2023-12-01 TODOs:

* Talk a little bit about semantic kernel in the talk track instead of the lab
* Streamlit dashboard for website instead of front-end service
* Exercise 3 function calling in Python instead of .NET
* Exercise 5 live audio transcription: try to keep NER if possible but drop if too slow
* Exercise 6: focus on summarization, noun extraction, and extracting details from the call

Overview of Primary Tasks

* We'll want to generate some artificial data for those API calls, so we'd need a script to load data into Cosmos DB:
* Resort information, such as country, city, travel options to get to the resort, on-site amenities, and other common things people might ask for.
* Hotel information: number of rooms, types of rooms, room amenities, etc.
* Room availability
* Customer history: prior bookings, upcoming stays, prior call information and summaries
* Resort and customer reviews
* At least one sample customer call as an audio file
* I think exercise 2 should be somewhat straightforward, based on [this quickstart on chatting with OpenAI models using your own data](https://learn.microsoft.com/en-us/azure/ai-services/openai/use-your-data-quickstart?tabs=powershell%2Cpython&pivots=programming-language-csharp) and a [GitHub sample app for chat](https://github.com/microsoft/sample-app-aoai-chatGPT/tree/main). Key tasks in exercise 2:
* Prepare a FAQ dataset to augment the model. The data will be in Cosmos DB, so we could have them import it as part of this exercise.
* Add the FAQ data to OpenAI. This is where they'd tie in Azure AI Search to the Cosmos DB database.
* Test completions using the ChatGPT playground.
* Update the website to support this chat.
* Exercise 3 performs function calling, and I think the answer here is to include this information as part of the context for prompts. Key tasks:
* Implement a customer account information lookup function. We'd already have API endpoints for customer account info, so I'm not positive how much more effort there is in this task.
* Define the function for the Azure OpenAI Chat Completions API. There is [some documentation on function creation in Python](https://learn.microsoft.com/en-us/azure/ai-services/openai/how-to/function-calling?tabs=python), though I figure the languages like the .NET ones would work as well.
* Test the function call using the ChatGPT playground.
* Update the website to support this function.
* Exercise 4 covers implementing audio transcription. The [AI-Powered Call Center Intelligence accelerator](https://github.com/amulchapla/AI-Powered-Call-Center-Intelligence) and [conversational insights](https://github.com/microsoft/Customer-Service-Conversational-Insights-with-Azure-OpenAI-Services/tree/master) repos should hopefully give us a starting point for this. Key tasks include:
* Implement real-time speech to text in the agent web application
* Test speech-to-text using an audio file. We'd give them the audio file in the repo and might want to have 2 or 3 different fake calls with different people.
* Test speech-to-text with a microphone, which would be optional.
* Exercise 5 covers providing a live chat transcription to the ChatGPT model. My thought on this exercise is that we could focus on a "speak to an automated agent" type of functionality, so there'd be audio input, translation to text, and some additional info about the customer appended as part of the prompt. Key tasks in here:
* Update the app to accept voice input (using mechanisms from exercise 4) and perform speech to text. Take that text, along with customer information, and send it to the ChatGPT deployment (using mechanisms from exercise 3).
* Determine if the current call is compliant using natural language prompts. Compliance here means that the customer was informed that the call is being recorded. For this, we'd probably have them test with audio files. Alternatively, if we accept microphone input or have "one-way" audio files for testing, we might make this task implementing a check if the audio is relevant--that is, if it's actually related to resort information or if someone is misbehaving.
* Determine the main subject using custom entity extraction. Only do this if there’s enough time for [named entity extraction](https://learn.microsoft.com/en-us/azure/ai-services/language-service/custom-named-entity-recognition/overview) to load and process the data.
* Exercise 6 looks at generating a call summary. Key tasks:
* Generate a call summary using the Completions API and audio transcript. Write it out to the screen on the front end.
* Create an API call to write the summary to Cosmos DB in an existing call summaries table.
* Have the front end website make the back end API call to write the summary to Cosmos DB.