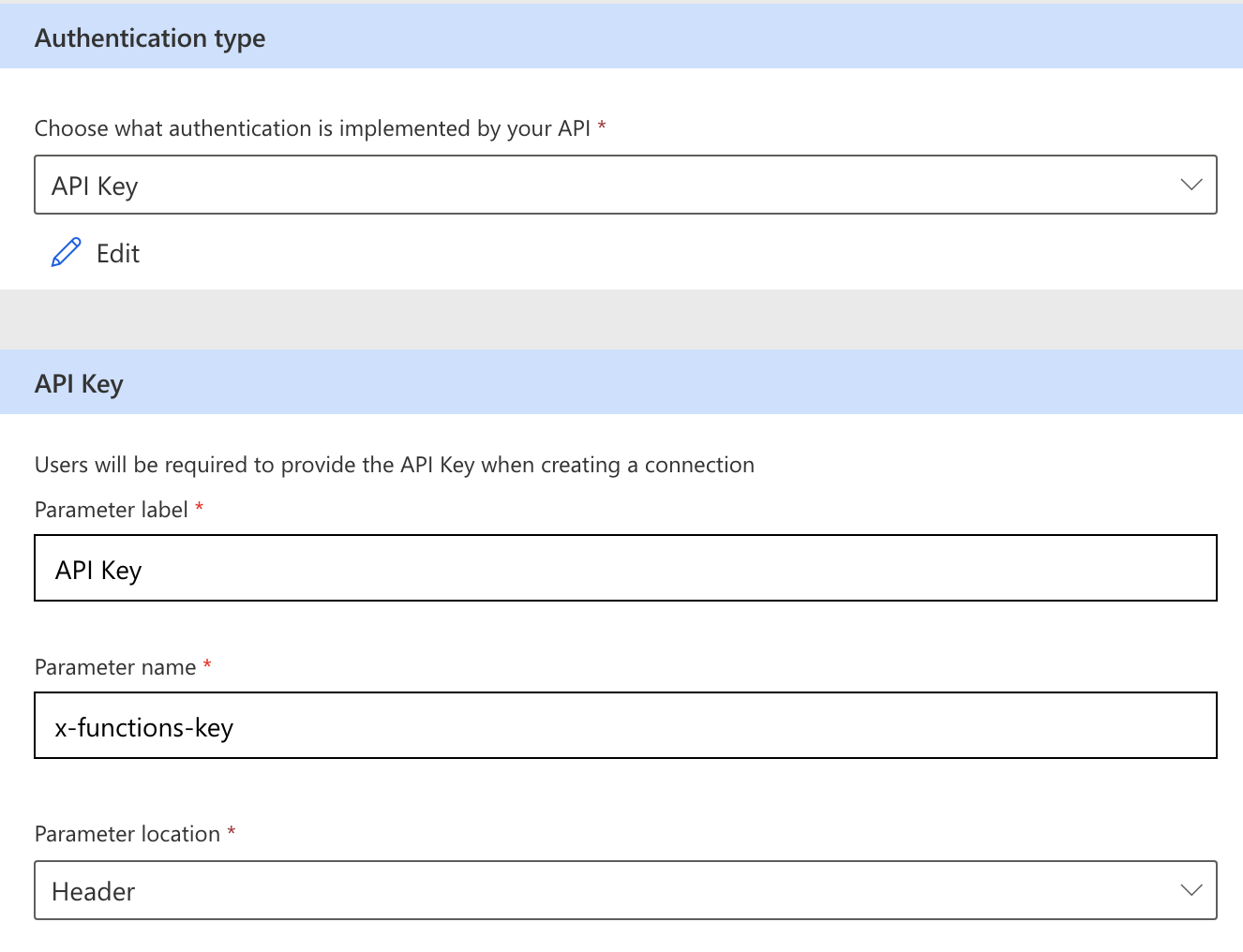
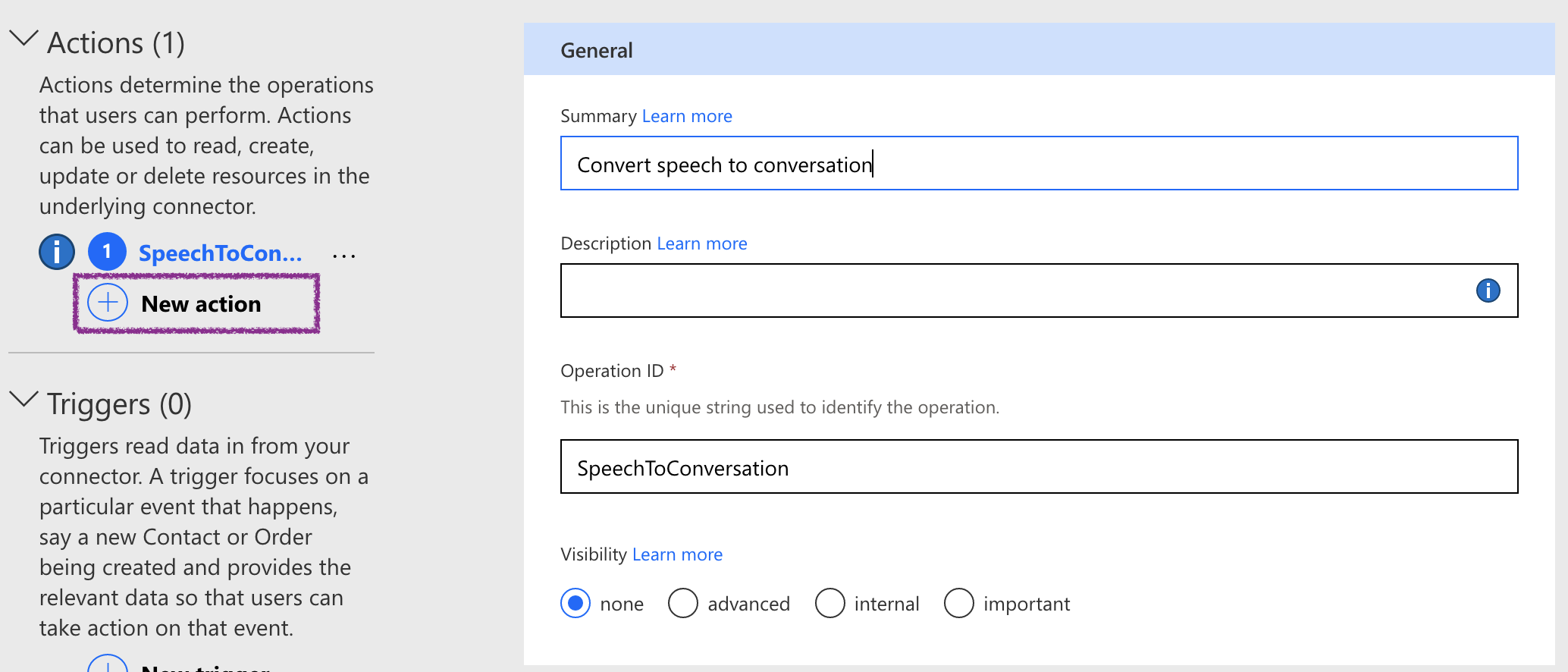
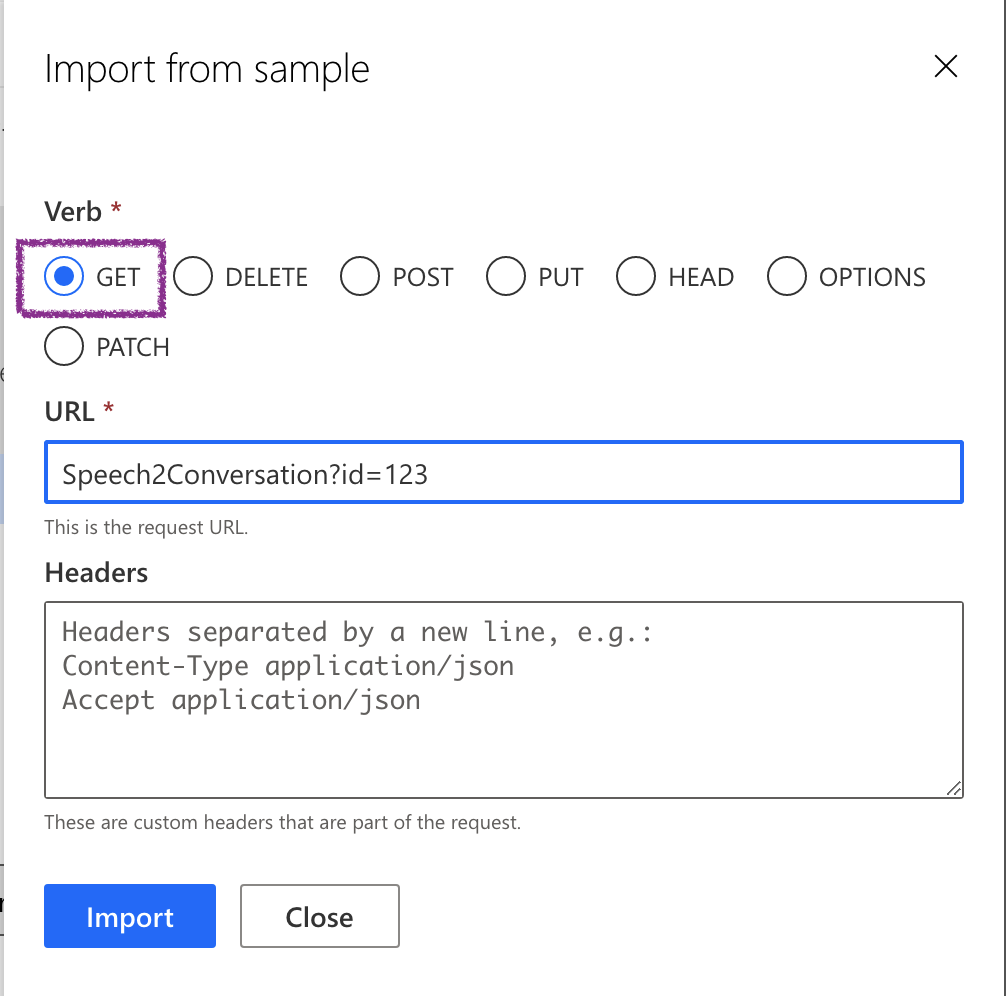
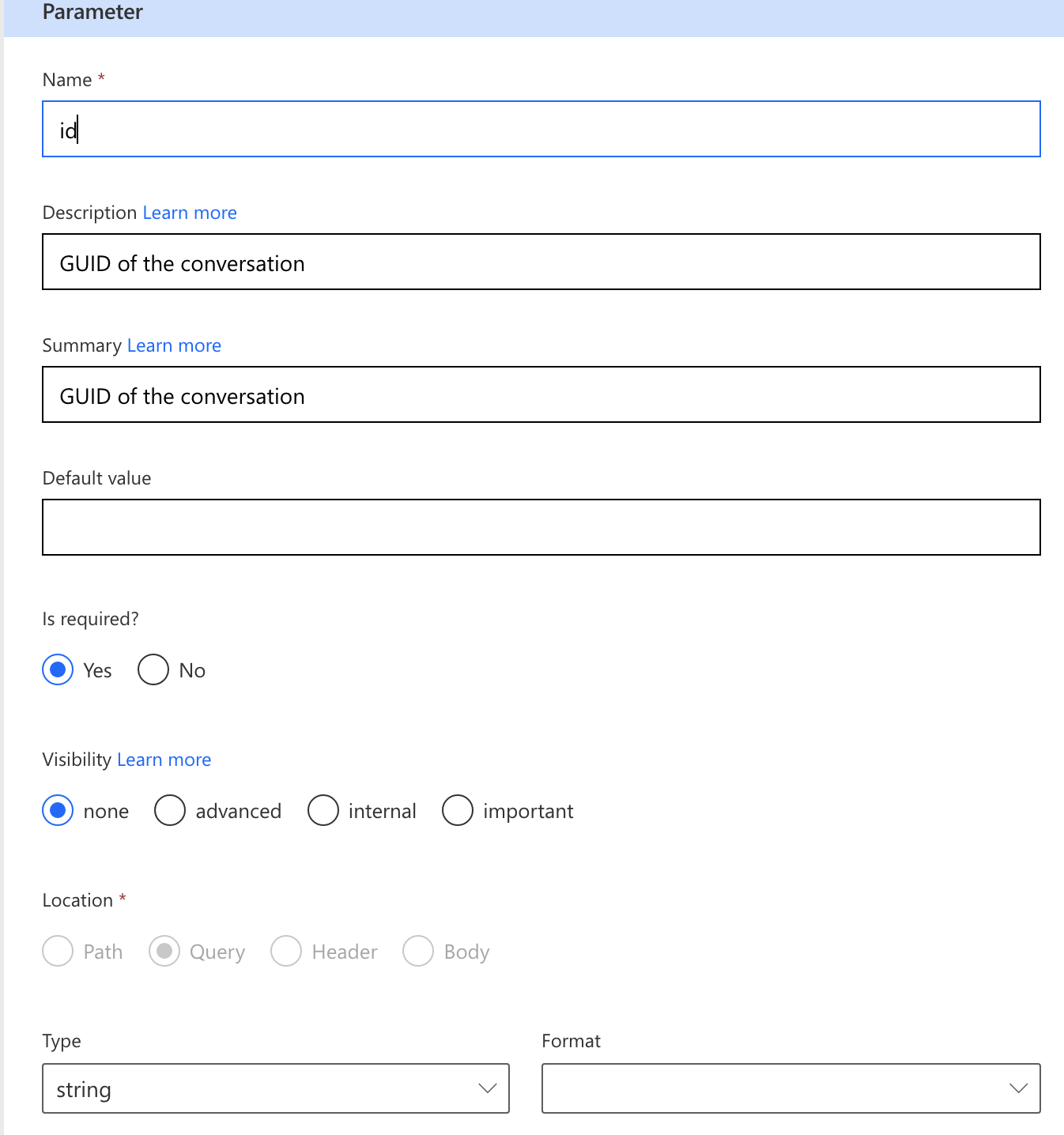
# Session 3 - Create a custom connector

**Aung Khaing**

Connector to Azure function (pre built) using authentication type (?)

Send binary (or GUID and pull the binary if it’s easier) return the sentiment.

## A basic custom connector: A step by step manual process

1. Go to <https://make.powerapps.com>. Ensure you are in the correct environment.
2. Create a new solution if there is no pre-existed solution for a custom connector.  
   
3. Create a new custom connector.  
   
4. Enter the description, host and base urls:  
   
5. The external API hosted in Azure Function Apps uses authentication key. The API key needs to be in the header.  
   Select **API Key** as authentication type. Parameter name should be same as the header which is accepted by the API.  
   
6. Add a new action. Enter summary and operation id. Note: if you have multiple actions, operation id must be unique.  
   
7. Add a request sample. Url is Speech2Conversation?id=GUID  
   Note: the external API currently accepts GET and POST. We need to accept **id** as a query string which is the GUID of a conversation record in Dataverse.  
   
8. You can fine tune the query string **id** properties such as required, visibility and type.   
   
9. Add a default response to the operation. This makes a custom connector more user friendly.   
   Note: Please use the following as body:  
   [

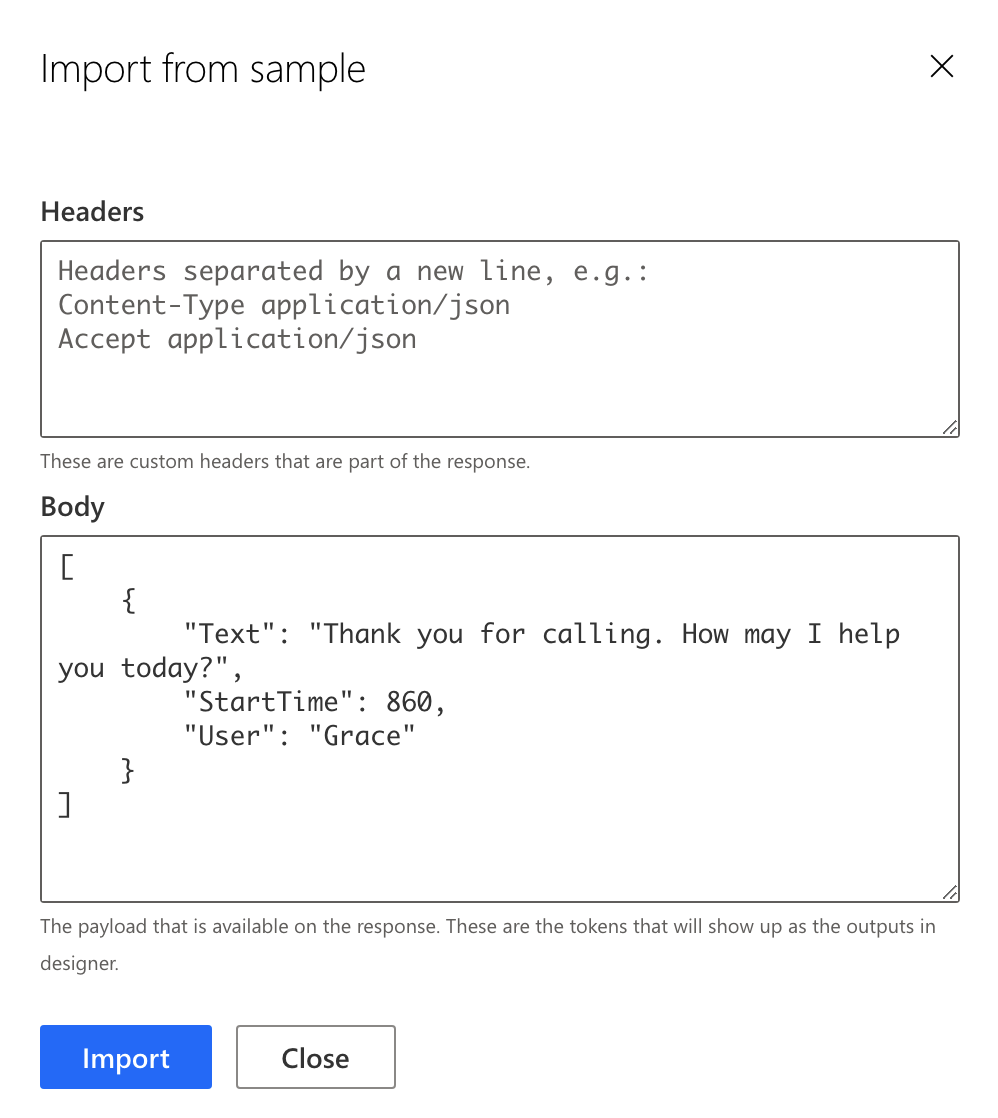
{

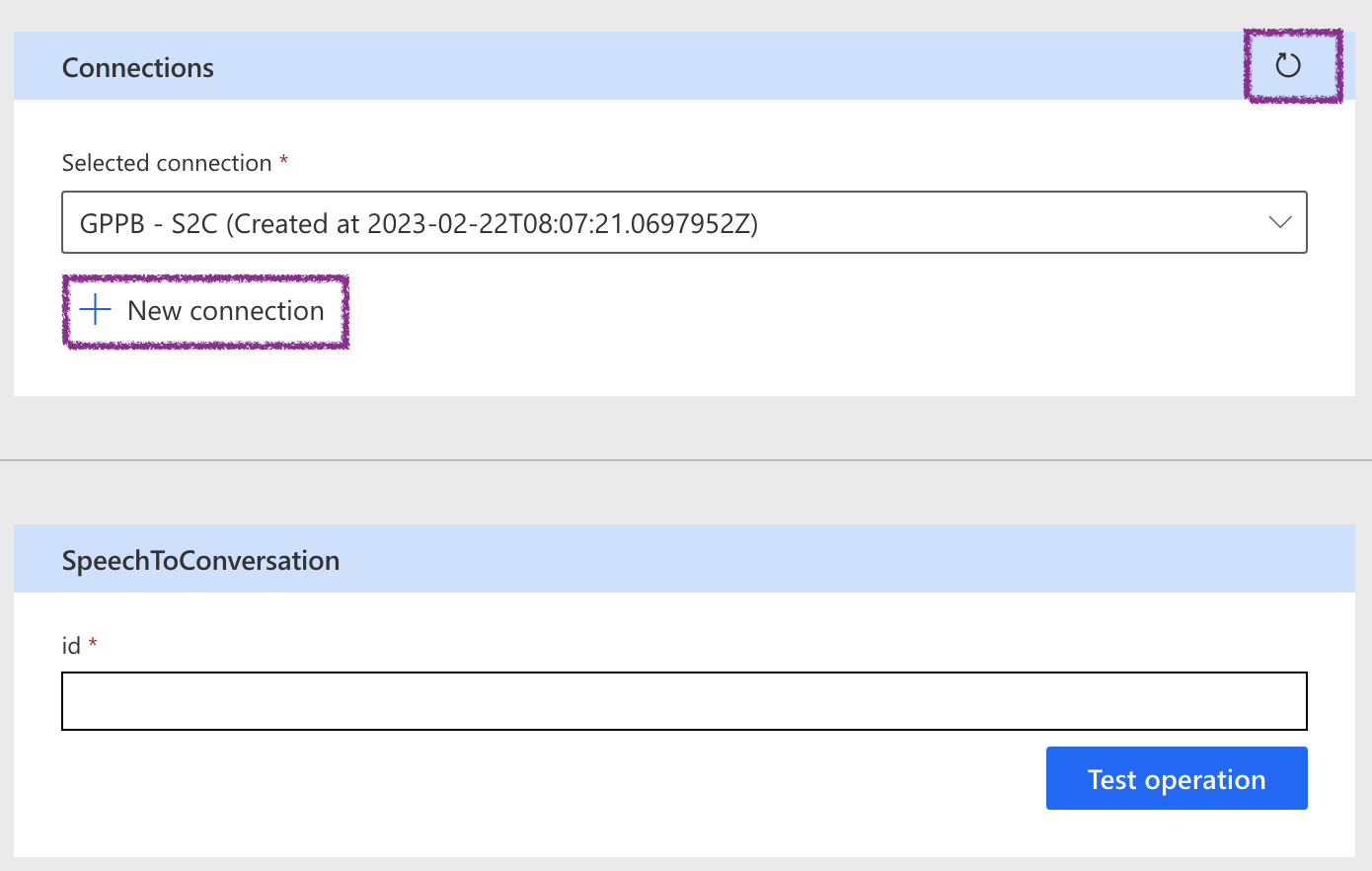
"Text": "Thank you for calling. How may I help you today?",

"StartTime": 860,

"User": "Grace"

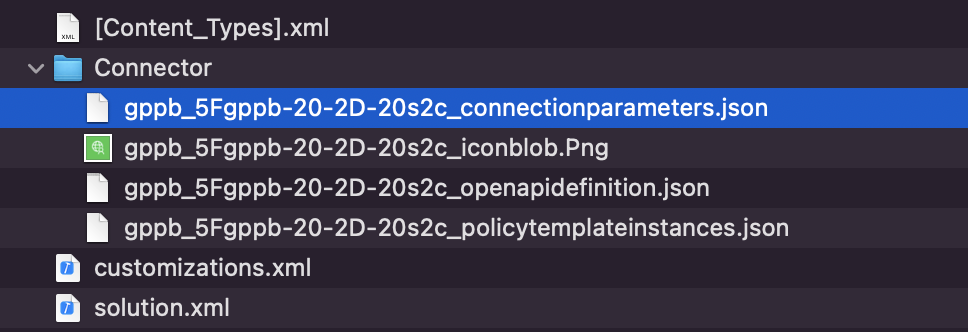
}

]  


1. Now, you can create a custom connector before testing.
2. Create a new connection to the external API. It will ask you the API key to connect to API. Enter the following API key: uZRlZNQafWOXIJYCprrdCvlWOaUWQWVvsv58M2Dl3rVGW0cTui3caA==  
   After that, you need to refresh the connection to see a newly created connection.  
   
3. Enter a valid conversation id from Dataverse to test operation.
4. Bravo! You have now successfully created a custom connector.

## A custom connector: Extended steps

The section focuses on creating a reusable custom connector which can connect to different endpoint.

1. Export the custom connector solution as an un-managed solution.
2. Un-zip the solution and open the connectionparameters.json  
   
3. Add the following text before api\_key.  
   This will be used in a custom policy to over-write the API url.

"api\_url": {

"type": "string",

"uiDefinition": {

"displayName": "API Url",

"description": "The API Url for this api",

"tooltip": "Provide your API Url",

"constraints": {

"tabIndex": 1,

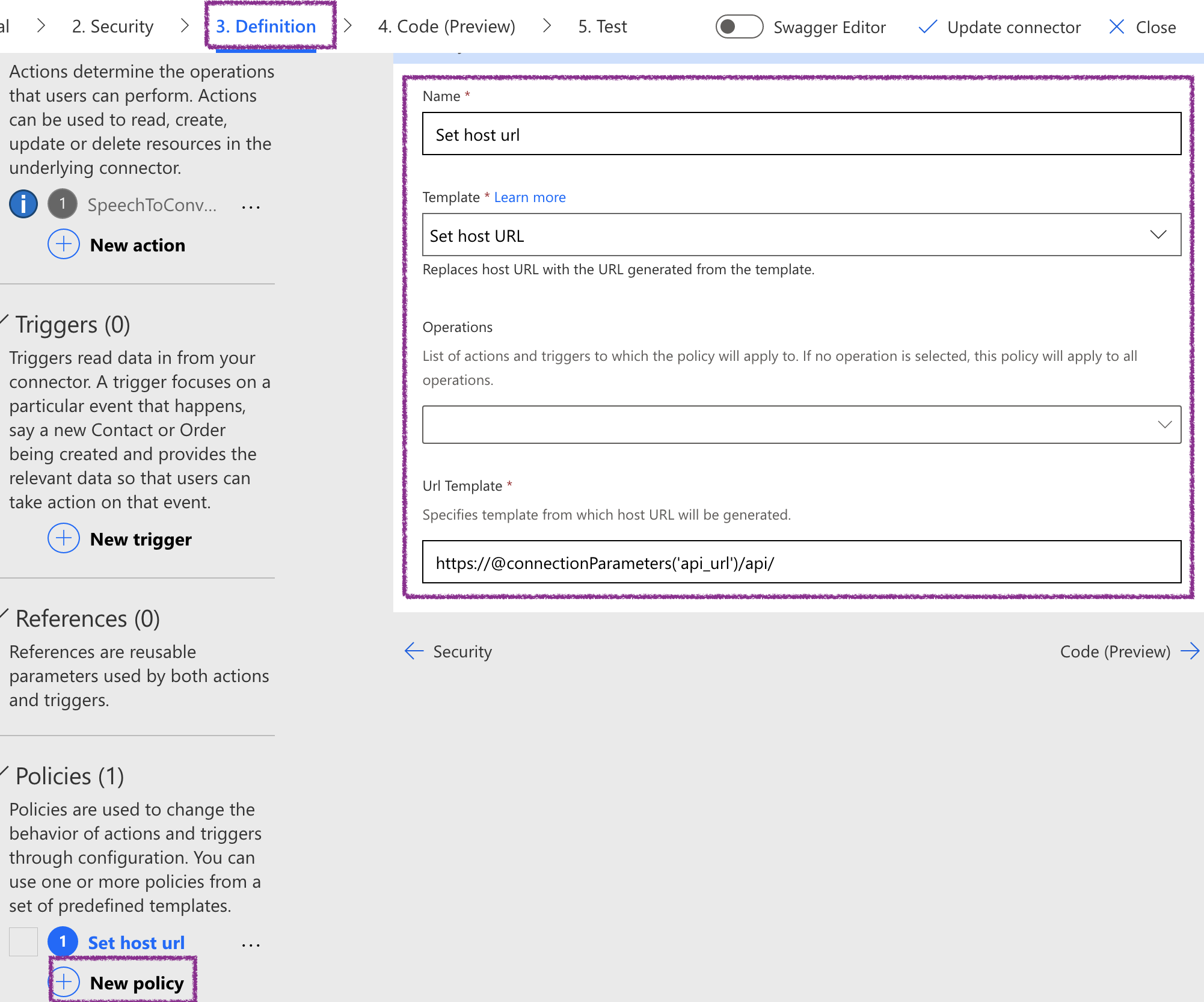
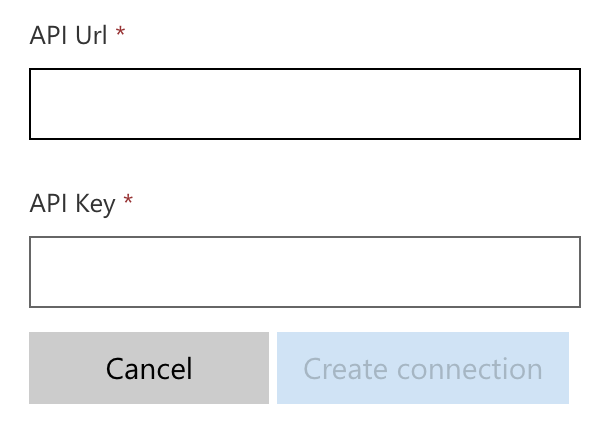
"clearText": true,

"required": "true"

}

}

},

1. Zip all files and import it to Dataverse.
2. Edit the custom connector. Add a new policy under Definition.  
   
3. Save and test the custom connector again. You will need to create a new connection.
4. You will see the connection is now asking the API URL.  
   Enter gppbfunction.azurewebsites.net in API Url, and uZRlZNQafWOXIJYCprrdCvlWOaUWQWVvsv58M2Dl3rVGW0cTui3caA== in API key.  
   
5. This method is useful if you have a same API in different environment. You don’t need to create multiple custom connectors. Instead, values can be parameterised.

# Session 4 - Power Automate

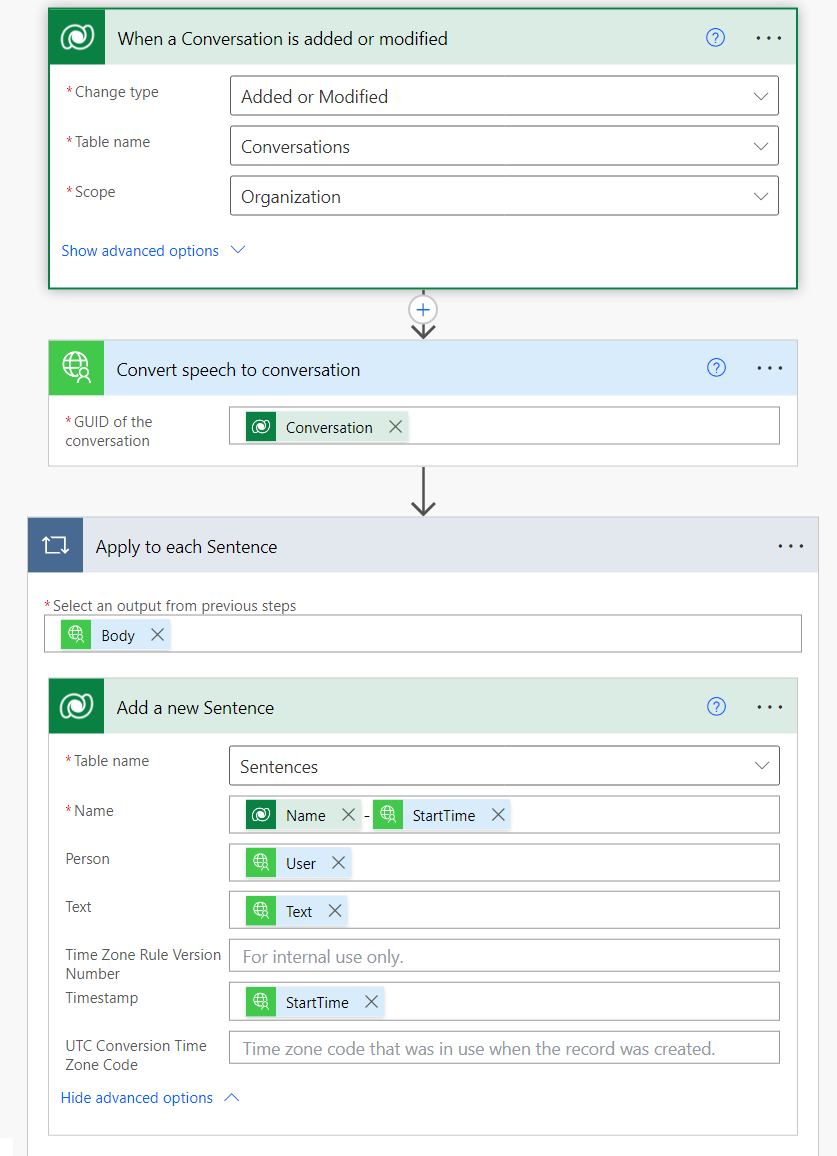
**Linn Zaw Win**

On create of conversation, pass the recording to Custom Connector and create sentences as per the returned JSON (no sentiment)

On create of sentences trigger the AI model and get the sentiment.

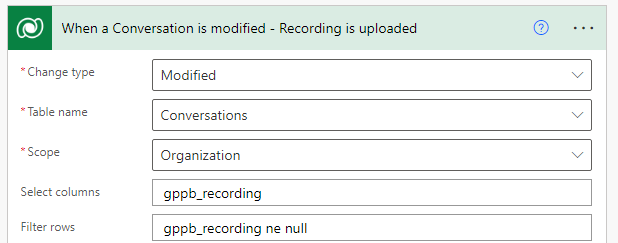
## Integrate with Power Automate

### Summary of a cloud flow using Speech to Conversation custom connector

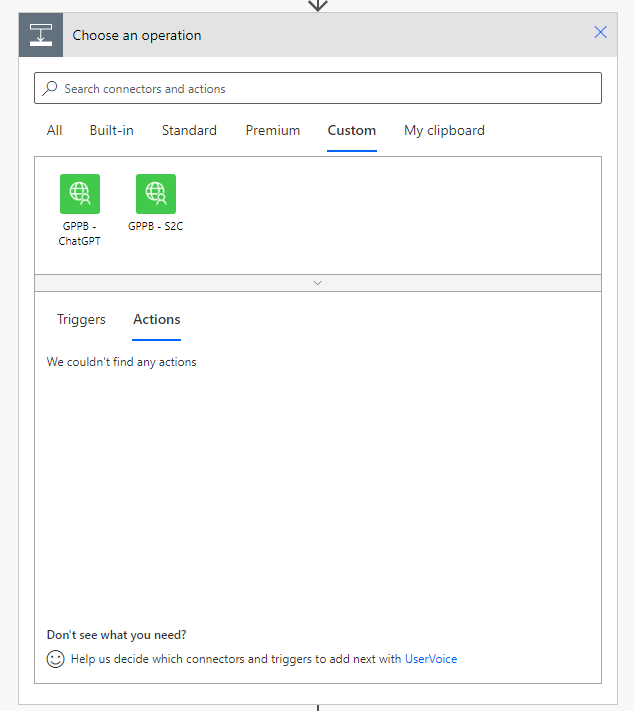


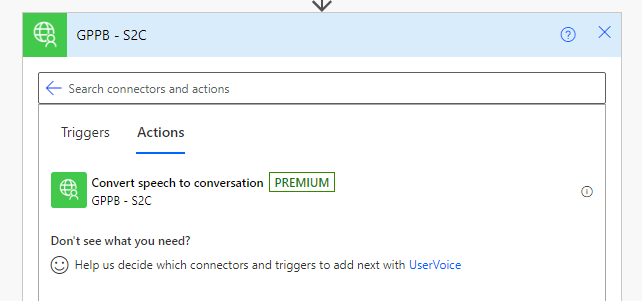
## Building flow: A step by step manual process

1. Navigate to <https://make.powerapps.com/> and open Solution.
2. Click on **New > Automation > Cloud flow > Automated**
3. Set the Flow name and choose the Dataverse trigger “When a row is added, modified or deleted”
4. Start by setting the Trigger details as follows:
5. Change type: Modified
6. Table name: Conversations
7. Scope: Organization
8. Select columns: gppb\_recording

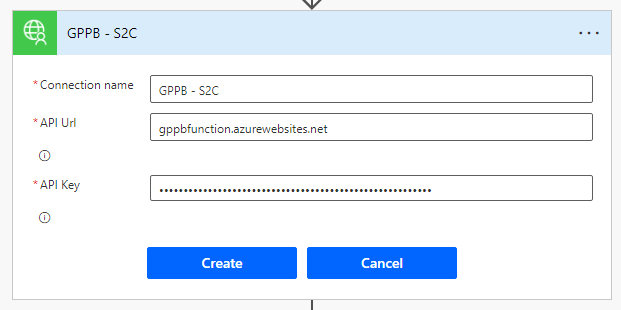


1. Insert a new step and select the **GPPB - S2C** custom connector in the Custom tab and choose **Convert speech to conversation** action.

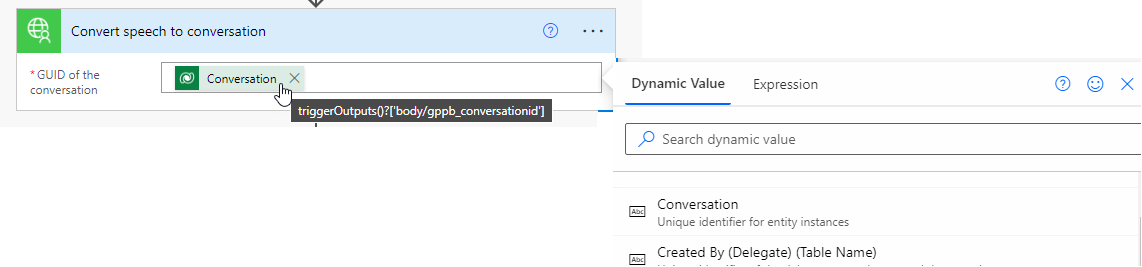




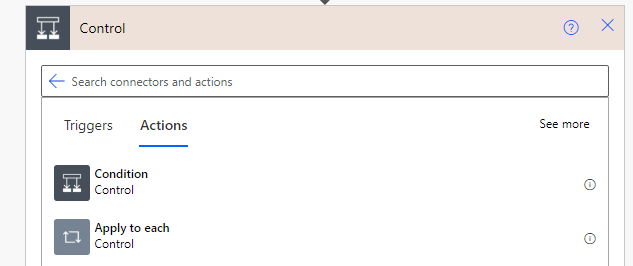
1. If you are prompted to create a new connection, set the Connection details as follows:
2. Connection name: «Any name for the connection»   
   API Url: gppbfunction.azurewebsites.net   
   API Key: uZRlZNQafWOXIJYCprrdCvlWOaUWQWVvsv58M2Dl3rVGW0cTui3caA==

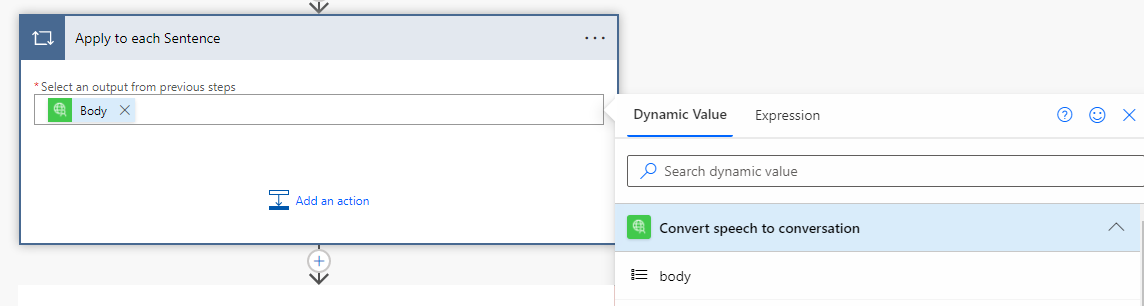


1. Once the connection is created, populate the GUID of the conversation with the unique identifier column of the Conversation table.

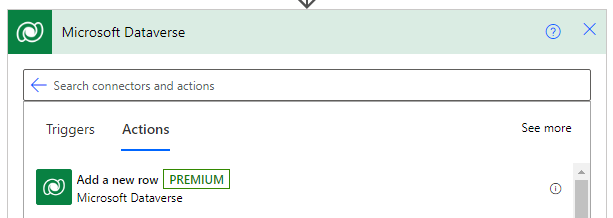


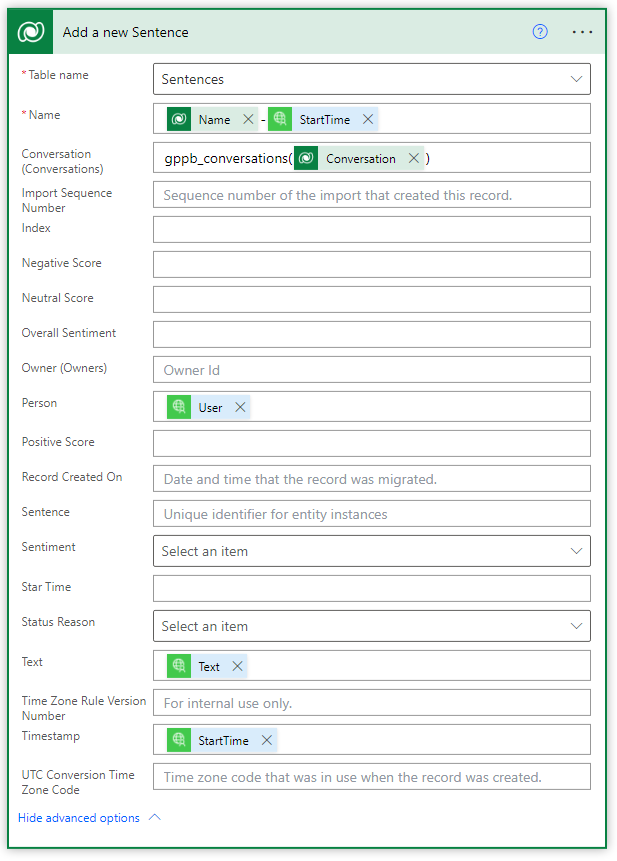
1. Add an **Apply to each** step and populate the array input with the body output from the **Convert speech to conversation** action.



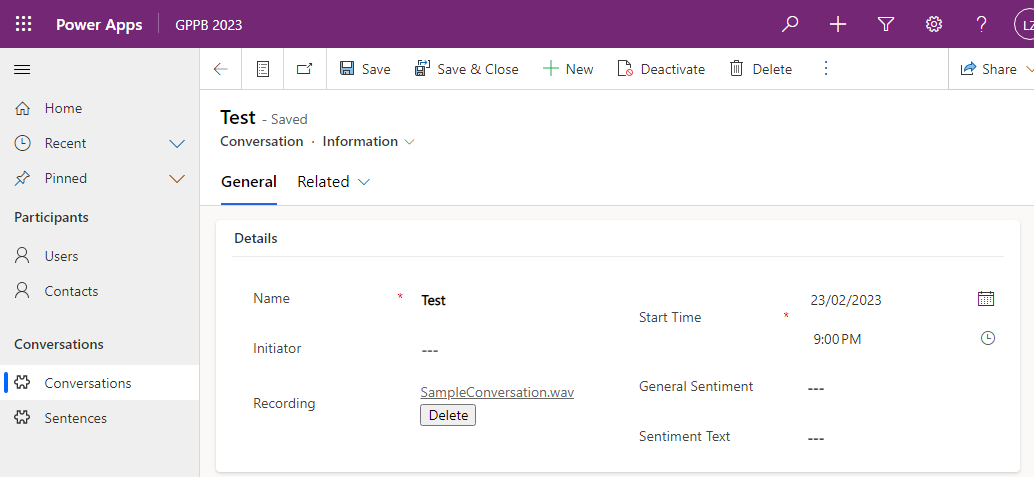


1. Inside the **Apply to each** loop, add an action and select **Microsoft Dataverse** connector > **Add a new row** action. Then, populate the details of the action as follows:
2. Table name: Sentences
3. Name: «Name column from Conversation»-«StartTime output from S2C action»
4. Conversation (Conversations): gppb\_conversations(«Unique Identifier of Conversation»)
5. Person: User output from S2C action
6. Text: Text output from S2C action
7. Timestamp: StartTime output from S2C action

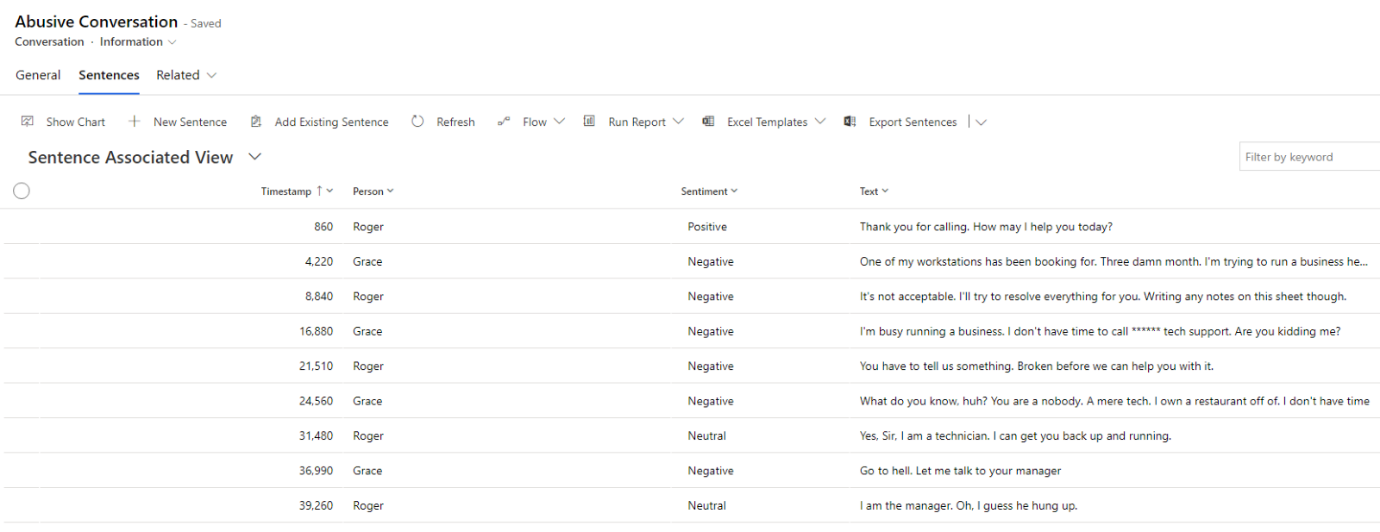




1. Test the flow by creating a Conversation record in GPPB 2023 model-driven app, save and upload the Recording file.



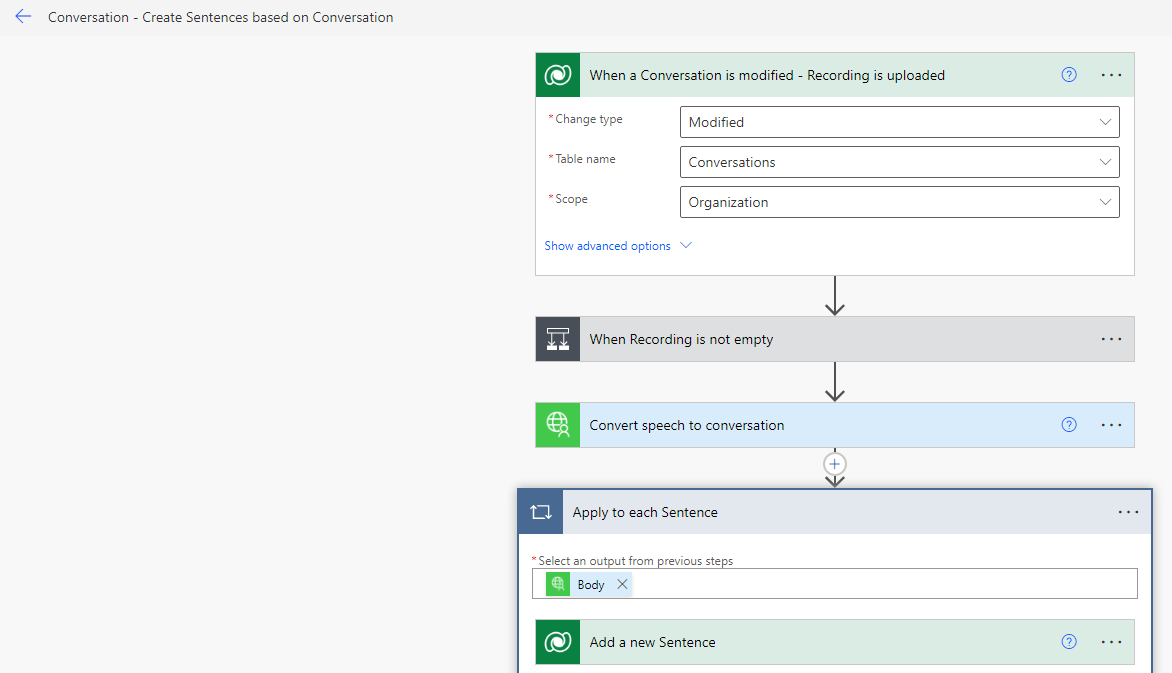
1. Once the flow is run successfully, verify the outcome by navigating to the Sentences associated view under the Related tab of the Conversation form.



## Power Automate Best Practices: Extended goals

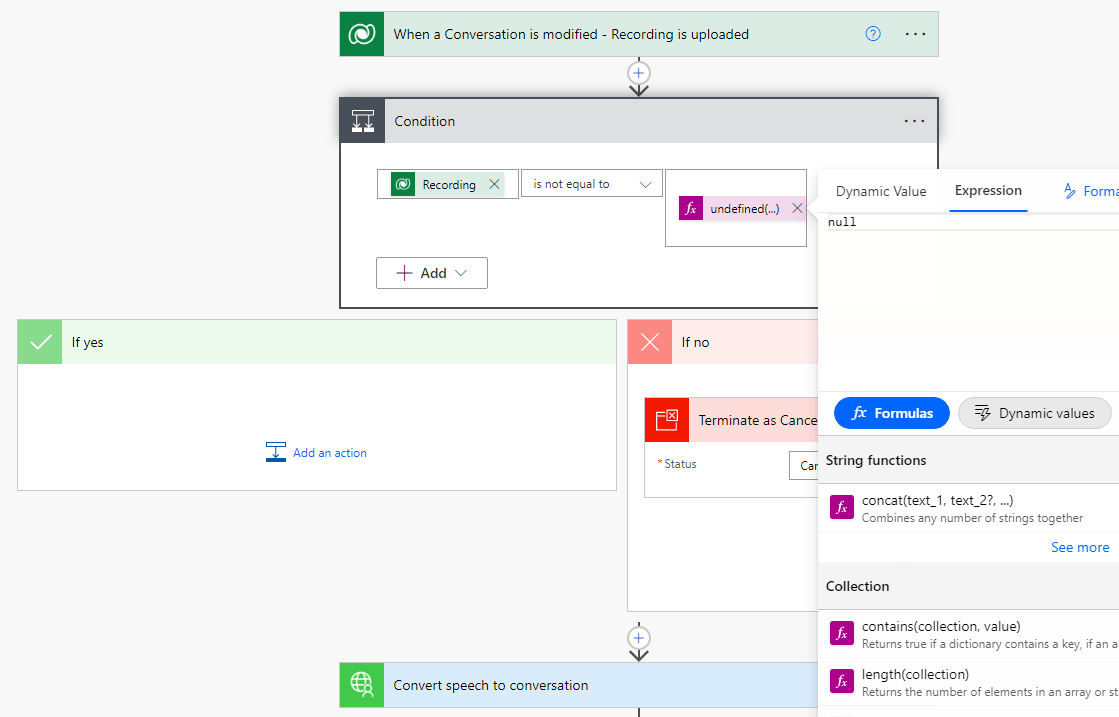
### Name the flow and steps properly

Set the name of the flow, trigger and action steps to easily identify and understand the purpose of the flows/actions.



### Handling empty recording

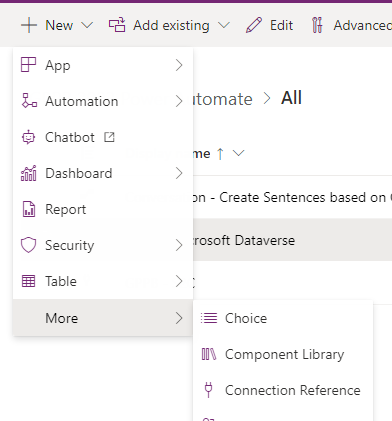
Handle the flow failure if the Recording is deleted



### Creating Connection Reference Manually

Create Connection Reference in the solution to set the logical name properly.

More details: <https://github.com/MattCollins-Jones/PowerAutomateArtemisFramework/wiki/Cloud-Flow-Coding-Standards>





**Creating Service Principal Connection for Microsoft Dataverse**

Create Service Principal Connection for Microsoft Dataverse connector to run the Microsoft Dataverse actions in the flow as an application user instead of the flow author user.

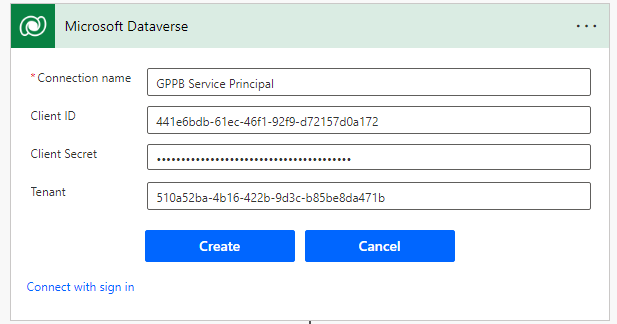
Step by step guide: <https://benediktbergmann.eu/2022/01/04/setup-a-service-principal-in-power-automate>

Details for this bootcamp if the App Registration cannot be created.

Client ID: 441e6bdb-61ec-46f1-92f9-d72157d0a172

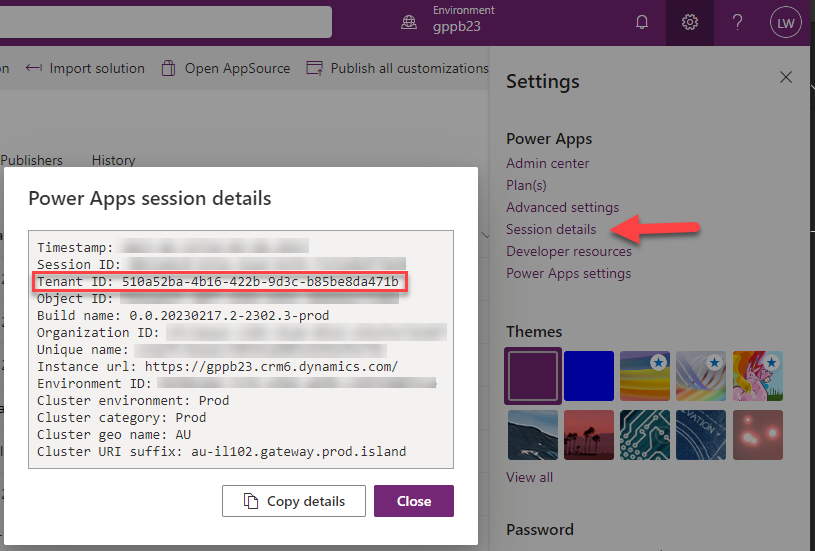
Client Secret: dfG8Q~y0NVvaNqOBPgKpvqgnl1m202hz3t5TZad7

Tenant: 510a52ba-4b16-422b-9d3c-b85be8da471b



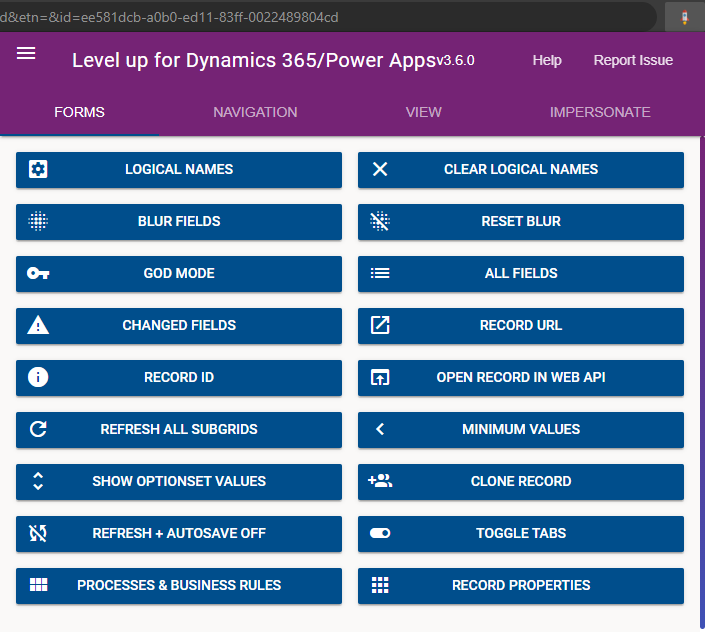
**Alternative Way of Getting Tenant ID**

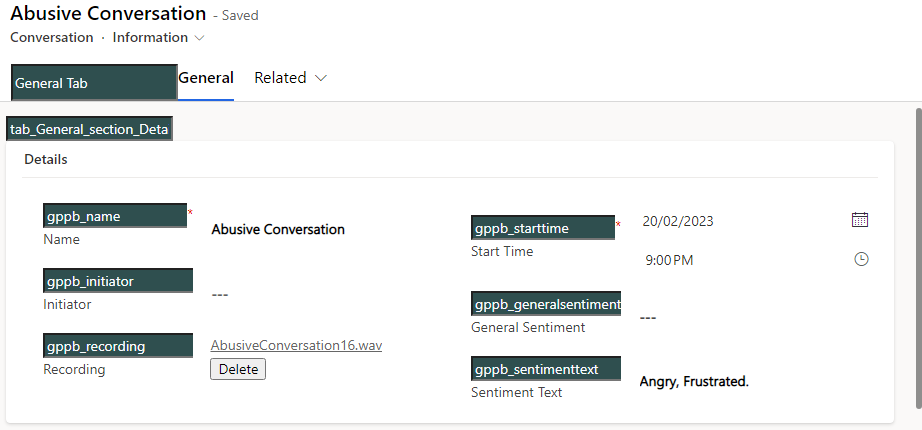
If we only have Client ID and Secret, we can get Tenant ID from make.powerapps.com without access Azure Portal.



**Level up for Dynamics 365/Power Apps Browser Extension**

Install Level up for Dynamics 365/Power Apps extension from [Chrome Web Store](https://chrome.google.com/webstore/detail/level-up-for-dynamics-365/bjnkkhimoaclnddigpphpgkfgeggokam)/[Edge Add-Ons](https://microsoftedge.microsoft.com/addons/detail/level-up-for-dynamics-365/mdjlgdkgmhlmcikdmeehcecolehipicf) to easily identify the EntitySetName, column logical name on the form, etc.





# Session 5 – Building a Power App (canvas)

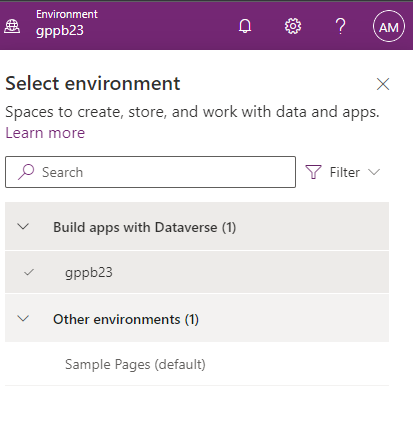
**Abhay Mishra**

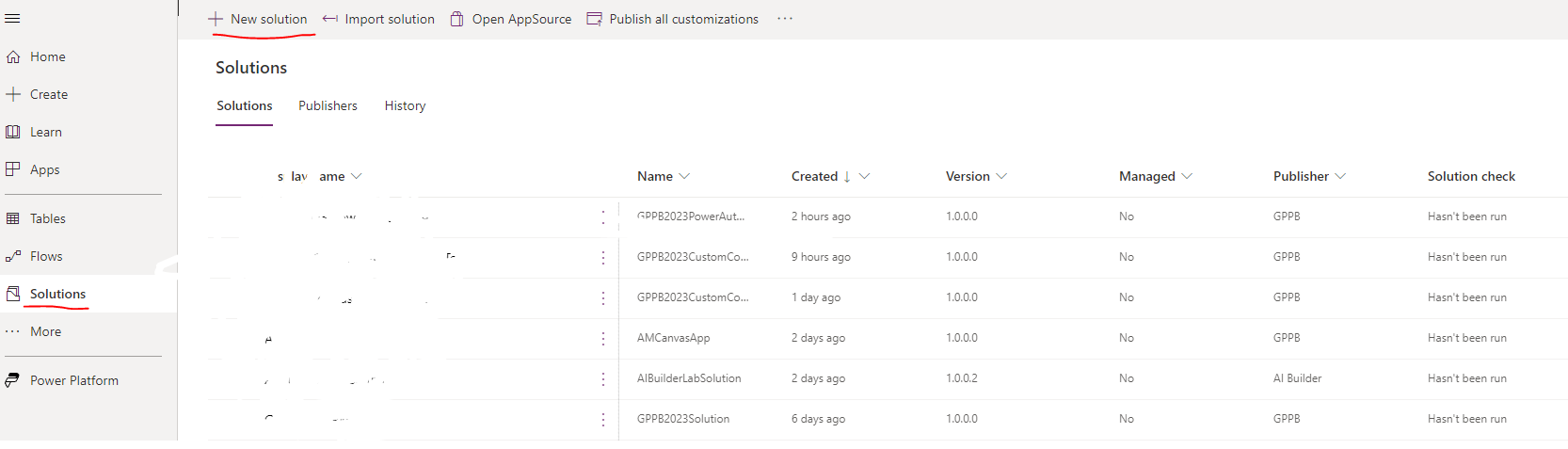
## Goals

1. Create a new solution (You can use your existing solution if you have already created one in one of the previous sessions today)
2. Create a data structure (You can use the existing schema and the following tables: **1. Conversation (gppb\_conversation)** 2. **Sentence (gppb\_sentence)**
3. Create a canvas app containing one screen, and two galleries that consist of a person’s image with sentences in **Red** or **Green** according to the sentiment of the conversation.
4. Display image of the person next to the conversation.

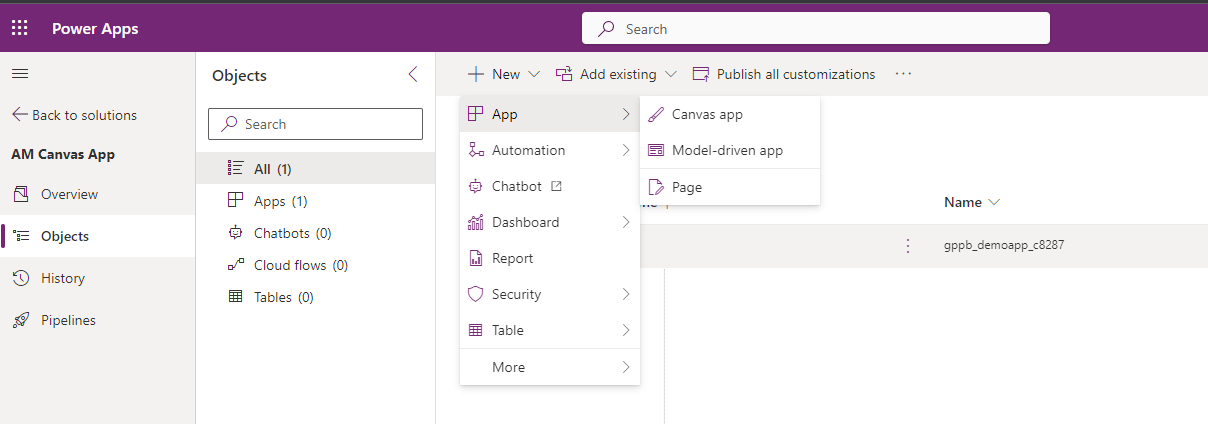
## Steps to create your Canvas-App

1. Log in to <https://make.powerapps.com/> using the credentials supplied to you this morning and click Solutions and either open an existing solution or click the new solution button at the top.
2. Make sure that you are working in the correct environment – gppb23

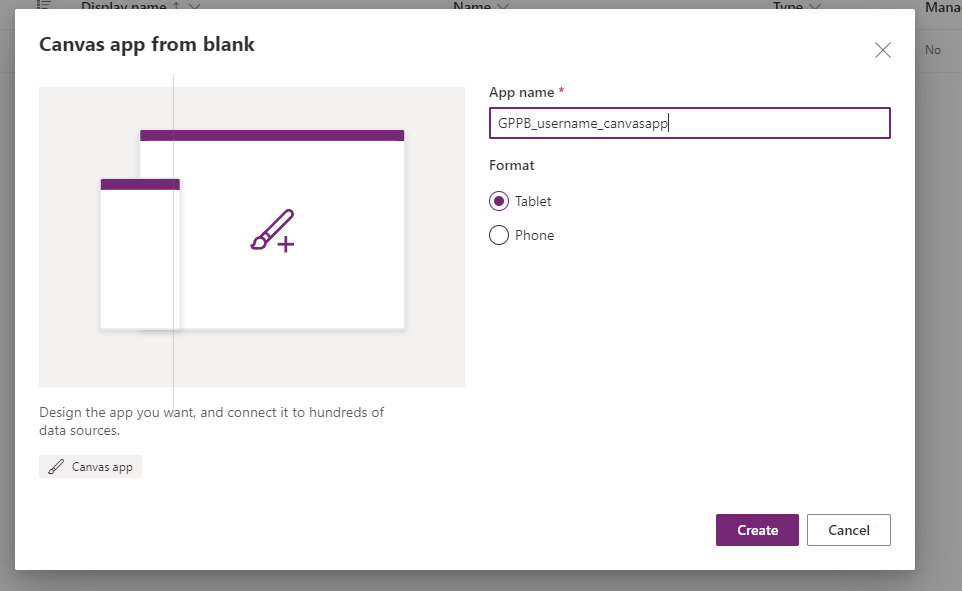




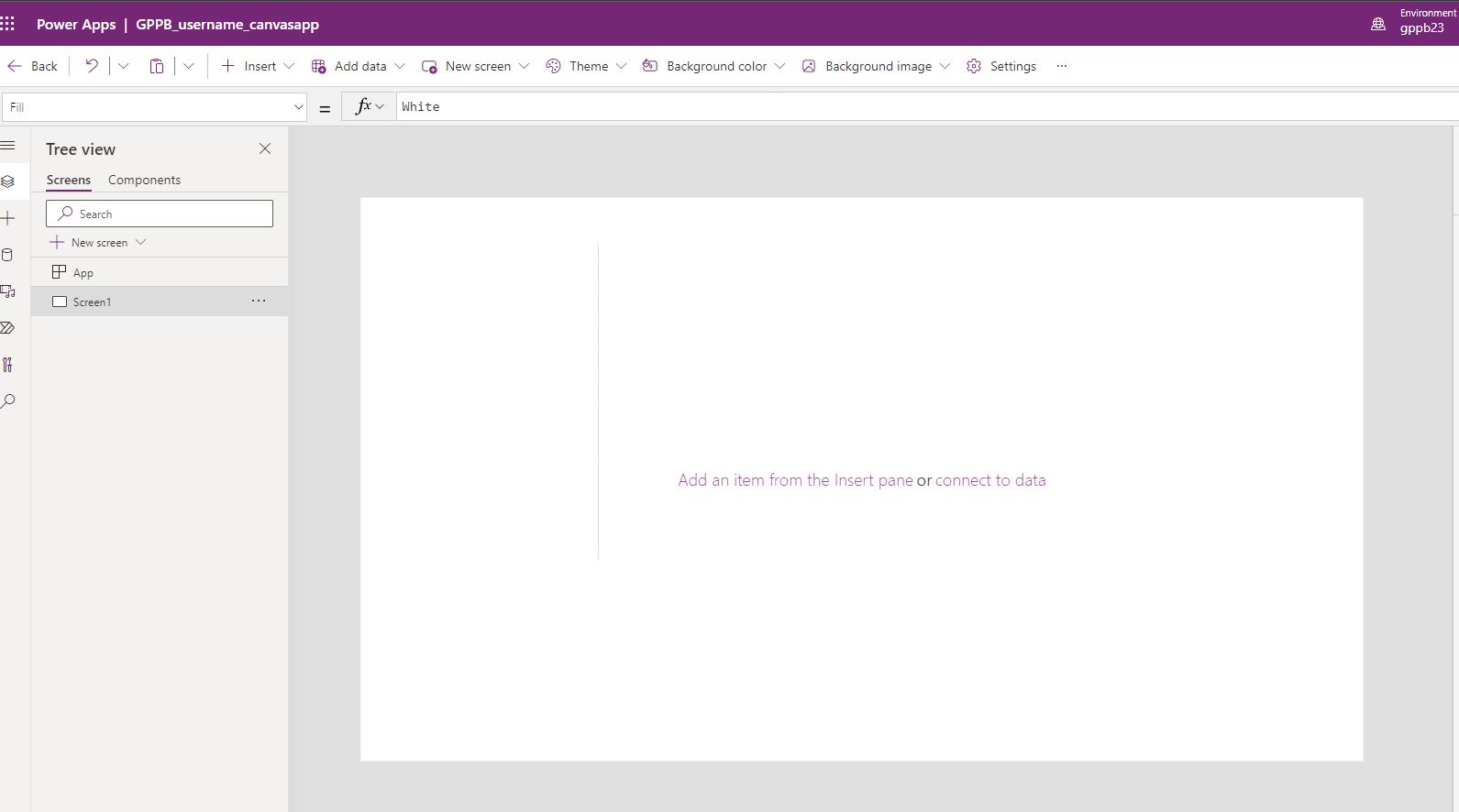
1. Click New -> App – Canvas App from the top as shown below



1. Give your app a name and Select **Create** to create the blank canvas app.

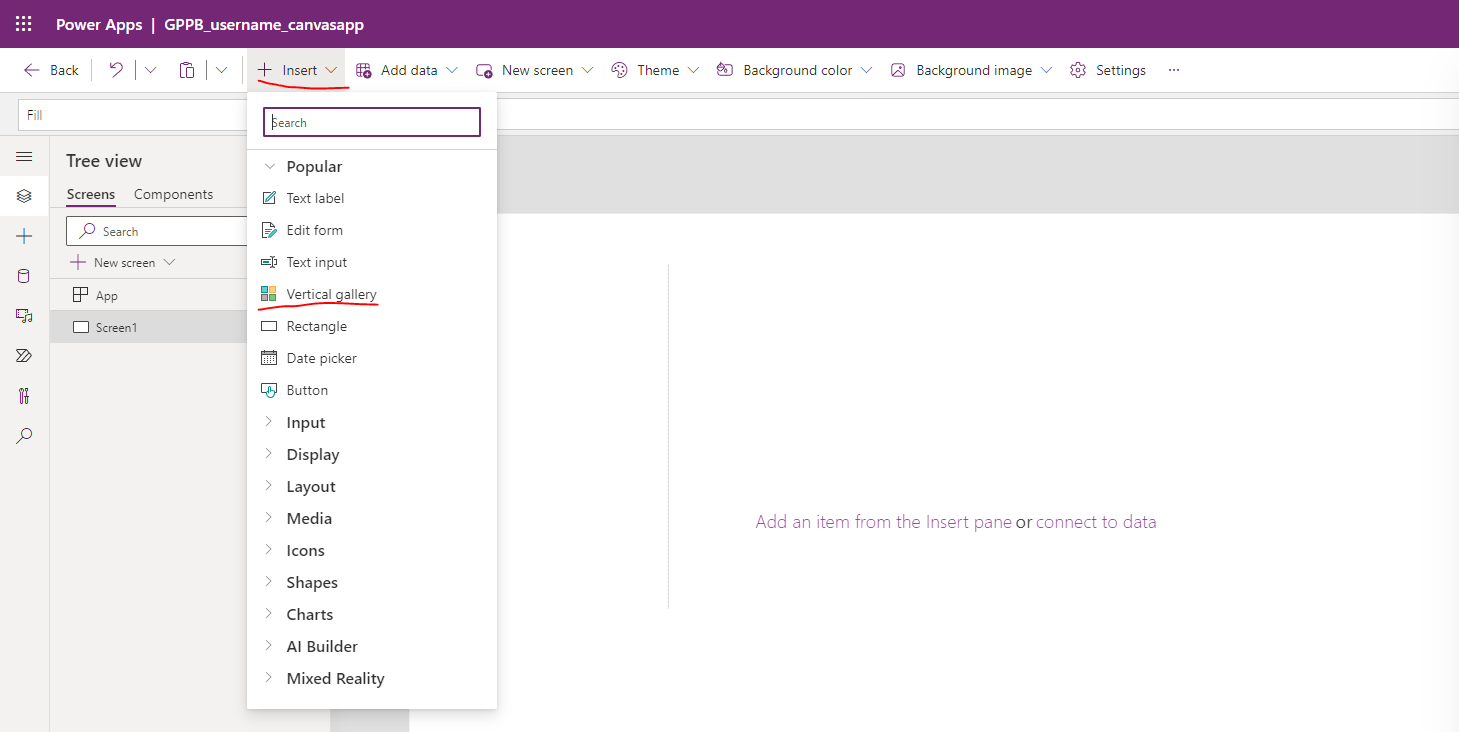


1. Once created, the app opens in Power Apps Studio for you to start configuring the app functionality.

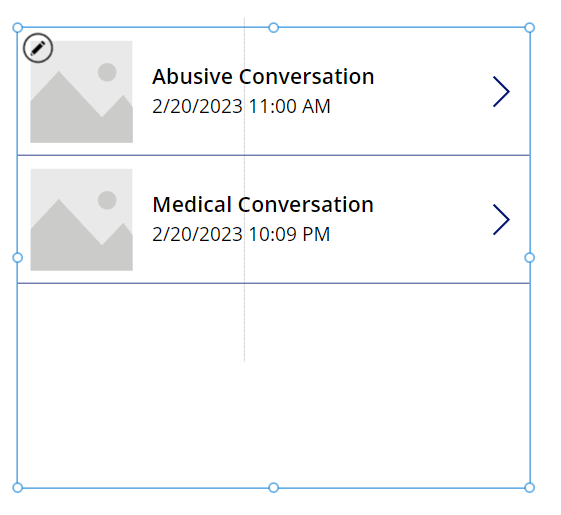


1. You can change the background colour of the app from the menu.

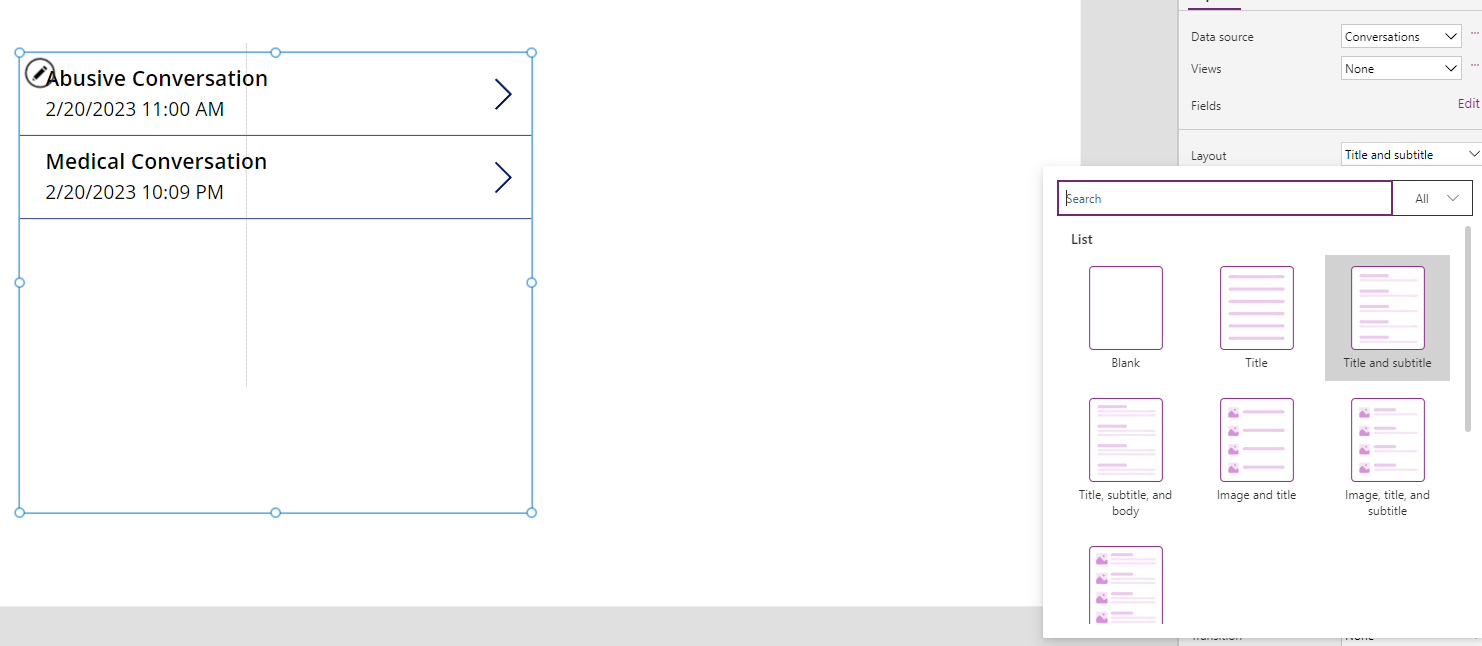
Also from the menu, click Insert and select vertical gallery as shown below



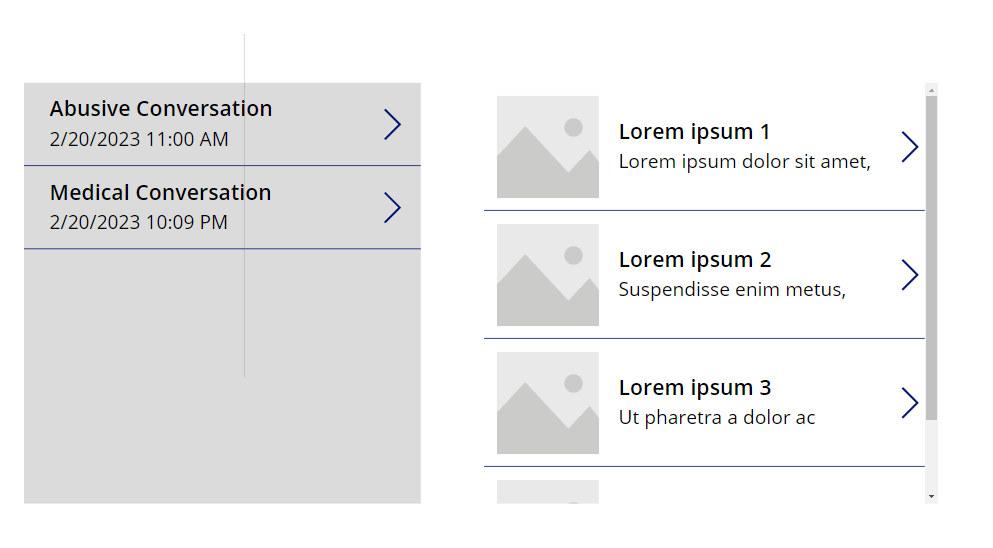
1. Select a data source for the gallery, we want to display the conversations in this galley. To do so, please search for the table name “ **Conversations**”

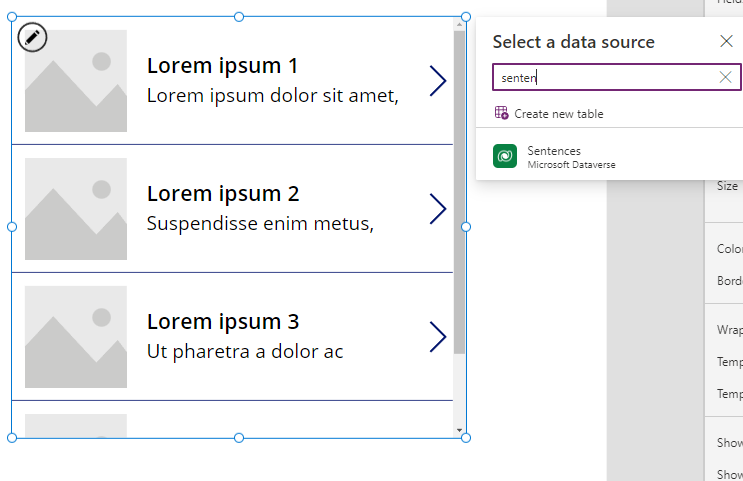


1. Select the galley as shown below and change the Layout to the Title and Subtitle and select a background colour for the gallery.

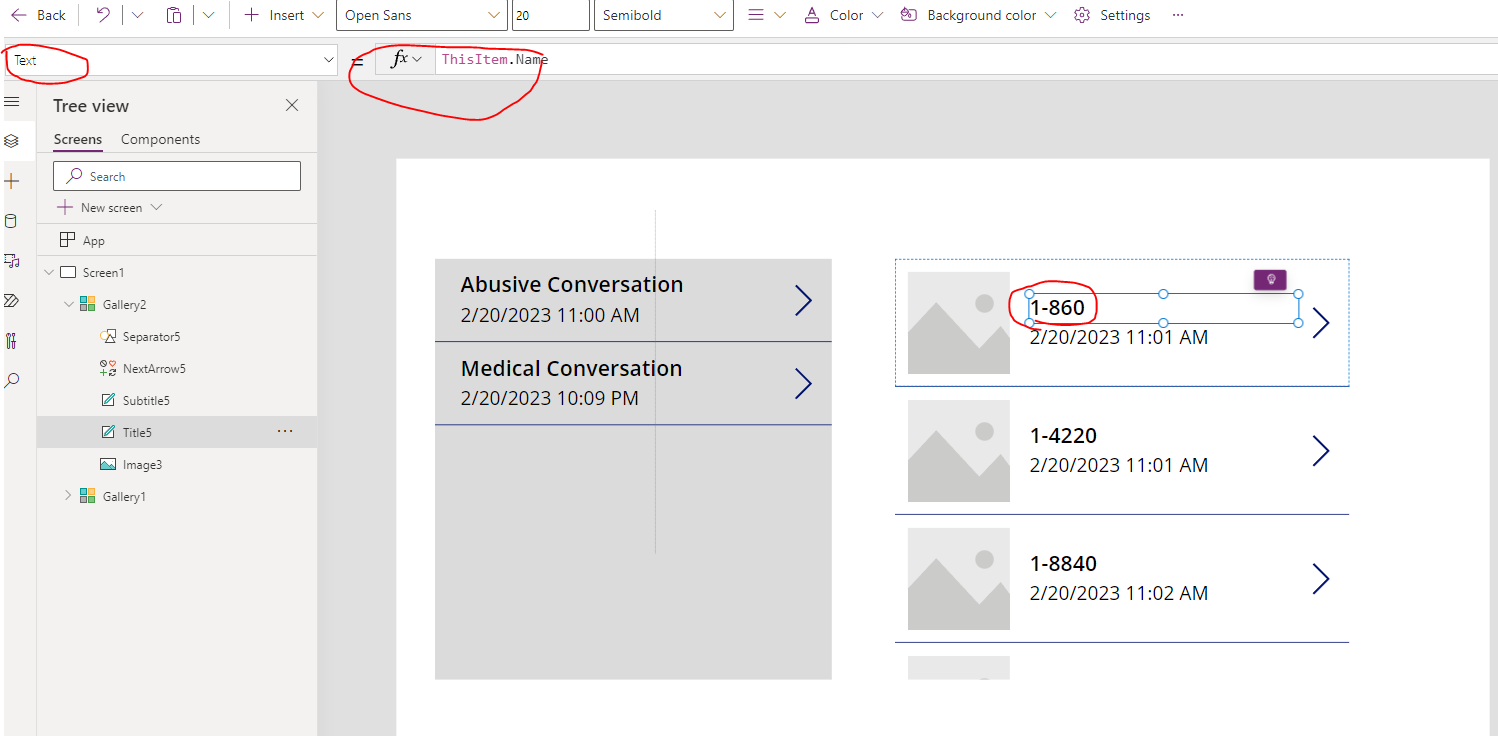


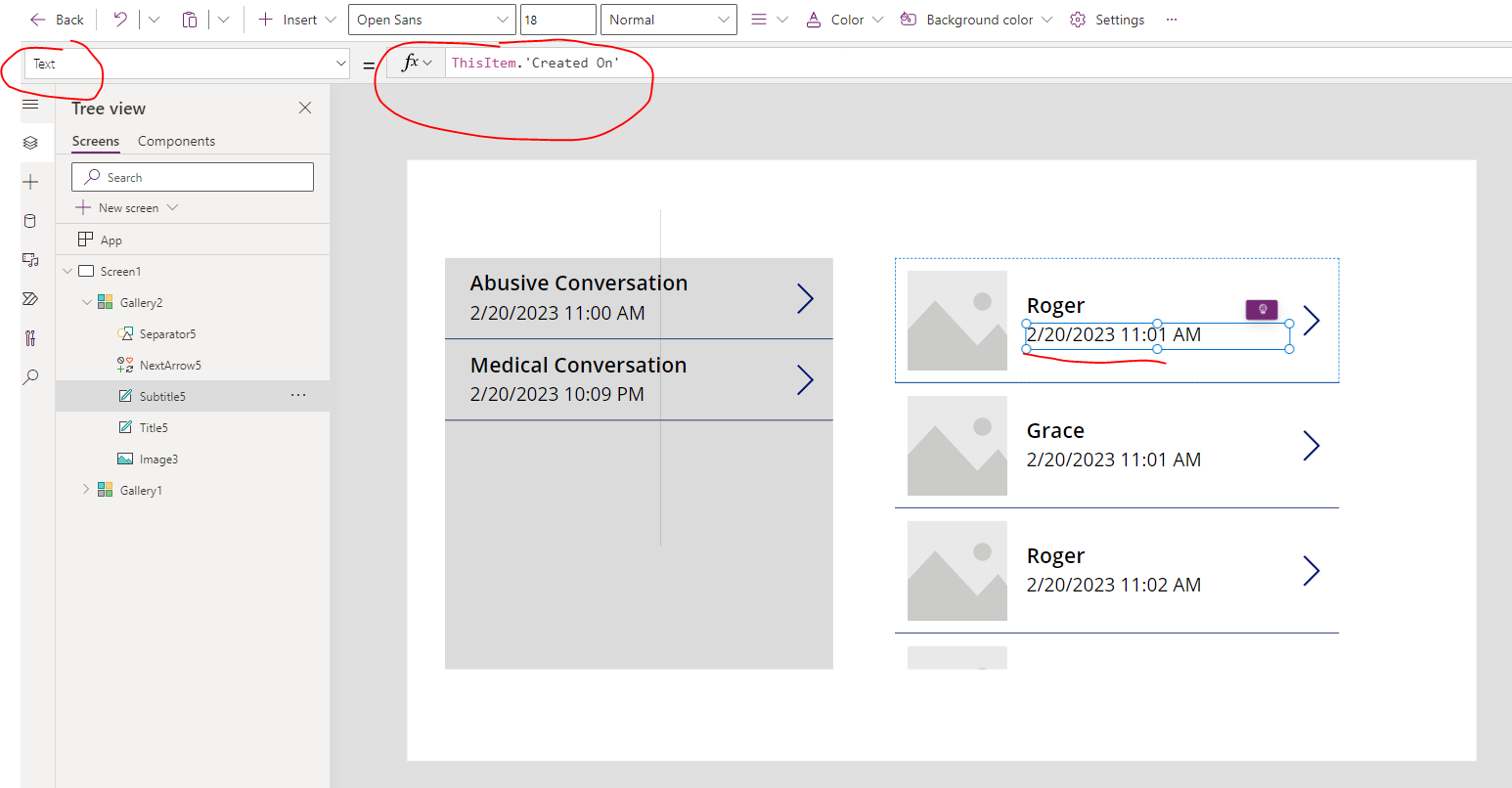
1. Add one more vertical gallery from the menu as shown below. The data source for the second gallery should be a table called **Sentences**.

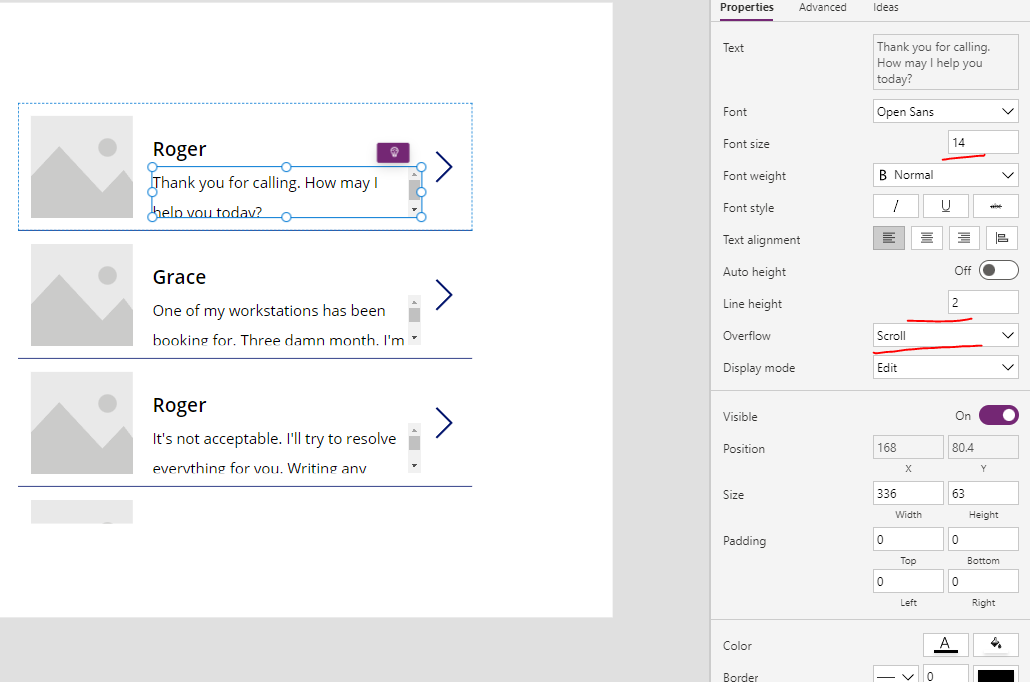




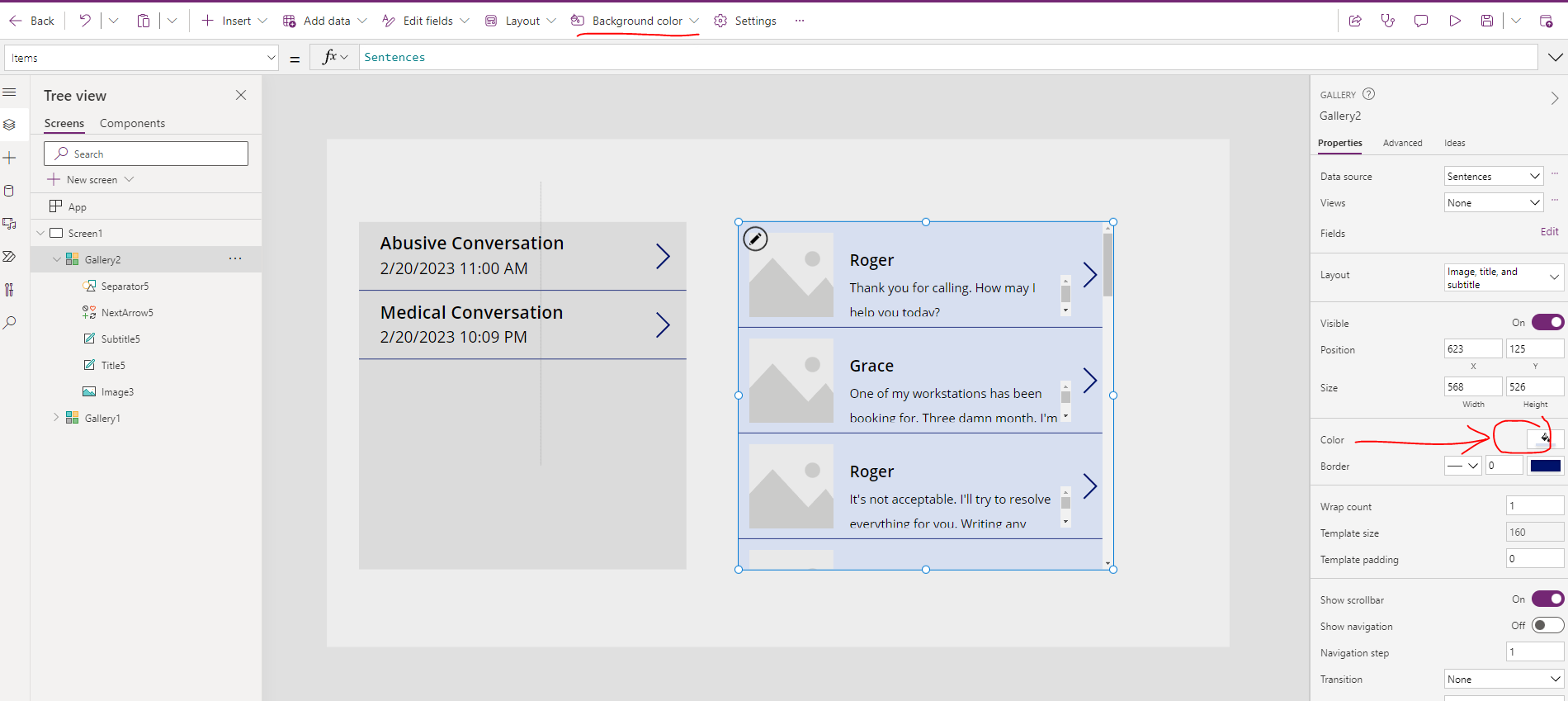
1. Select the first Subtitle and change the formula of **Text** to “ **ThisItem.Person**” as shown below



1. Select the second subtitle ad change the formula of **Text** to “**ThisItem.Text**” as shown below 
2. While selecting the second subtitle modify the formatting options from the properties tab as shown below

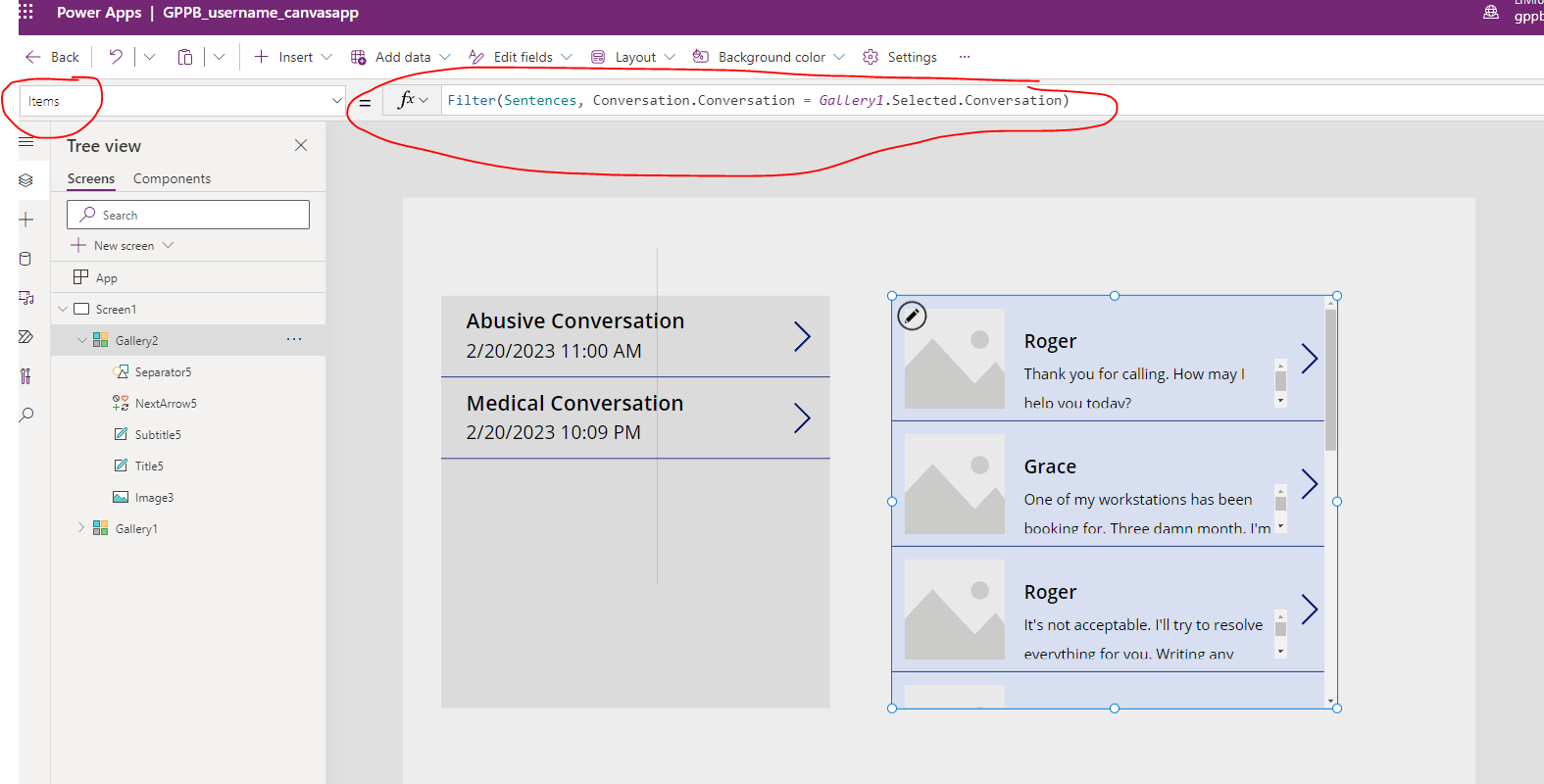


1. Modify the background color of the screen and for the second gallery as shown below

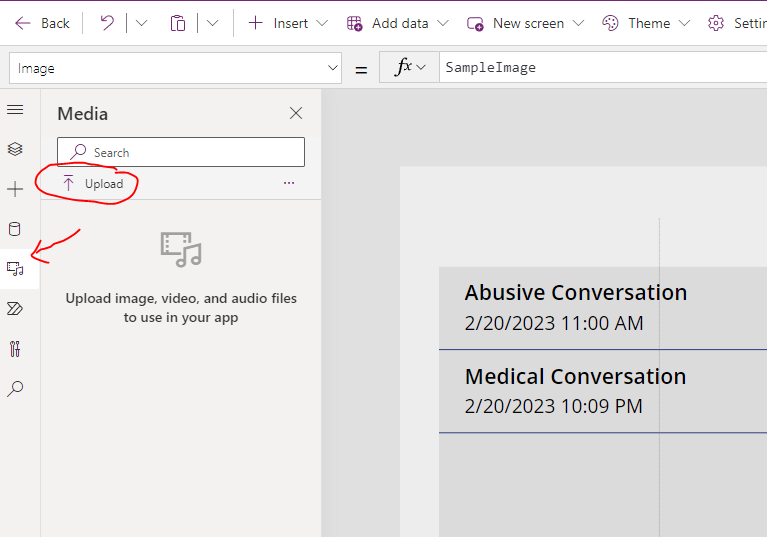


1. Now select the second Gallery as shown below and modify the formula for Item as following to filter the sentences relevant for the selected conversation

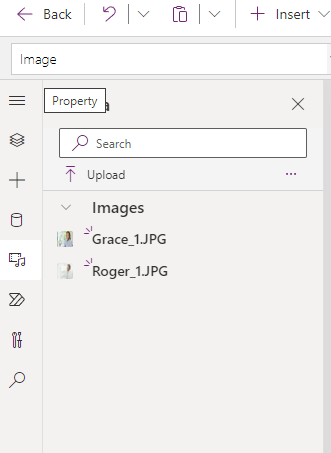
Formula: **Filter(Sentences, Conversation.Conversation = Gallery1.Selected.Conversation)**



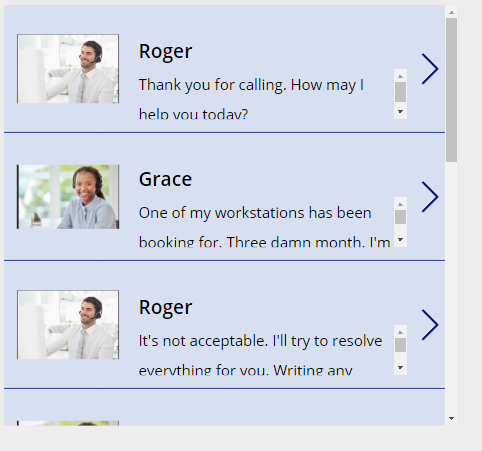
1. We will now add an image for the persons in this conversation, Roger and Grace. To do this, download any random pictures and give the image file a preferred name. You will need to add these pictures to the canvas app and will have to use the exact name in the formula.
2. To add pictures to the canvas app library, click Media from the side menu and click upload to select the files. In this example, I will be using files with names, **Roger\_1** and **Grace\_1** and will be using those names in my formula.



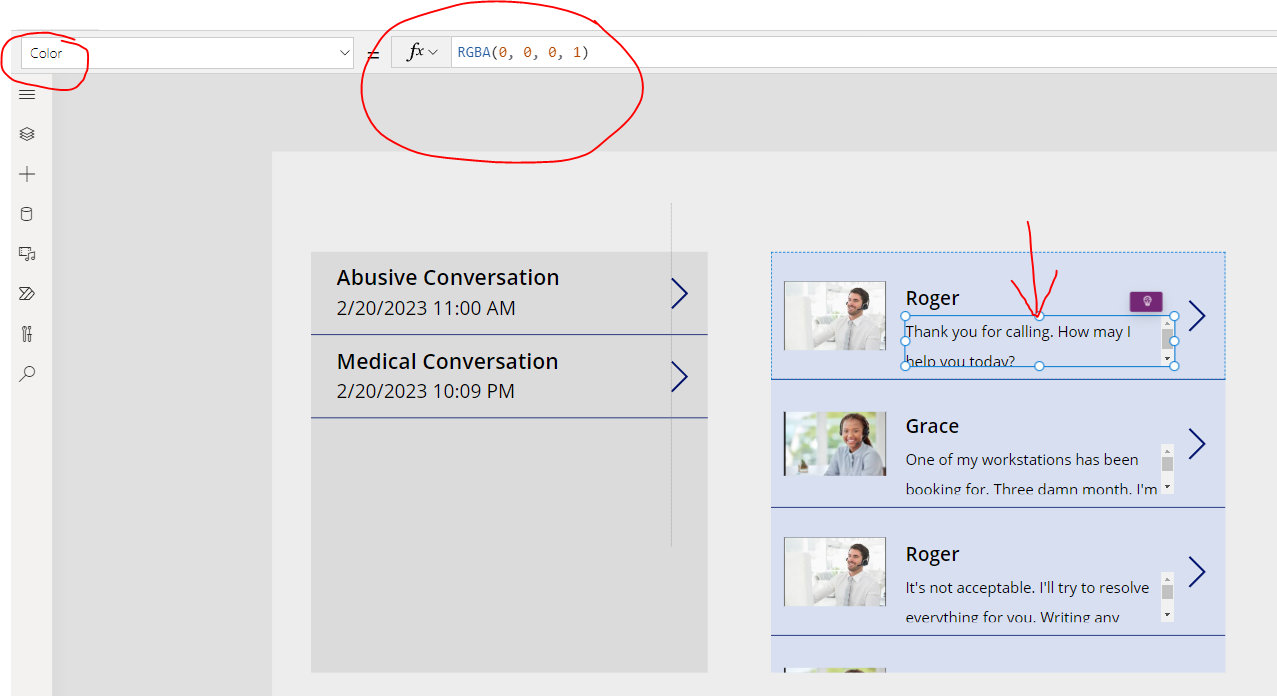
1. The pictures should appear in the media library as shown below



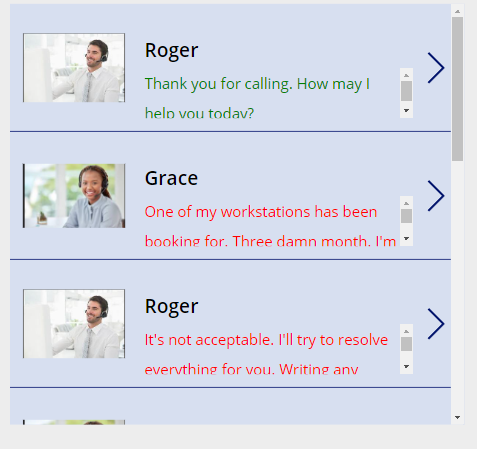
1. Now select the image component as highlighted below and change the formula for the image as following: **If(ThisItem.Person = "Roger",Roger\_1,Grace\_1)**
2. Your gallery should now look like this



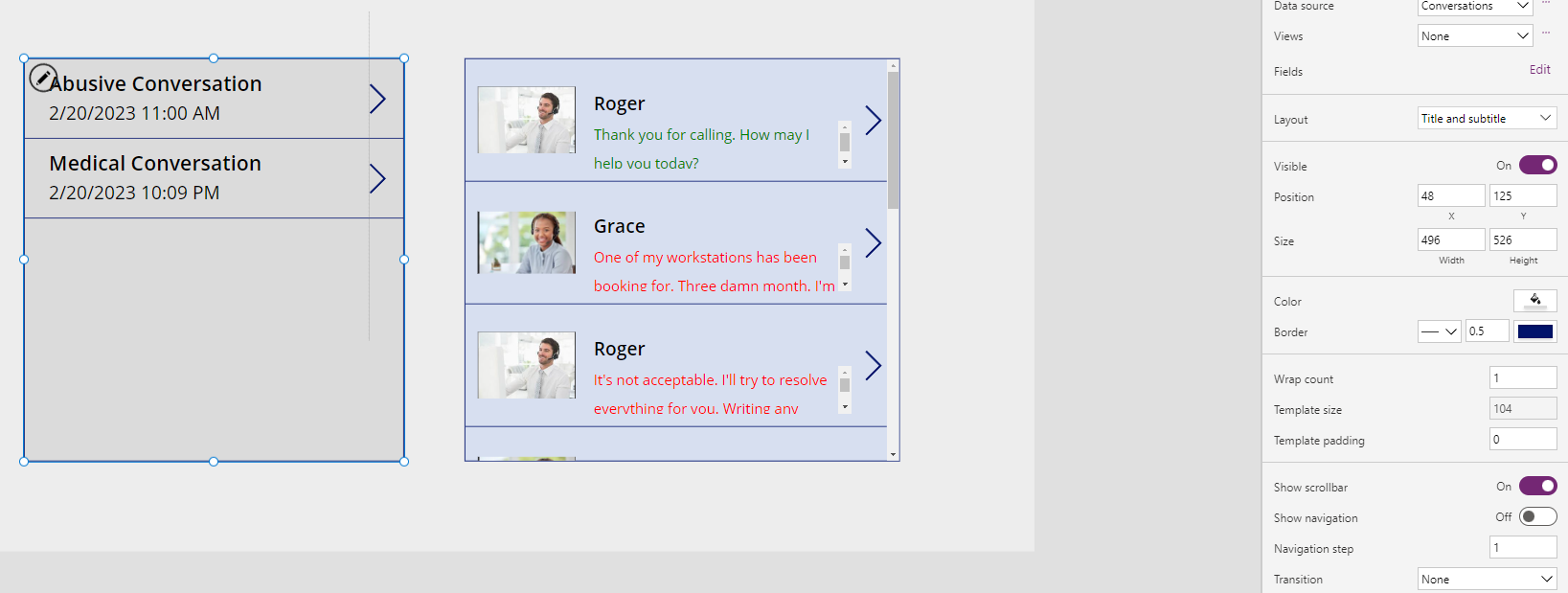
1. Now let’s add a formula to change the colour of the text according to the sentiment of the sentence, colour Red for negative and Green for Positive Sentiment.
2. Select the text component as shown below and update the formula for **Color** to **If(Text(ThisItem.Sentiment) = "Positive", Color!Green,If(Text(ThisItem.Sentiment) = "Negative", Color!Red,Color!Black))**



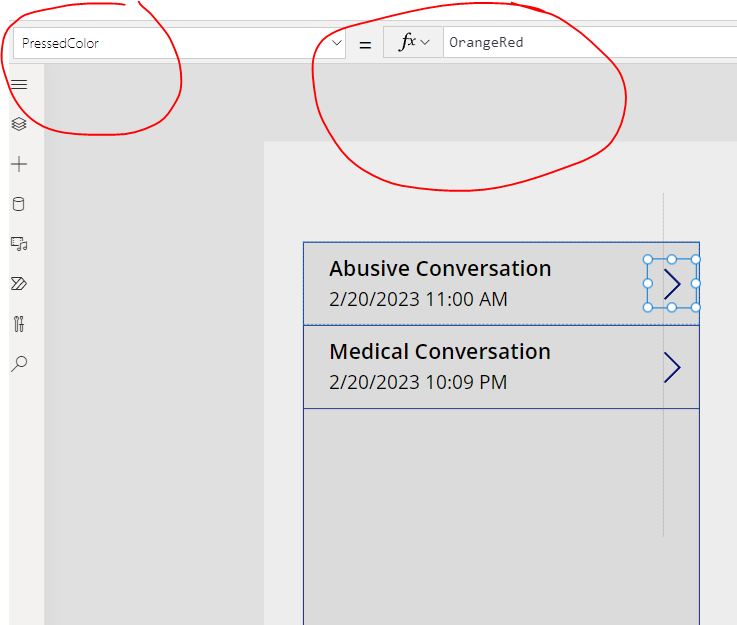
1. Your second gallery should look like this now



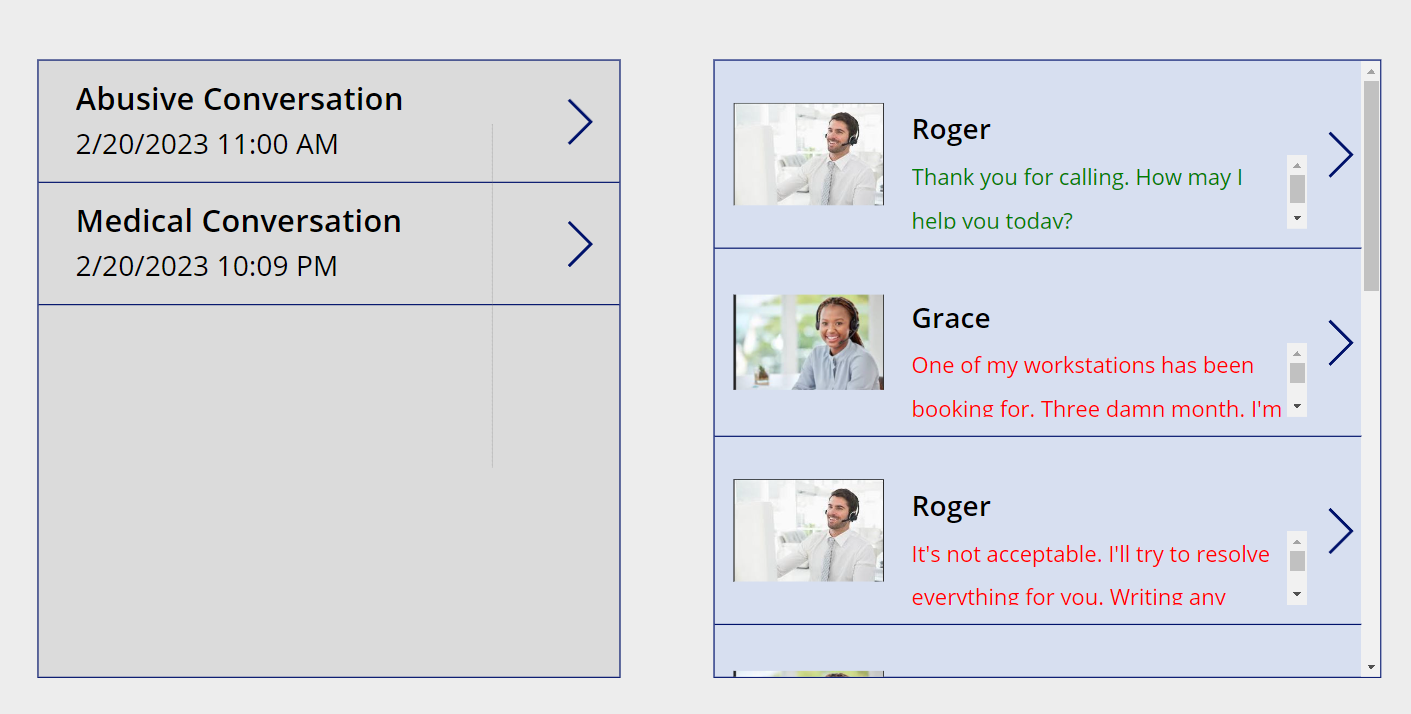
1. Add an appropriate border for both galleries from the properties tab on the right side



1. You can also modify the Transition style for galleries by selecting an appropriate value.
2. Feel free to explore the PressedFill and and on select properties of the button



1. When done formatting the app including the background and galleries, make sure to save the app from the top right and your app is now ready to be used. Click the run button from the top right….Congratulations you have created your first Canvas App.



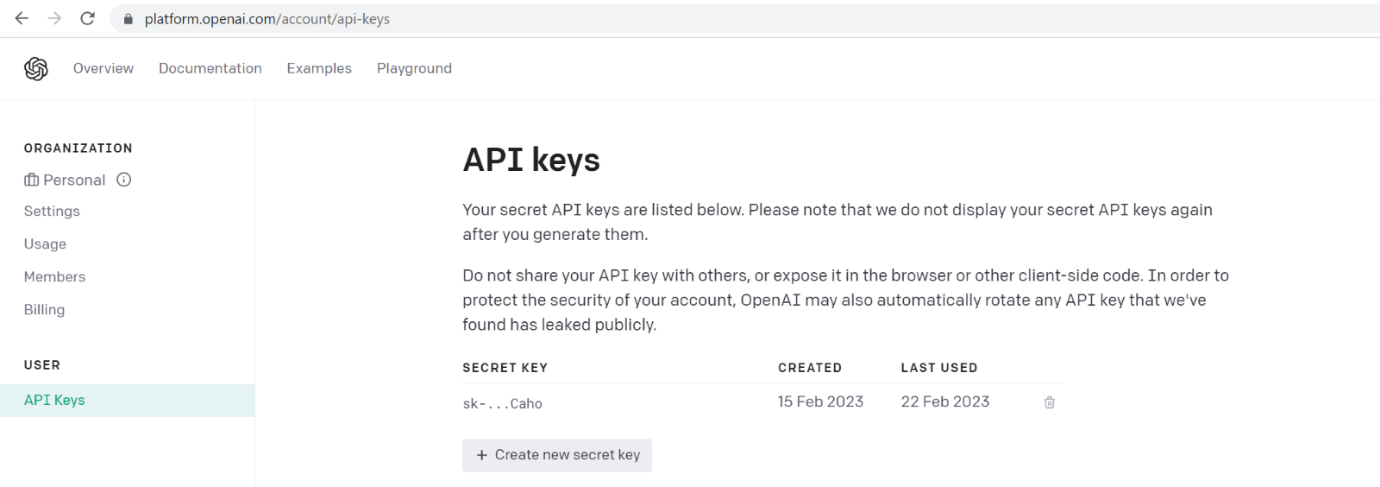
# Session 6 – Connect with ChatGPT

**Shobhit Bhatnagar**

Connect to Power Automate (desktop) to chatGPT and get the sentiment for the overall summary.

## **Pre-requisites:**

1. Create an account on ChatGPT: <https://chat.openai.com/>
2. Use your personal or work email to create an account.
3. Generate an API Key for connecting to ChatGPT via API
4. Navigate to <https://platform.openai.com/account/api-keys>
5. Create an API Key and copy and store it on your local machine in One Note.



For our session we will invoke API: <https://api.openai.com/v1/completions>

Documentation here: <https://platform.openai.com/docs/api-reference/completions>

## Steps

On update of Summary field of Conversation (once transcription of recording is done), trigger a Power Automate to invoke ChatGPT API using a HTTP connector and return the Emotion and Sentiment from the API. Use Parse Json to read and update the Emotion and Sentiment back on the Conversation record in Dataverse.

Steps

1. Navigate to <https://make.powerapps.com/> and open Solution.
2. Open Solutions from left hand navigation

Graphical user interface, application

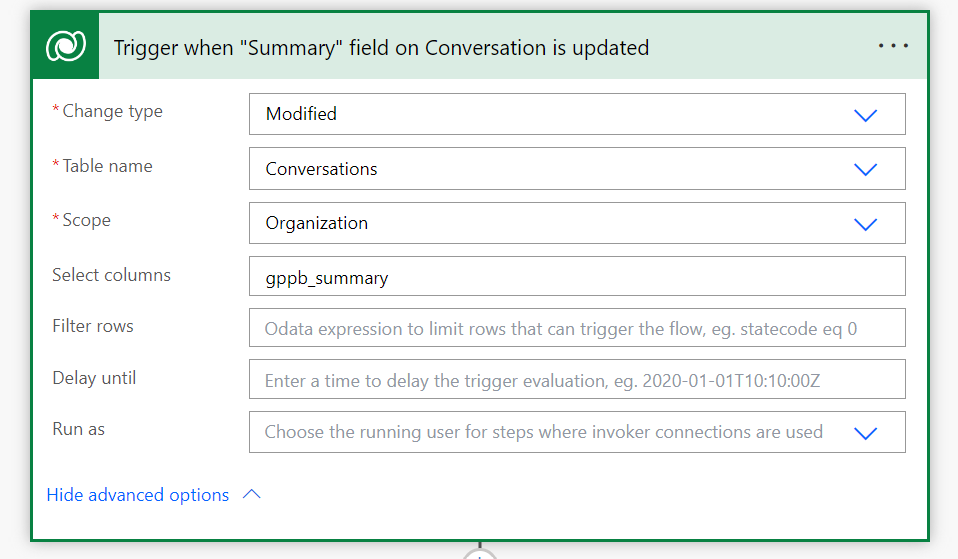
Description automatically generated

1. Create a new solution or open an existing one
2. Within Solution window, Click on “New” -> “Automation” -> Cloud Flow -> “Automated”

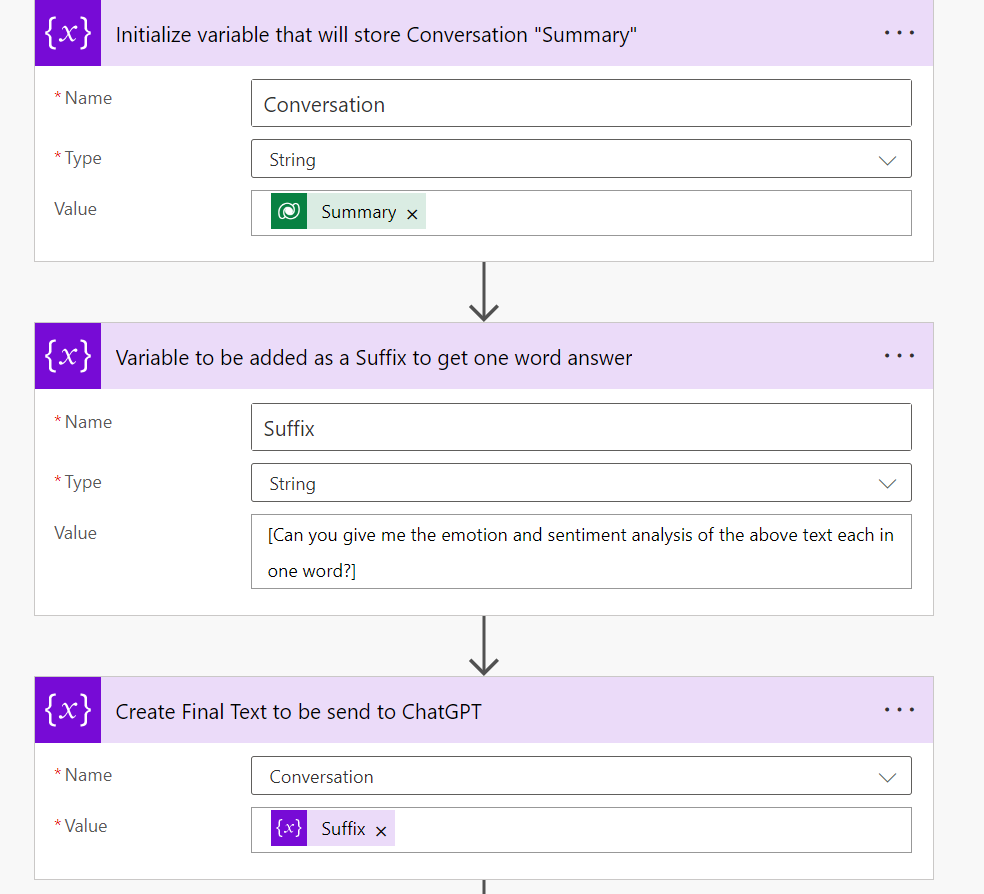
Graphical user interface, text, application

Description automatically generated

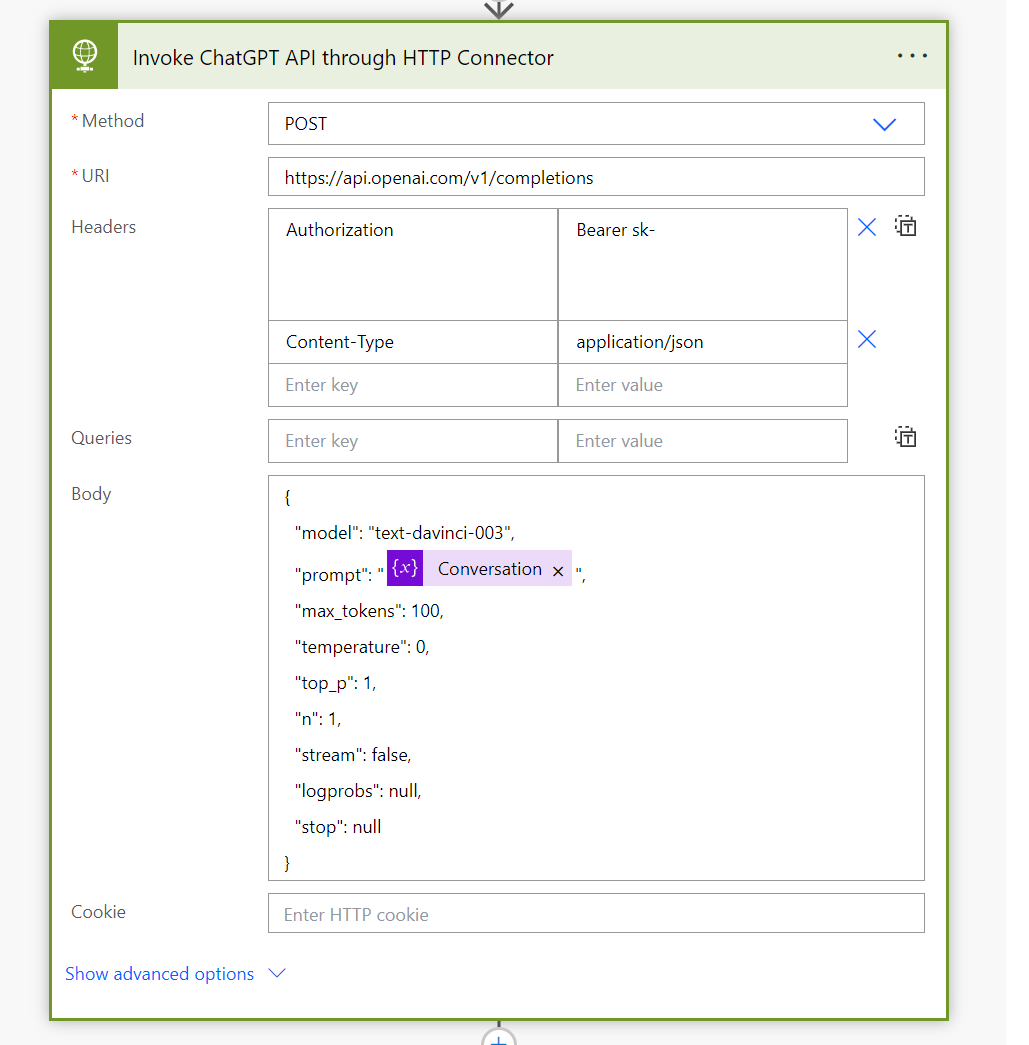
1. Start by adding a new Trigger as follows:
   1. Name: Conversation – Get Sentiment from ChatGPT
   2. Trigger type: Dataverse
   3. Trigger when “Record is modified”
   4. Table: Conversations
   5. Select column: Summary



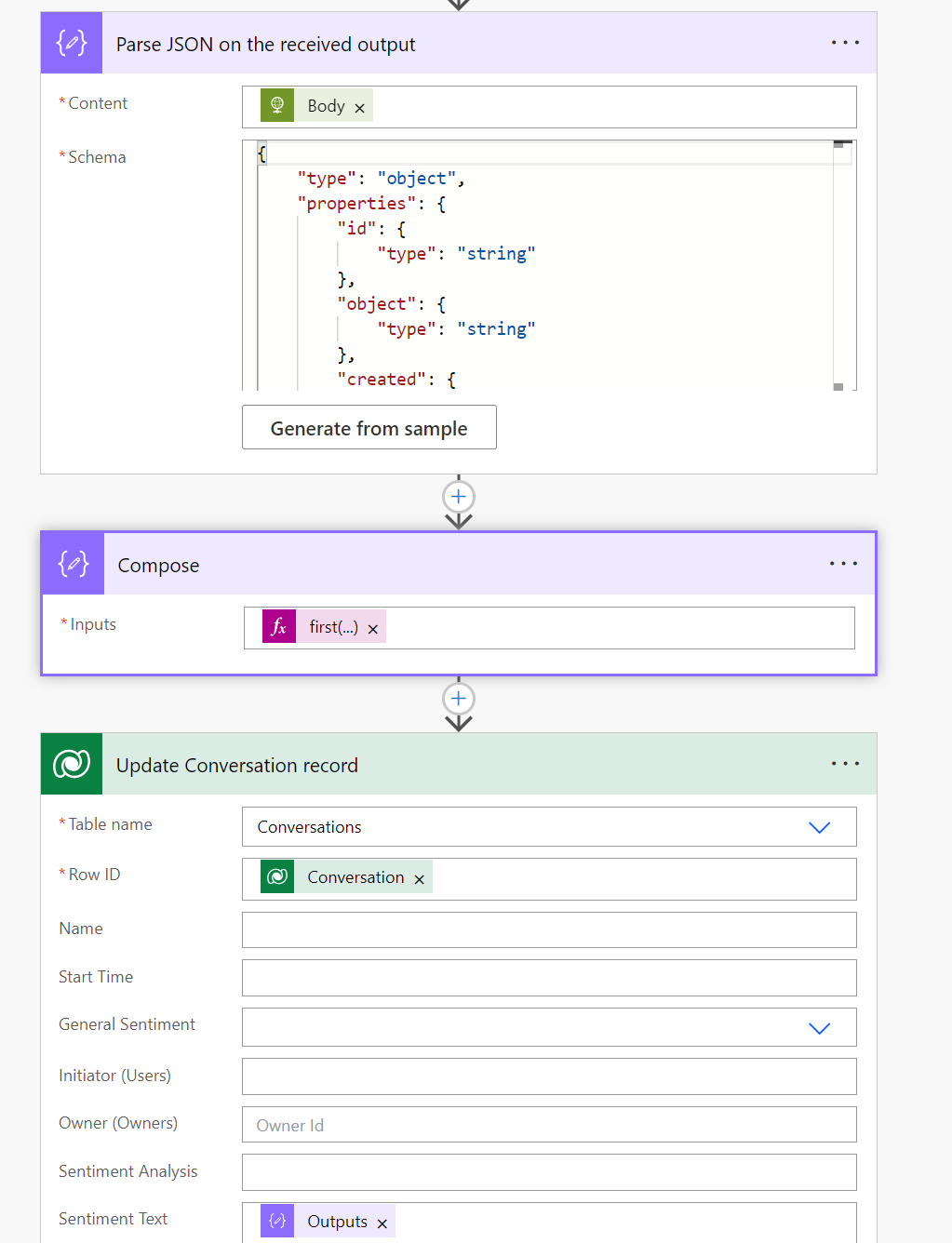
1. Set up a local Flow variable to store the “Summary” text and append specific “ChatGPT” formatted text at the end.



1. Use HTTP Connector to invoke ChatGPT API as follows:



1. Read the result and update Dataverse record
   1. We can use the following expression to get the desired text from the result set: first(body('Parse\_JSON')?['choices'])['text']



The final Cloud flow will look like this

Graphical user interface, application, Teams

Description automatically generated

# Appendix

## Azure function app

**URL** <https://gppbfunction.azurewebsites.net/api/Speech2Conversation>

### Security

Function key is uZRlZNQafWOXIJYCprrdCvlWOaUWQWVvsv58M2Dl3rVGW0cTui3caA==

When connecting add the following header

|  |  |
| --- | --- |
| Key | Value |
| x-functions-key | uZRlZNQafWOXIJYCprrdCvlWOaUWQWVvsv58M2Dl3rVGW0cTui3caA== |

### Parameters

Pass query parameter id=123

e.g. <https://gppbfunction.azurewebsites.net/api/Speech2Conversation?id=123>

### Request Type

Accepts both get and post

### Response

JSON array containing a sentence’s text, start time, and user. Right now it is static, eventually it will pick the file associated with the record id passed and convert speech to text (wip).

[

    {

        "Text": "Hello",

        "StartTimeInMilliseconds": 0,

        "User": "1"

    },

    {

        "Text": "Welcome",

        "StartTimeInMilliseconds": 1500,

        "User": "3"

    }

]

