Create a custom copilot that uses your own data

Retrieval Augmented Generation (RAG) is a technique used to build applications that integrate data from custom data sources into a prompt for a generative AI model. RAG is a commonly used pattern for developing custom *copilots* - chat-based applications that use a language model to interpret inputs and generate appropriate responses.

In this exercise, you'll use Azure AI Studio to integrate custom data into a generative AI prompt flow.

This exercise takes approximately **45** minutes.

Before you start

To complete this exercise, your Azure subscription must be approved for access to the Azure OpenAI service. Fill in the registration form to request access to Azure OpenAI models.

Create an Azure AI Search resource

Your copilot solution will integrate custom data into a prompt flow. To support this integration, you'll need an Azure AI Search resource with which to index your data.

1. In a web browser, open the Azure portal at `https://portal.azure.com` and sign in using your Azure credentials.
2. On the home page, select **+ Create a resource** and search for `Azure AI Search`. Then create a new Azure AI Search resource with the following settings:
    - **Subscription**: *Select your Azure subscription*
    - **Resource group**: *Select or create a resource group*
    - **Service name**: *Enter a unique service name*
    - **Location**: *Make a **random** choice from any of the following regions**
        - Australia East
        - Canada East
        - East US
        - East US 2
        - France Central
        - Japan East
        - North Central US
        - Sweden Central
        - Switzerland
    - **Pricing tier**: Standard

> ! * Later, you're going to create an Azure AI Hub (which includes an Azure OpenAI service) in the same region as your Azure AI Search resource. Azure OpenAI resources are constrained at the tenant level by regional quotas. The listed regions include default quota for the model type(s) used in this exercise. Randomly choosing a region reduces the risk of a single region reaching its quota limit in scenarios where you are sharing a tenant with other users. In the event of a quota limit being reached later in the exercise, there's a possibility you may need to create another Azure AI hub in a different region.

3. Wait for your Azure AI Search resource deployment to be completed.

Create an Azure AI project

Now you're ready to create an Azure AI Studio project and the Azure AI resources to support it.

1. In a web browser, open Azure AI Studio at `https://ai.azure.com` and sign in using your Azure credentials.
2. On the Azure AI Studio **Home** page, select **+ New project**. Then, in the **Create a project** wizard, create a project with the following settings:
    - **Project name**: *A unique name for your project*
    - **Hub**: *Create a new resource with the following settings:*
        - **Hub name**: *A unique name*
        - **Azure Subscription**: *Your Azure subscription*
        - **Resource group**: *Select the resource group containing your Azure AI Search resource*
        - **Location**: *The same location as your Azure AI Search resource*
        - **Azure OpenAI**: (New) *Autofills with your selected hub name*
        - **Azure AI Search**: *Select your Azure AI Search resource*
3. Wait for your project to be created.

Deploy models

You need two models to implement your solution:

- An *embedding* model to vectorize text data for efficient indexing and processing.
- A model that can generate natural language responses to questions based on your data.

1. In the Azure AI Studio, in your project, in the navigation pane on the left, under **Components**, select the **Deployments** page.
2. Create a new deployment of the **text-embedding-ada-002** model with the following settings:
    - **Deployment name**: `text-embedding-ada-002`
    - **Model version**: *Default*
    - **Advanced options**:
        - **Content filter**: *Default*
        - **Tokens per minute rate limit**: `5K`
3. Repeat the previous steps to deploy a **gpt-35-turbo-16k** model with the deployment name `gpt-35-turbo-16k`.

> ❗ **Note**: Reducing the Tokens Per Minute (TPM) helps avoid over-using the quota available in the subscription you are using. 5,000 TPM is sufficient for the data used in this exercise.

Add data to your project

The data for your copilot consists of a set of travel brochures in PDF format from the fictitious travel agency *Margie's Travel*. Let's add them to the project.

1. Download the zipped archive of brochures from `https://github.com/MicrosoftLearning/mslearn-ai-studio/raw/main/data/brochures.zip` and extract it to a folder named **brochures** on your local file system.
2. In Azure AI Studio, in your project, in the navigation pane on the left, under **Components**, select the **Data** page.
3. Select **+ New data**.
4. In the **Add your data** wizard, expand the drop-down menu to select **Upload files/folders**.
5. Select **Upload folder** and select the **brochures** folder.
6. Set the data name to `brochures`.

7. Wait for the folder t be uploaded and note that it contains several .pdf files.

Create an index for your data

Now that you've added a data source to your project, you can use it to create an index in your Azure AI Search resource.

1. In Azure AI Studio, in your project, in the navigation pane on the left, under **Components**, select the **Indexes** page.
2. Add a new index with the following settings:
    - **Source data**:
        - **Data source**: Data in Azure AI Studio
            - *Select the **brochures** data source*
    - **Index settings**:
        - **Select Azure AI Search service**: *Select the **AzureAISearch** connection to your Azure AI Search resource*
        - **Index name**: `brochures-index`
        - **Virtual machine**: Auto select
    - **Search settings**:
        - **Vector settings**: Add vector search to this search resource
        - **Select an embedding model**: *Select the default Azure OpenAI resource for your hub.*
3. Wait for the indexing process to be completed, which can take several minutes. The index creation operation consists of the following jobs:
    - Crack, chunk, and embed the text tokens in your brochures data.
    - Create the Azure AI Search index.
    - Register the index asset.

Test the index

Before using your index in a RAG-based prompt flow, let's verify that it can be used to affect generative AI responses.

1. In the navigation pane on the left, under **Project playground**, select the **Chat** page.
2. On the Chat page, in the Options panel, ensure that your **gpt-35-turbo-16k** model deployment is selected. Then, in the main chat session panel, submit the prompt `Where can I stay in New York?`
3. Review the response, which should be a generic answer from the model without any data from the index.
4. On the Setup panel, select the **Add your data** tab, and then add the **brochures-index** project index and select the **hybrid (vector + keyword)** search type.
5. After the index has been added and the chat session has restarted, resubmit the prompt `Where can I stay in New York?`
6. Review the response, which should be based on data in the index.

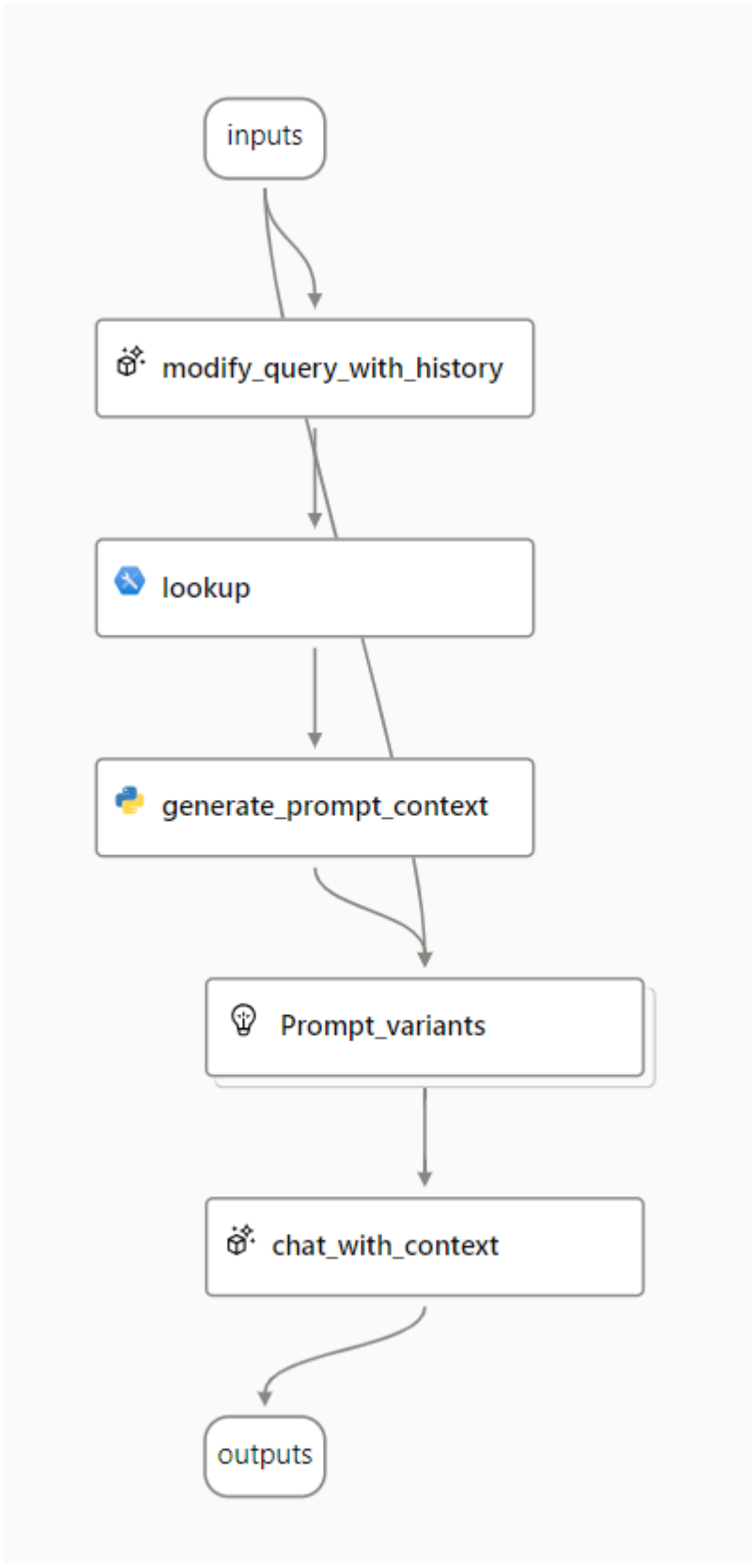
Use the index in a prompt flow

Your vector index has been saved in your Azure AI Studio project, enabling you to use it easily in a prompt flow.

1. In Azure AI Studio, in your project, in the navigation pane on the left, under **Tools**, select the **Prompt flow** page.
2. Create a new prompt flow by cloning the **Multi-Round Q&A on Your Data** sample in the gallery. Save your clone of this sample in a folder named `brochure-flow`.

    ▶ **Troubleshooting tip**: Permissions error

3. When the prompt flow designer page opens, review **brochure-flow**. Its graph should resemble the following image:



The sample prompt flow you are using implements the prompt logic for a chat application in which the user can iteratively submit text input to chat interface. The conversational history is retained and included in the context for each iteration. The prompt flow orchestrate a sequence of *tools* to:

- Append the history to the chat input to define a prompt in the form of a contextualized form of a question.
- Retrieve the context using your index and a query type of your own choice based on the question.
- Generate prompt context by using the retrieved data from the index to augment the question.
- Create prompt variants by adding a system message and structuring the chat history.
- Submit the prompt to a language model to generate a natural language response.

4. Use the **Start compute session** button to start the runtime compute for the flow.

   Wait for the runtime to start. This provides a compute context for the prompt flow. While you're waiting, in the **Flow** tab, review the sections for the tools in the flow.
5. In the **Inputs** section, ensure the inputs include:

   - **chat_history**
   - **chat_input**

   The default chat history in this sample includes some conversation about AI.

6. In the **Outputs** section, ensure that the output includes:

   - **chat_output** with value ${chat_with_context.output}

7. In the **modify_query_with_history** section, select the following settings (leaving others as they are):

   - **Connection**: *The default Azure OpenAI resource for your AI hub*

    - **Api**: chat
    - **deployment_name**: gpt-35-turbo-16k
    - **response_format**: {"type":"text"}

8. In the **lookup** section, set the following parameter values:

    - **mlindex_content**: *Select the empty field to open the Generate pane*

        - **index_type**: Registered Index
        - **mlindex_asset_id**: brochures-index:1
    - **queries**: ${modify_query_with_history.output}
    - **query_type**: Hybrid (vector + keyword)
    - **top_k**: 2

9. In the **generate_prompt_context** section, review the Python script and ensure that the **inputs** for this tool include the following parameter:

    - **search_result** *(object)*: ${lookup.output}

10. In the **Prompt_variants** section, review the Python script and ensure that the **inputs** for this tool include the following parameters:

    - **contexts** *(string)*: ${generate_prompt_context.output}
    - **chat_history** *(string)*: ${inputs.chat_history}
    - **chat_input** *(string)*: ${inputs.chat_input}

11. In the **chat_with_context** section, select the following settings (leaving others as they are):

    - **Connection**: Default_AzureOpenAI
    - **Api**: Chat
    - **deployment_name**: gpt-35-turbo-16k
    - **response_format**: {"type":"text"}

    Then ensure that the **inputs** for this tool include the following parameters:

    - **prompt_text** *(string)*: ${Prompt_variants.output}

12. On the toolbar, use the **Save** button to save the changes you've made to the tools in the prompt flow.
13. On the toolbar, select **Chat**. A chat pane opens with the sample conversation history and the input already filled in based on the sample values. You can ignore these.
14. In the chat pane, replace the default input with the question `Where can I stay in London?` and submit it.
15. Review the response, which should be based on data in the index.
16. Review the outputs for each tool in the flow.
17. In the chat pane, enter the question `What can I do there?`
18. Review the response, which should be based on data in the index and take into account the chat history (so "there" is understood as "in London").
19. Review the outputs for each tool in the flow, noting how each tool in the flow operated on its inputs to prepare a contextualized prompt and get an appropriate response.

Deploy the flow

Now that you have a working flow that uses your indexed data, you can deploy it as a service to be consumed by a copilot application.

> ❗ **Note**: Depending on the region and datacenter load, deployments can sometimes take a while. Feel free to move on to the challenge section below while it deploys or skip the testing of your deployment if you're short on time.

1. On the toolbar, select **Deploy**.
2. Create a deployment with the following settings:

    - **Basic settings**:

        - **Endpoint**: New

        - **Endpoint name**: *Use the default unique endpoint name*
        - **Deployment name**: *Use the default deployment endpoint name*
        - **Virtual machine**: Standard_DS3_v2
        - **Instance count**: 3
        - **Inferencing data collection**: Selected
    - **Advanced settings**:
        - *Use the default settings*
3. In Azure AI Studio, in your project, in the navigation pane on the left, under **Components**, select the **Deployments** page.
4. Keep refreshing the view until the **brochure-endpoint-1** deployment is shown as having *succeeded* under the **brochure-endpoint** endpoint (this may take a significant period of time).
5. When the deployment has succeeded, select it. Then, on its **Test** page, enter the prompt `What is there to do in San Francisco?` and review the response.
6. Enter the prompt `Where else could I go?` and review the response.
7. View the **Consume** page for the endpoint, and note that it contains connection information and sample code that you can use to build a client application for your endpoint - enabling you to integrate the prompt flow solution into an application as a custom copilot.

Challenge

Now you've experienced how to integrate your own data in a copilot built with the Azure AI Studio, let's explore further!

Try adding a new data source through the Azure AI Studio, index it, and integrate the indexed data in a prompt flow. Some data sets you could try are:

- A collection of (research) articles you have on your computer.
- A set of presentations from past conferences.
- Any of the datasets available in the Azure Search sample data repository.

Be as resourceful as you can to create your data source and integrate it in your prompt flow. Try out the new prompt flow and submit prompts that could only be answered by the data set you chose!

Clean up

To avoid unnecessary Azure costs and resource utilization, you should remove the resources you deployed in this exercise.

1. If you've finished exploring Azure AI Studio, return to the Azure portal at `https://portal.azure.com` and sign in using your Azure credentials if necessary. Then delete the resources in the resource group where you provisioned your Azure AI Search and Azure AI resources.