### Step 1: Pre-Requisites

Before starting, you'll need an Azure subscription, Azure Speech Services, PowerAutomate, Power Apps Canvas App, CloudConvert, and an OpenAI API key that can access the Davinci-003 model.

Ensure you have all necessary permissions and access to create resources and configure services in Azure.

#### Step 2: Create a PowerApps Canvas App

- 1. Create a **new app** and select the **Phone template**.
- 2. Add a **Microphone control** to the screen.
- 3. In the **OnStop**\_property of the Microphone control, add the following code:

```
ClearCollect(RecordCollection, Recorder.Audio);
Set(searchText, JSON(RecordCollection, JSONFormat.IncludeBinaryData));
```

```
Set(result,
AudioText2.Run(JSON(First(RecordCollection), JSONFormat.IncludeBinaryData)).result);
Set(searchText, Left(result, Len(result)));
```

4. Add a Label control to the screen and set its Text property to:

Text(SearchText)

5. Add a **Send button** to the screen.



### **Step 3: Set up Azure Speech Services**

1. Create a new **Speech Service resource in Azure**.

2. Obtain the Endpoint URL and Subscription Key for the Speech Service resource and keep them handy.

Step 4: Create a Power Automate Flow

- 1. Create a new Flow in **Power Automate from a PowerApps Trigger**.
- 2. Add a **Compose Inputs** and a **Parse JSON** with the Outputs from Compose

PowerApps		0						
	$\checkmark$							
{		0						
*Inputs	Compose_Inputs ×							
+ *								
<i></i> <b>₽ ₽ a r s € P a r s € P a r s € F S O N S S O N S S S S S S S S S S</b>		0						
*Content	(/) Outputs ×							
*Schema	<pre>"type": "object", "properties": {     "type": {         "type": "string"         },         "properties": {         "type": "object",         "properties": {             "Url": {         Generate from sample         </pre>							

**3.** Add a **CloudConvert** Connector and add Canvas App and use in the Flow and it will convert the audio format from webm to wav.

4. Add a **Cognitive Services** action to the Flow and configure it to use the Speech Service resource you created in Azure.

	$\checkmark$			
Convert File			··· (9)	
*Input File Content	$f_{\rm X}$ dataUriToBinar $ imes$			
		Add dynar	nic content +	Add an expression to do basic things like access
*Input Filename	convert.webm		convert, and compare values. Learn more about	
*Output Format	wav 🗸 🗸			dynamic content.
Input Format	Input Format		$\sim$	Dynamic content Expression
Output Filename	The filename of the output file. If it is not provided, input filename with output			$f_x$ dataUriToBinary(body('Parse_JSON')?['Varse_JSON')?['Varse_JSON')?['Varse_JSON')?['Varse_JSON')?['Varse_JSON']
				Update
Ф нттр	· · ·		···· ⑦	Format your data
*Method	POST		~	Format data by examples Provide examples and we'll suggest an expression
* URI	https://eastus.stt.speech.microsoft.com/speech/recognition/conversation/cog		rsation/cog	String functions See more
	nitiveservices/v1?language=en-l	JS		fx concat(text_1, text_2?,) Combines any number of strings together
Headers	Ocp-Apim-Subscription-Key	<speech key="" text="" to=""></speech>	× 🖻	
	Content-type	audio/wav	×	Collection See more
	Enter key	Enter value		fe contains(collection, value) Returns true if a dictionary contains a key, if an array cont.
Queries	Enter key	Enter value	ŝ	fx length(collection)
Body	Sody ×			sort(collection)
Cookie	Enter HTTP cookie			Returns an array sorted in ascending order
	L			reverse(collection)

- 5. Create a Parse JSON to take the Results from Azure Cognitive Services Speech to Text.
- 6. Create a **Compose 2** to parse the **DISPLAY TEXT** property from the **BODY** of the JSON.

7. Select **RESPOND TO A POWERAPP OR FLOW**, and Select **TEXT**, name the Result and Select **Outputs** from the Compose 2.

6. Return the results in the form of TEXT back to the Canvas App's Label Control.

	PowerApps		Os 🗸
		$\downarrow$	•
{0}	Compose		Os
		$\downarrow$	
<i>{ø}</i>	Parse JSON		Os
		$\downarrow$	
٨	Convert File		8s
		$\downarrow$	
₽	НТТР		3s
		$\downarrow$	
<i>{ø}</i>	Parse JSON 2		0s
		$\downarrow$	
<i>{ø}</i>	Compose 2		Os
		$\downarrow$	
	Respond to a PowerApp or flow		0s

# Step 5: Set up OpenAI ChatGPT Davinci-003

- 1. Create an **OpenAl API key** that can access the Davinci-003 model.
- 2. Use the key to call the OpenAI GPT-3 API.

## Step 6: Add the Chatbot Interface to Your PowerApps Canvas App

- 1. You should have a Label control on your screen to capture user Voice input.
- 2. Add a **Gallery control** to the screen to display the chatbot responses.
- 3. In the **OnSelect** property of the **Send button**, add the following code:

ClearCollect(varMyColl, ChatGPT.Completions({'Content-

Type':"application/JSON",model:"text-davinci-

```
003",prompt:Label2.Text,max_tokens:2000,temperature:0,top_p:1,n:1,stream: false
,stop:"\n"}));
```



Step 7: Test and Deploy Your Voice-Enabled PowerApps Canvas App

1. **Test the app** by **speaking into the Microphone control** and checking if the recognized text appears in the Label control.

# 2. **Test the chatbot by Sending the results** from the Label control to check if the chatbot's **responses appear in the Gallery control**.

3. Deploy the app to your organization's environment and make it available to end-users.

**Note:** Don't forget to replace the **<YOURFLOW>** and **<YOURGALLERY>** placeholders with the actual names of your Flow and Gallery controls, respectively. Also, ensure that you have the correct input/output parameters set up for your Flow and OpenAI GPT-3 API.