Develop an audio-enabled chat app

In this exercise, you use the *Phi-4-multimodal-instruct* generative AI model to generate responses to prompts that include audio files. You'll develop an app that provides AI assistance for a produce supplier company by using Azure AI Foundry and the Python OpenAI SDK to summarize voice messages left by customers.

While this exercise is based on Python, you can develop similar applications using multiple language-specific SDKs; including:

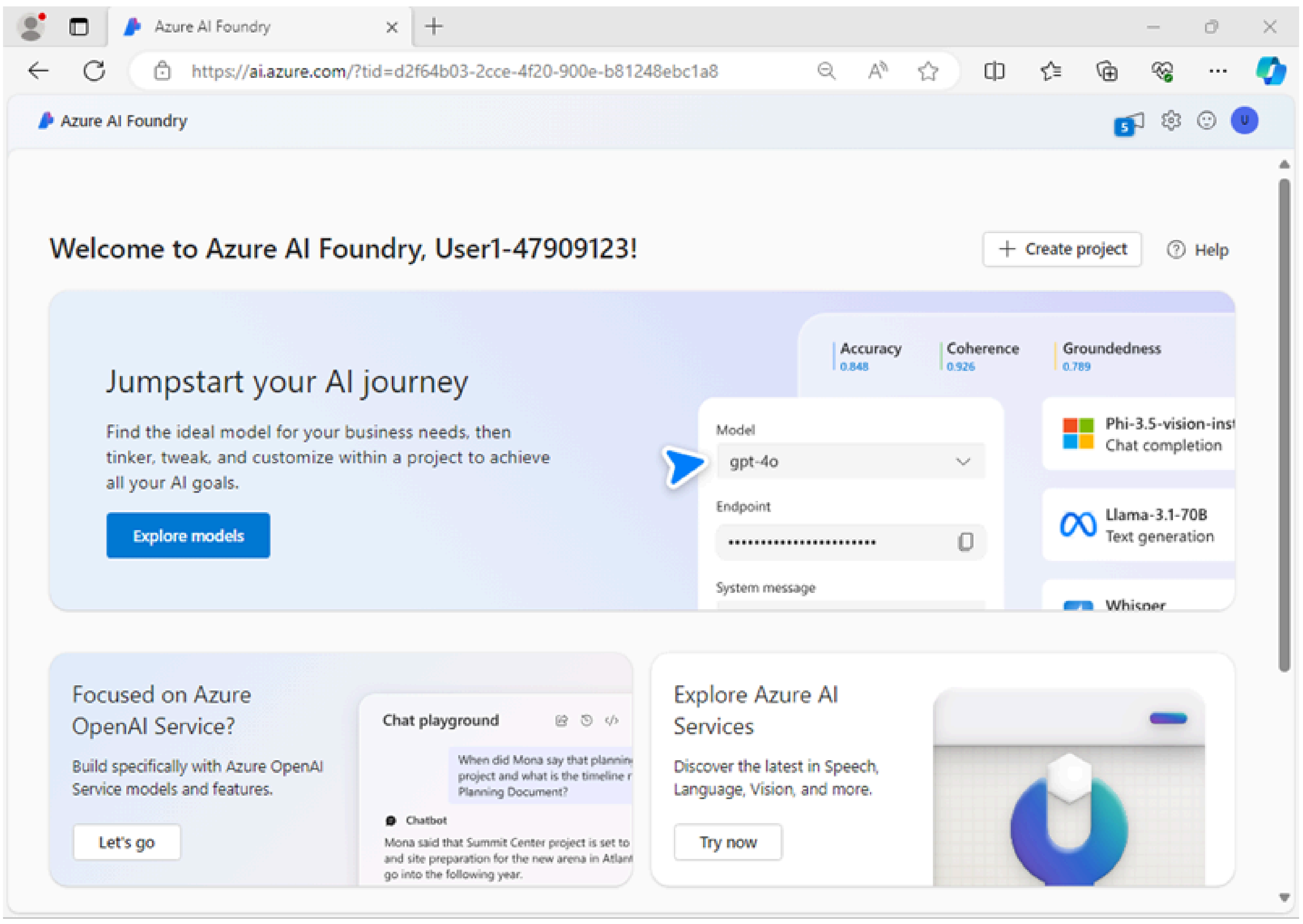
- Azure AI Projects for Python
- OpenAI library for Python
- Azure AI Projects for Microsoft .NET
- Azure OpenAI client library for Microsoft .NET
- Azure AI Projects for JavaScript
- Azure OpenAI library for TypeScript

This exercise takes approximately **30** minutes.

Create an Azure AI Foundry project

Let's start by deploying a model in an Azure AI Foundry project.

1. In a web browser, open the Azure AI Foundry portal at `https://ai.azure.com` and sign in using your Azure credentials. Close any tips or quick start panes that are opened the first time you sign in, and if necessary use the **Azure AI Foundry** logo at the top left to navigate to the home page, which looks similar to the following image:


Azure AI Foundry
https://ai.azure.com/?tid=d2f64b03-2cce-4f20-900e-b81248ebc1a8
Welcome to Azure AI Foundry, User1-47909123!
Create project
Help
Jumpstart your AI journey
Find the ideal model for your business needs, then tinker, tweak, and customize within a project to achieve all your AI goals.
Explore models
Accuracy 0.848
Coherence 0.926
Groundedness 0.789
Model
gpt-4o
Endpoint
System message
Phi-3.5-vision-inst
Chat completion
Llama-3.1-70B
Text generation
Whisper
Focused on Azure OpenAI Service?
Build specifically with Azure OpenAI Service models and features.
Let's go
Chat playground
Explore Azure AI Services
Discover the latest in Speech, Language, Vision, and more.
Try now

2. In the home page, in the **Explore models and capabilities** section, search for the `Phi-4-multimodal-instruct` model; which we'll use in our project.

3. In the search results, select the **Phi-4-multimodal-instruct** model to see its details, and then at the top of the page for the model, select **Use this model**.

4. When prompted to create a project, enter a valid name for your project and expand **Advanced options**.

5. Select **Customize** and specify the following settings for your hub:

   - **Azure AI Foundry resource**: *A valid name for your Azure AI Foundry resource*

- **Subscription**: *Your Azure subscription*
- **Resource group**: *Create or select a resource group*
- **Region**: *Select any **AI Foundry recommended****

> * Some Azure AI resources are constrained by regional model quotas. In the event of a quota limit being exceeded later in the exercise, there's a possibility you may need to create another resource in a different region. You can check the latest regional availability for specific models in the Azure AI Foundry documentation

6. Select **Create** and wait for your project to be created.

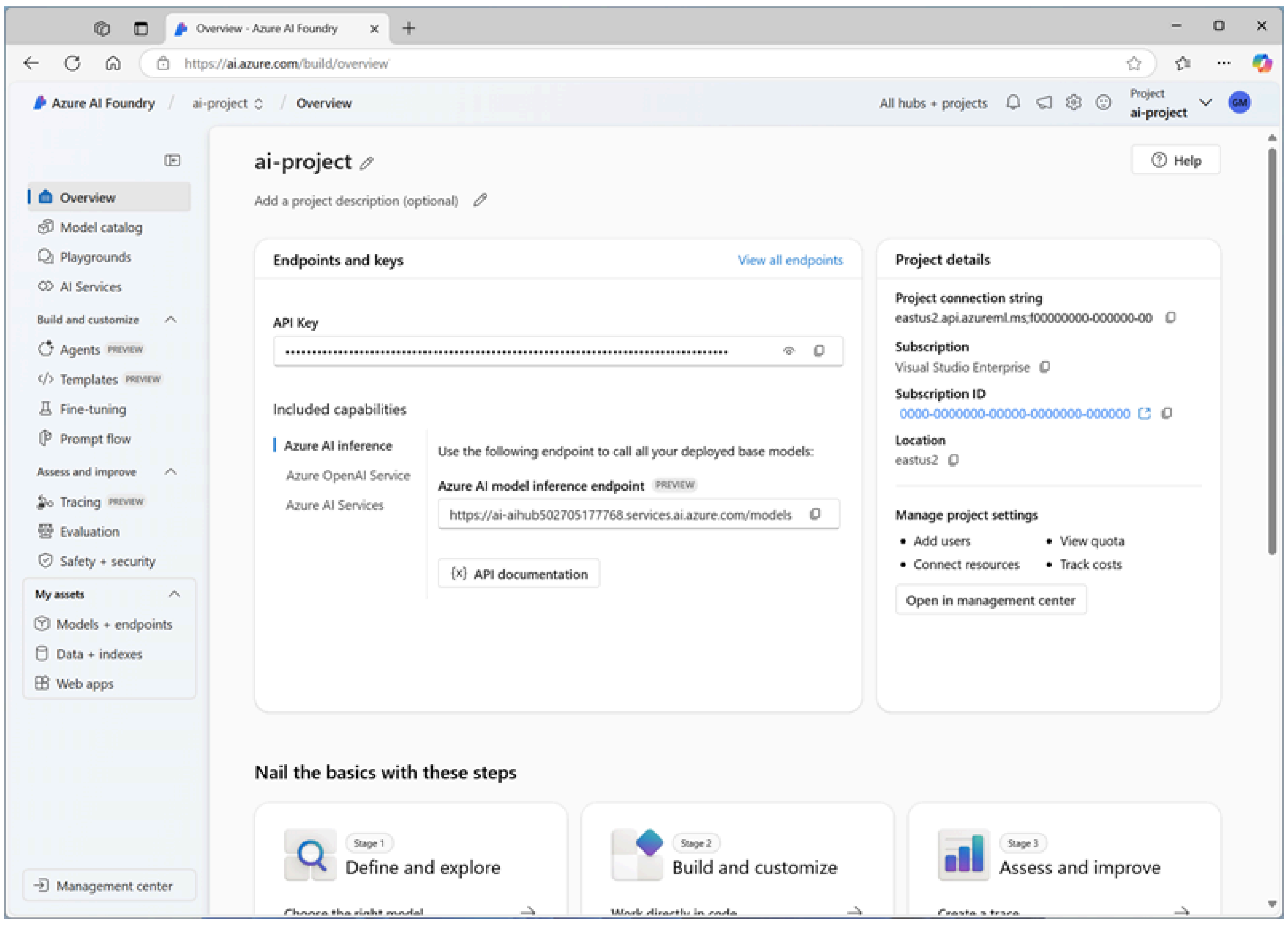
   It may take a few moments for the operation to complete.

7. Select **Agree and Proceed** to agree to the model terms, then select **Deploy** to complete the Phi model deployment.

8. When your project is created, the model details will be opened automatically. Note the name of your model deployment; which should be **Phi-4-multimodal-instruct**

9. In the navigation pane on the left, select **Overview** to see the main page for your project; which looks like this:

   > **Note**: If an *Insufficient permissions** error is displayed, use the **Fix me** button to resolve it.



ai-project

Add a project description (optional)

Endpoints and keys

API Key

Included capabilities

Azure AI inference | Azure OpenAI Service | Azure AI Services

Use the following endpoint to call all your deployed base models:

Azure AI model inference endpoint PREVIEW

https://ai-aihub502705177768.services.ai.azure.com/models

Project details

Project connection string
eastus2.api.azureml.ms;f00000000-000000-00

Subscription
Visual Studio Enterprise

Subscription ID
0000-0000000-00000-0000000-000000

Location
eastus2

Manage project settings
- Add users
- View quota
- Connect resources
- Track costs

Nail the basics with these steps

Create a client application

Now that you deployed a model, you can use the Azure AI Foundry and Azure AI Model Inference SDKs to develop an application that chats with it.

> **Tip**: You can choose to develop your solution using Python or Microsoft C#. Follow the instructions in the appropriate section for your chosen language.

Prepare the application configuration

1. In the Azure AI Foundry portal, view the **Overview** page for your project.

2. In the **Project details** area, note the **Azure AI Foundry project endpoint**. You'll use this endpoint to connect to your project in a client application.
3. Open a new browser tab (keeping the Azure AI Foundry portal open in the existing tab). Then in the new tab, browse to the Azure portal at `https://portal.azure.com`; signing in with your Azure credentials if prompted.

   Close any welcome notifications to see the Azure portal home page.
4. Use the **[>_]** button to the right of the search bar at the top of the page to create a new Cloud Shell in the Azure portal, selecting a ***PowerShell*** environment with no storage in your subscription.

   The cloud shell provides a command-line interface in a pane at the bottom of the Azure portal. You can resize or maximize this pane to make it easier to work in.

   > **Note**: If you have previously created a cloud shell that uses a *Bash* environment, switch it to ***PowerShell***.
5. In the cloud shell toolbar, in the **Settings** menu, select **Go to Classic version** (this is required to use the code editor).

   Ensure you've switched to the classic version of the cloud shell before continuing.
6. In the cloud shell pane, enter the following commands to clone the GitHub repo containing the code files for this exercise (type the command, or copy it to the clipboard and then right-click in the command line and paste as plain text):

   Code Copy
   ```
   rm -r mslearn-ai-audio -f
   git clone https://github.com/MicrosoftLearning/mslearn-ai-language
   ```

   > **Tip**: As you paste commands into the cloudshell, the ouput may take up a large amount of the screen buffer. You can clear the screen by entering the `cls` command to make it easier to focus on each task.
7. After the repo has been cloned, navigate to the folder containing the application code files:

   Code Copy
   ```
   cd mslearn-ai-language/Labfiles/09-audio-chat/Python
   ```
8. In the cloud shell command line pane, enter the following command to install the libraries you'll use:

   Code Copy
   ```
   python -m venv labenv
   ./labenv/bin/Activate.ps1
   pip install -r requirements.txt azure-identity azure-ai-projects openai
   ```
9. Enter the following command to edit the configuration file that has been provided:

   Code Copy
   ```
   code .env
   ```

   The file should open in a code editor.

10. In the code file, replace the **your_project_endpoint** placeholder with the endpoint for your project (copied from the project **Overview** page in the Azure AI Foundry portal), and the **your_model_deployment** placeholder with the name you assigned to your Phi-4-multimodal-instruct model deployment.
11. After you replace the placeholders, in the code editor, use the **CTRL+S** command or **Right-click > Save** to save your changes and then use the **CTRL+Q** command or **Right-click > Quit** to close the code editor while keeping the cloud shell command line open.

Write code to connect to your project and get a chat client for your model

Tip: As you add code, be sure to maintain the correct indentation.

1. Enter the following command to edit the code file:

Code Copy

```
code audio-chat.py
```

2. In the code file, note the existing statements that have been added at the top of the file to import the necessary SDK namespaces. Then, Find the comment **Add references**, add the following code to reference the namespaces in the libraries you installed previously:

Code Copy

```
# Add references
from azure.identity import DefaultAzureCredential
from azure.ai.projects import AIProjectClient
```

3. In the **main** function, under the comment **Get configuration settings**, note that the code loads the project connection string and model deployment name values you defined in the configuration file.
4. Find the comment **Initialize the project client** and add the following code to connect to your Azure AI Foundry project:

Tip: Be careful to maintain the correct indentation level for your code.

Code Copy

```
# Initialize the project client
project_client = AIProjectClient(
    credential=DefaultAzureCredential(
        exclude_environment_credential=True,
        exclude_managed_identity_credential=True
    ),
    endpoint=project_endpoint,
)
```

5. Find the comment **Get a chat client**, add the following code to create a client object for chatting with your model:

Code Copy

```
# Get a chat client
openai_client = project_client.get_openai_client(api_version="2024-10-21")
```

Write code to submit an audio-based prompt

Before submitting the prompt, we need to encode the audio file for the request. Then we can attach the audio data to the user's message with a prompt for the LLM. Note that the code includes a loop to allow the user to input a prompt until they enter "quit".

1. Under the comment **Encode the audio file**, enter the following code to prepare the following audio file:

   Code Copy

   ```
   # Encode the audio file
   file_path = "https://github.com/MicrosoftLearning/mslearn-ai-
   language/raw/refs/heads/main/Labfiles/09-audio-chat/data/avocados.mp3"
   response = requests.get(file_path)
   response.raise_for_status()
   audio_data = base64.b64encode(response.content).decode('utf-8')
   ```

2. Under the comment **Get a response to audio input**, add the following code to submit a prompt:

   Code Copy

   ```
   # Get a response to audio input
   response = openai_client.chat.completions.create(
      model=model_deployment,
      messages=[
          {"role": "system", "content": system_message},
          { "role": "user",
              "content": [
              {
                  "type": "text",
                  "text": prompt
              },
              {
                  "type": "input_audio",
                  "input_audio": {
                      "data": audio_data,
                      "format": "mp3"
                  }
              }
          ] }
      ]
   )
   print(response.choices[0].message.content)
   ```

3. Use the **CTRL+S** command to save your changes to the code file. You can also close the code editor (**CTRL+Q**) if you like.

Sign into Azure and run the app

1. In the cloud shell command-line pane, enter the following command to sign into Azure.

   Code Copy

   ```
   az login
   ```

You must sign into Azure - even though the cloud shell session is already authenticated.

> **Note**: In most scenarios, just using *az login* will be sufficient. However, if you have subscriptions in multiple tenants, you may need to specify the tenant by using the *--tenant* parameter. See Sign into Azure interactively using the Azure CLI for details.

2. When prompted, follow the instructions to open the sign-in page in a new tab and enter the authentication code provided and your Azure credentials. Then complete the sign in process in the command line, selecting the subscription containing your Azure AI Foundry hub if prompted.

3. In the cloud shell command-line pane, enter the following command to run the app:

   Code Copy

   ```
   python audio-chat.py
   ```

4. When prompted, enter the prompt

   Code Copy

   ```
   Can you summarize this customer's voice message?
   ```

5. Review the response.

Use a different audio file

1. In the code editor for your app code, find the code you added previously under the comment **Encode the audio file**. Then modify the file path url as follows to use a different audio file for the request (leaving the existing code after the file path):

   Code Copy

   ```
   # Encode the audio file
   file_path = "https://github.com/MicrosoftLearning/mslearn-ai-
   language/raw/refs/heads/main/Labfiles/09-audio-chat/data/fresas.mp3"
   ```

   The new file sounds like this:

2. Use the **CTRL+S** command to save your changes to the code file. You can also close the code editor (**CTRL+Q**) if you like.

3. In the cloud shell command line pane beneath the code editor, enter the following command to run the app:

   Code Copy

   ```
   python audio-chat.py
   ```

4. When prompted, enter the following prompt:

   Code Copy

```
Can you summarize this customer's voice message? Is it time-sensitive?
```

5. Review the response. Then enter `quit` to exit the program.

   > **Note**: In this simple app, we haven't implemented logic to retain conversation history; so the model will treat each prompt as a new request with no context of the previous prompt.

6. You can continue to run the app, choosing different prompt types and trying different prompts. When you're finished, enter `quit` to exit the program.

   If you have time, you can modify the code to use a different system prompt and your own internet-accessible audio files.

   > **Note**: In this simple app, we haven't implemented logic to retain conversation history; so the model will treat each prompt as a new request with no context of the previous prompt.

Summary

In this exercise, you used Azure AI Foundry and the Azure AI Inference SDK to create a client application uses a multimodal model to generate responses to audio.

Clean up

If you've finished exploring Azure AI Foundry, you should delete the resources you have created in this exercise to avoid incurring unnecessary Azure costs.

1. Return to the browser tab containing the Azure portal (or re-open the Azure portal at `https://portal.azure.com` in a new browser tab) and view the contents of the resource group where you deployed the resources used in this exercise.
2. On the toolbar, select **Delete resource group**.
3. Enter the resource group name and confirm that you want to delete it.