**Handout – UBI Chatbot**

The project can be mainly divided into three parts:  
1] Enable chatbot to access UBI SharePoint data by uploading it to blob storage.  
2] Create an Azure Cognitive Search Service.  
3] Deploy an Azure Open AI model and connect it to the data source.

The architecture of the project is shown below.

**Delphi Architecture**

A diagram of software development

Description automatically generated with medium confidence

Conceptual UBI Architecture based on DelphiA diagram of a software flow

Description automatically generated  
System Message Used:  
You are an AI assistant for Fluke UBI project.  
You will help people find information in the fluke UBI website.  
start the conversation by introducing yourself.  
The tabular data contains information about the glossary.   
You are given a tabular data with Nick Name, Name, Definition and Expert as its columns.  
Nick Name column refers to the most used name for any item.  
Name column refers to the actual name of the item.  
Definition column defines the item.  
More about system message is discussed in later sections.

With Azure Open AI we had two options to tweak the model as per our requirements. One is use our own data and the other is give the model sample questions and answers (few shot learning). We choose the first one as we wanted to make the model answer on enterprise data which was never seen by the model.

**1 – Enable chatbot to access UBI SharePoint data by uploading it to blob storage**

We need to start with gathering the data before we start uploading it to azure blob storage. In the first phase of this project, we connected the data to the SharePoint website and to the purview glossary in the second phase of this project.

**1-a SharePoint Website data**

* In order to get started, we understood the structure of the website data and converted that to txt files. Each txt file had information from different web pages. In our case we had useful information in 5 web pages of the website. Therefore, we had 5 txt files with text information of the website. One of the known limitations of our model is that it is not capable of handling audio and image data therefore we excluded those modes of data. The model can be used to extract the links to those modes of data if present in the text.  
  A screenshot of a computer

  Description automatically generated
* Once the data is prepared and ready to be fed, we created a separate container in an existing blob storage account.

A screenshot of a computer

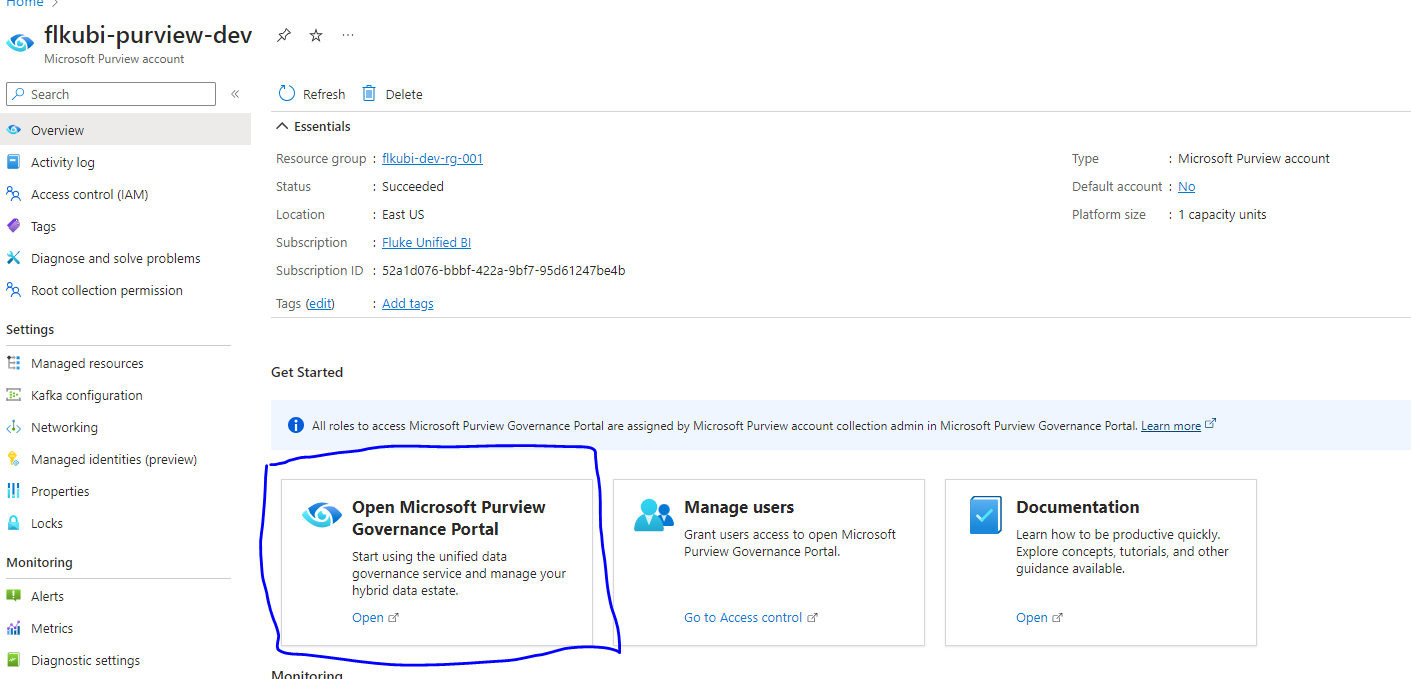
Description automatically generated

* We then manually uploaded the files to the container using the upload option. This can be done using the python SDK too.

A screenshot of a computer

Description automatically generated

**1-b Purview Glossary data**

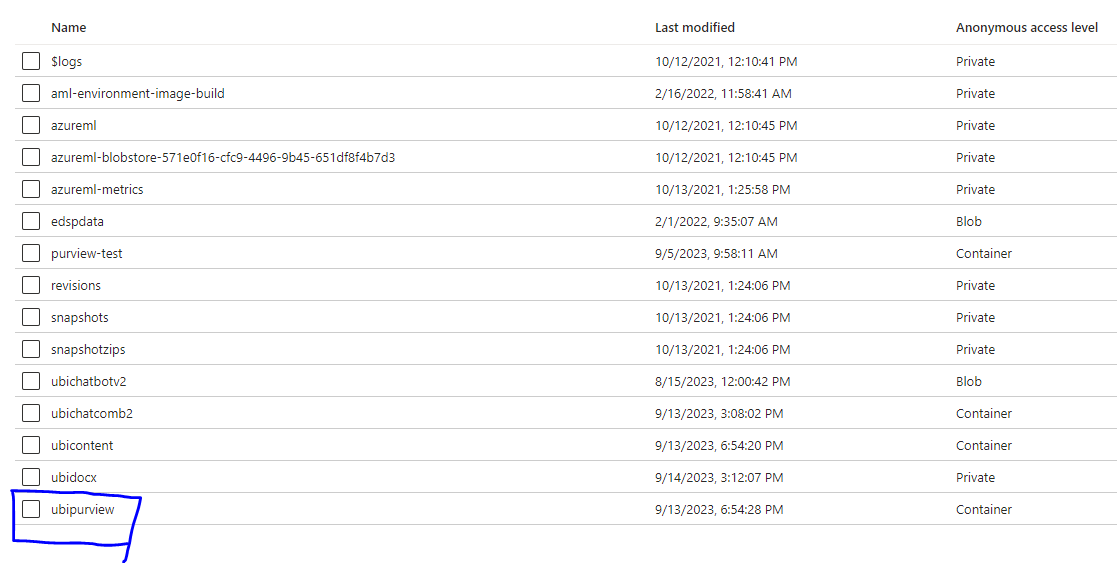
* To start with, we need to open the Microsoft purview studio from the overview of the Microsoft purview account. 
* Once in the Microsoft purview we redirected to the Glossary page.
* We then Selected the group of glossaries that we want to export and exported the terms to store it in a CSV file.  
  A screenshot of a computer

  Description automatically generated

A screenshot of a computer

Description automatically generated

* The downloaded CSV is then stored in a separate blob storage.



At this point all our data is stored in their respective blob storage and ready to be connected to the azure cognitive search service

**2 – Configure Azure Cognitive Search Service**

* To start with we connected the data by adding as a data source which can be later used. The data source option is the left-hand side menu in search management.

A screenshot of a computer

Description automatically generated

* Select Azure Blob storage as the data source and click on choose an existing connection and browse to the respective container and give a container name.

A screenshot of a computer

Description automatically generated

* We repeated this process for both the sources and saved them as UBIpurview and UBIcontent.
* Now we need to create an index using the indexes section in search management in the cognitive search service. Click on Add Index to create a new index.

A screenshot of a computer

Description automatically generated

* Now we added the fields using the add field button. Since we are using two datasets, we need to add fields for both the datasets for the same index. First for the CSV dataset of glossary we added each of the columns as a separate field and added content as a field for all the txt files.

A screenshot of a survey

Description automatically generated

It looked something like this after we added all the fields.

A screenshot of a computer

Description automatically generated

* After we added all the fields and gave it a name, we created an index using the create button.
* In the next step we need to create indexers for the created index. In our case we created couple of indexers for two different datasets in the two blob storage containers.
* We click on add indexer to create a new indexer. A screenshot of a computer

  Description automatically generated
* In the next page, we give the indexer a name and select the recently created index and add the respective dataset which we added earlier. We then schedule the indexer to automatically to sync with the storage every day by selecting the Schedule to Daily. By doing this we just have to add data to the blob storage and it will be reflected to the indexers therefore reflected to the UBI chatbot even if the model is deployed.

A screenshot of a computer

Description automatically generated

* We repeated this process for both the data source and created 2 indexers for both the data source.
* Now our cognitive search service is read to be connected to Azure Open AI and prepare the model for deployment.

**3 – Configure Azure Open AI**

* To start with we opened the Azure Open Ai studio by clicking on Go to Azure OpenAI studio on the azure Open AI

A screenshot of a computer

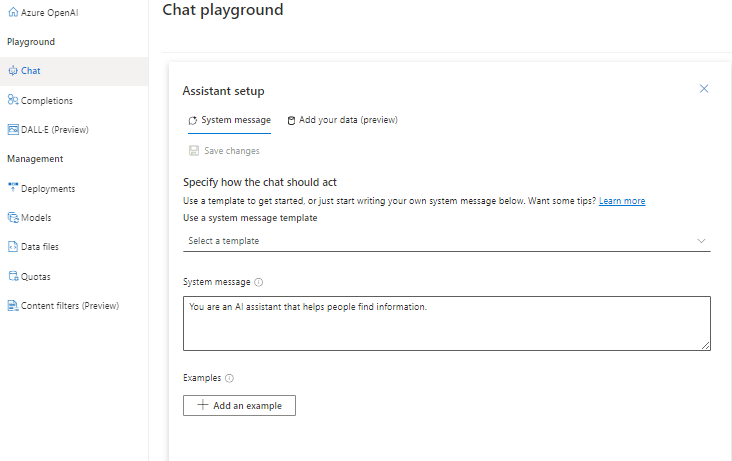
Description automatically generated

* Once in the studio we started first by deploying a GPT 3.5 turbo model and named it as UnifiedBIChatBot.  
  A screenshot of a computer

  Description automatically generated
* Once we have the model deployed we then Selected that model and went to the Chat section in the studio.

A screenshot of a computer

Description automatically generated

* There are two parts in using the model for enterprise data one is giving the Data and other is the system message where we can tell model more about the task we are trying to perform and give any special instructions.   
  
* After giving an appropriate system message we connected the data using the following specifications as shown below.  
  A screenshot of a computer

  Description automatically generated  
    
  A screenshot of a computer

  Description automatically generated
* Currently we selected the “Keyword” search type but we have an option to use semantic search which can result in better output. A screenshot of a computer

  Description automatically generated
* After selecting we then deploy the model that is connected to our data and is ready to respond based on the data.  
  A screenshot of a computer

  Description automatically generated
* We can either use the chat session in the portal for testing or deploy the model using the “Deploy to” function.  
  A screenshot of a computer

  Description automatically generatedA screenshot of a computer

  Description automatically generated