



Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

HumanTech

Technology for
Human Wellbeing Institute

Avatar-Bot

University

Jean-Pierre Gergie

Mandated by : HumanTech

Supervised by :

Dr. Mira El Kamali

Dr. Marine Capallera.

Date:
Summer 2022

Table of content

1	Introduction	6
1.1	Context and motivations	6
1.2	Goals	6
1.2.1	Main goals	6
1.2.2	Secondary goals	6
1.3	Report structure	6
2	Planning	7
3	State of the Art/Literature Review	8
4	Design/Conception	9
5	Implementation	12
6	Evaluation/Validation/Results	20
7	Conclusion	32
7.1	Interpretation of results	32
7.2	Limitations and Further work	32
7.3	Personal conclusion	32
8	Glossary	33
	References	34

List of figures

FIGURE 4-1 GENERAL ARCHITECTURE	9
FIGURE 4-2 WORKFLOW OF THE BOT- WHEN THE BOT RECEIVES A MESSAGE, IT WILL DETECT THE INTENT OF THE UTTERANCE AND DISPATCH TO THE RESPECTIVE DIALOG. THEN IT WILL MAKE ACCORDING API CALLS (IF NEEDED) AND RETURN A RESPONSE	11
FIGURE 5-1 BOT REGISTRATION	13
FIGURE 5-2- SET MEETING DIALOG; IF NEEDED, THE BOT WILL PROMPT THE USER TO PROVIDE THE MISSING ENTITY	14
FIGURE 5-3- SET_MEETING_DIALOG; WHEN THE BOT DETECT" ADD CALENDAR ENTRY" INTENT, THE SET_MEETING_DIALOG WILL BE INITIATED. THE LUIS RESULTS ALONG WITH THE EXTRACTED ENTITIES WILL BE PASSED AS OPTION (PARAMETER). IT WILL GO THROUGH THE STEPS IF NEEDED.	15
FIGURE 5-4 STATE MACHINE REPRESENTING THE WORKFLOW OF THE CLIENT APP	18
FIGURE 5-5 COMPLETE ARCHITECTURE OF THE ASSISTANT	19
FIGURE 6-11 ADDING A TASK	29

List of Tables

TABLE 1 - GANTT	7
TABLE 2- COMPARISON BETWEEN THE ALTERNATIVES.....	8
TABLE 3- PRE-BUILT DOMAINS.....	17

1 Introduction

1.1 Context and motivations

This project is part of an internship we held with HEIA in the HumanTech institute. During our journey we were supervised by Dr. Mira El Kamali and Dr. Marine Capallera. The idea behind the project is to create a virtual assistant that can replace a certain person during a meeting, in case this person is unavailable. The assistant would have access to a knowledge base where information about the user is saved. And for added efficiency, the assistant could set up meetings, remind the user about tasks, keep everyone aware of the owner schedule...etc.

1.2 Goals

For the motivations mentioned above, we were assigned to build an Avatar Bot that is able to hold general conversation, have knowledge about the owner, and have access to the owner's schedule. And by that we mean setting meetings, adding tasks, listing To-do, done and doing tasks...etc. The bot would also have a virtual humanoid form that will make the interaction more natural. Ultimately, the bot will be integrated in Microsoft Teams but this is not part of our internship.

1.2.1 Main goals

Implement a chatbot able to hold a conversation, understand intents, have knowledge about the owner, be able to save calendar dates and appointments

1.2.2 Secondary goals

The bot should be able to remind the owner of a given task and list what the owner was up to.

1.3 Report structure

In this report, we will first talk about the state of the art and advancement in chatbots, then we will talk about the conception and the design of our project. Later in this report we will talk about the implementation and finally we analyse the results.

2 Planning

The GANTT diagram is very useful to estimate the time necessary for the completion of each task. Even if it is not respected, it is interesting to see which task took more time than expected. Analyzing the causes of these mismatches could be useful to improve our abilities to estimate time needs for a given task or to detect hidden problems that could be resolved for future projects. The GANTT Diagram can be made using



Table 1 - GANTT

Tasks:

1. Specification
2. Researching technologies
3. Going more in depth in each platform. Choosing one platform based on it criteria score.
4. Thinking about Scenarios and how it could be implemented
5. Implementation
6. Testing
7. Fixing issues

3 State of the Art/Literature Review

Nowadays, there is multiple platforms to create a chatbot; but during this internship we mainly looked at the following three:

- Bot framework SDK¹
- RASA²
- Dialog Flow³

That being said, there is multiple criterion that differ one the other...

- Price
- Ease of implementation
- Ease of integration (Integration with another platform like outlook, or a database...etc)
- Personal preference

The following table (table 2) shows each alternative with it score:

Table 2- comparison between the alternatives

Criteria	Price	Ease of implementation	Ease of integration in Unity	Personal preference	Score
Criterion Weight	3/5	4/5	4/5	3.5/5	
Alternative					
RASA	5/5	2/5	3/5	4/5	3.37931
Bot Framework	3/5	4/5	4/5	4/5	3.793103
Dialog Flow	3/5	4/5	3/5	3/5	3.275862

By using MAUT method (Multi-Attribute Utility Theory (MAUT) is a structured methodology designed to handle the tradeoffs among multiple objectives) and by considering those criteria, we decided to use Bot framework SDK.

¹ <https://docs.microsoft.com/en-us/azure/bot-service/?view=azure-bot-service-4.0>

² <https://rasa.com/docs>

³ <https://cloud.google.com/dialogflow/docs>

4 Design/Conception

General design:

The project is composed of two main parts. A “frontend” where we implement the physical form of the Avatar. Here we will create a humanoid figure. The avatar will know how to speak, lip sync, interact with the users and ultimately make some animations. A “backend” where we implement the intelligence of the bot. Here we teach the bot how to respond to a given prompt. Furthermore, the bot will make different API calls to access calendars and manage projects tasks.

In this report we will only detail the backend part, but we will explain briefly the overall conception of the project.

Figure 4-1 shows the general architecture of the project.

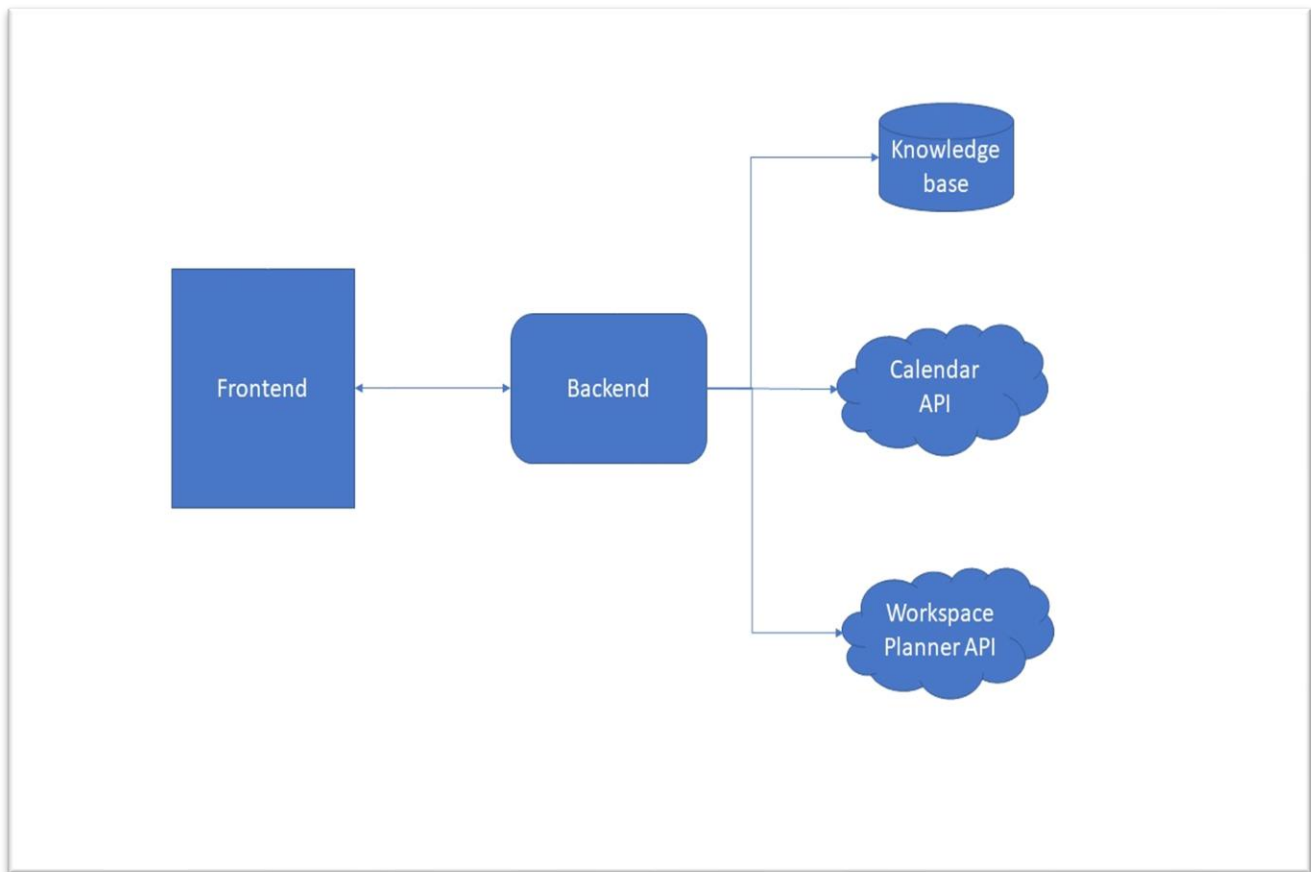


Figure 4-1 General architecture

Frontend:

The frontend is where implement the physical form of the Avatar. (We decided to use **Ready Player Me** SDK⁴). The speech-to-text is also implemented at this stage.

When the user prompt is ready it will be sent to the “backend” (where the intelligence of the bot is implemented). Then when we have a response , we convert the received text to voice format via a text-to-speech technology.

Backend:

The Backend is where the bot knows how to respond to a given prompt. We used multiple technologies to give the bot all the skills he has. We will talk more into details about these technologies in the implementation section.

Technologies:

- We used Azure LUIS⁵ (Language Understanding cloud-based conversational AI) for detecting the intents of a given prompt and extract all entities related
- Azure QnA service⁶ that allowed us to save and access static user information in a knowledge base
- Google Calendar API⁷, through which we were able to set meetings and appointments
- Notion API⁸, through which we were able to add and retrieve tasks of a certain project
- GPT3 API⁹, that allowed the bot to response to general information and hold a conversation even if the user prompts were random.
- Azure Bot Framework SDK python¹⁰, is the core of the project. Using this we were able organise the conversation flow and orchestrate all the other technologies.
- Azure Direct Line Channel¹¹, enables the connection between the frontend and the backend

⁴ <https://docs.readyplayer.me/ready-player-me/>

⁵ <https://docs.microsoft.com/en-us/azure/cognitive-services/luis/what-is-luis>

⁶ <https://docs.microsoft.com/en-us/azure/cognitive-services/qnamaker/overview/overview>

⁷ <https://developers.google.com/docs/api/reference/rest>

⁸ <https://developers.notion.com/>

⁹ <https://beta.openai.com/docs/>

¹⁰ <https://docs.microsoft.com/en-us/azure/bot-service/index-bf-sdk?view=azure-bot-service-4.0>

¹¹ <https://docs.microsoft.com/en-us/azure/bot-service/rest-api/bot-framework-rest-direct-line-3-0-api-reference?view=azure-bot-service-4.0>

Backend design:

The following diagram (Figure 4-2) shows on a conceptual level the workflow of the bot:

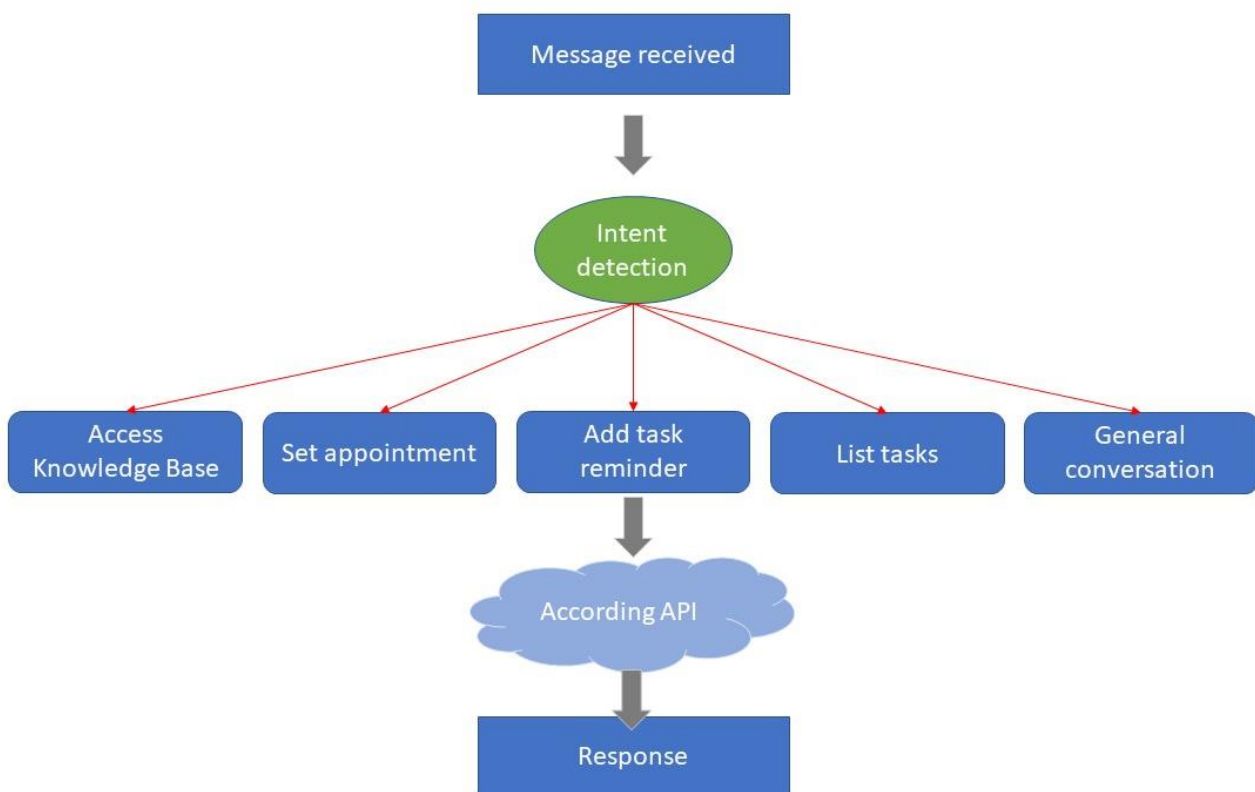


Figure 4-2 workflow of the bot- when the bot receives a message, it will detect the intent of the utterance and dispatch to the respective dialog. then it will make according API calls (if needed) and return a response

When the bot receives the message, it will detect the top intent (e.g. Set an appointment) of the utterance using an NLP model. It will then extract the relevant entities (e.g. Starting time of the meeting...). Now given the entities, it will call the respective API – if

needed- and send the response text back to the frontend.

5 Implementation

In this section, we will talk in detail about each technology implemented and the workflow of the bot.

Azure Bot Framework SDK Python.

The Bot Framework, along with the Azure Bot Service, provides tools to build, test, deploy, and manage intelligent bots, all in one place¹². The core of our project resides in this technology. At its simplest form, it consists of an `app.py` and a `bot.py` file.

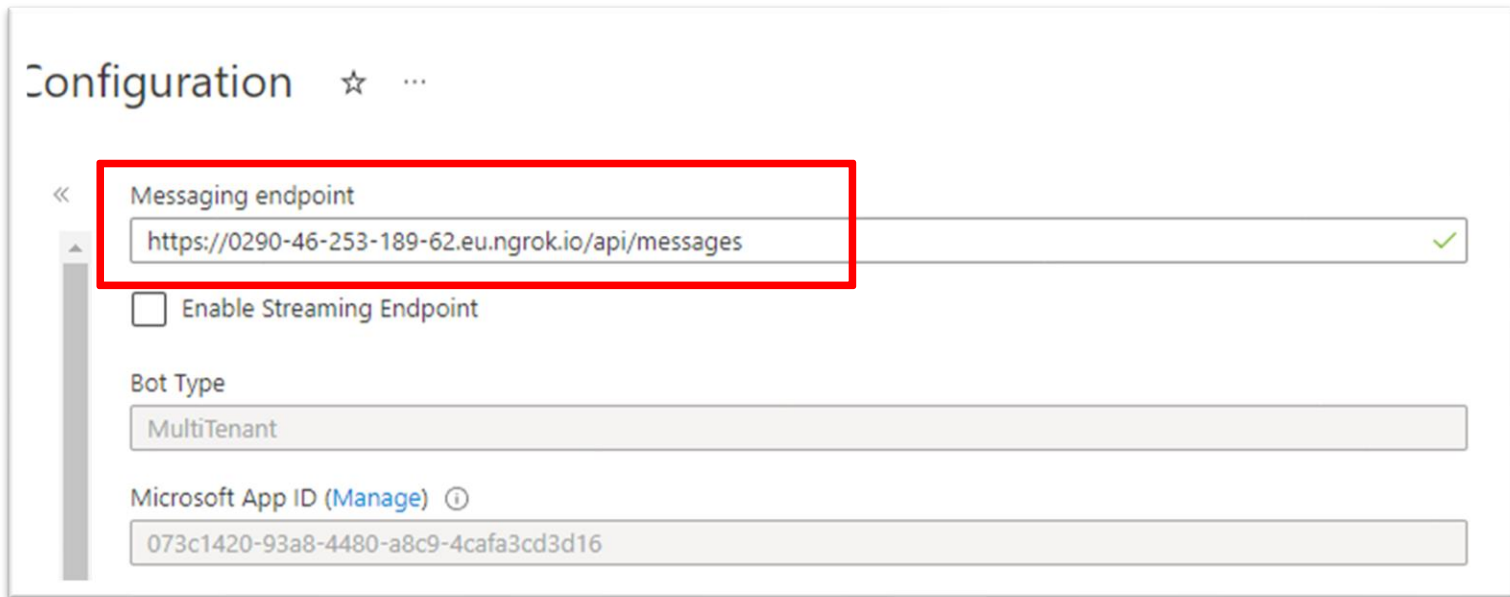
The `app.py` file handles the connection between Web App running locally on the PC and the deployed bot service on Azure cloud services. The `bot.py` file controls the flow of the messages.

Now it is worth noting that we could have made the bot run on an Azure WebApp service (instead of running it locally) but this was challenging since the framework is under development and our skills regarding DevOps are limited).

Since the `app.py` is running locally, we needed a way to make it communicate with the deployed bot; Hence, we used Ngrok to tunnel using a temporary domain. The only drawback is that the domain expires every 2 hours.

During normal functioning of the bot, we will connect Ngrok to the local port 3978 and we set the bot endpoint (of the deployed Bot Service) to the valid Ngrok domain.

¹² <https://docs.microsoft.com/en-us/azure/bot-service/index-bf-sdk?view=azure-bot-service-4.0>



Configuration ☆ ...

« Messaging endpoint

https://0290-46-253-189-62.eu.ngrok.io/api/messages ✓

☐ Enable Streaming Endpoint

Bot Type

MultiTenant

Microsoft App ID (Manage) ⓘ

073c1420-93a8-4480-a8c9-4cafa3cd3d16

Figure 5-1 Bot registration

Note that the Microsoft App ID is also needed to connect the app online. The process of deploying a bot is well detailed in the Bot Framework SDK documentation¹³

The documentation also provides multiple examples, one of which heavily inspired our project. You can check the link below to look at the following example (NLP-With-Dispatch¹⁴). In this report we will explain briefly this architecture and we will talk about the modification we did to suit our case.

As soon as the bot receives a message the *on_members_added_activity* function will be called and the bot will start the *main_dialog.py*. there's a *dialog_helper* class that will control the flow of the dialog.

The *MainDialog* class inherits from *ComponentDialog* class that allows it to add *WaterfallDialog*. It consists of multiple prompt arranged in steps and called on after the other. To be completely clear, the dispatch technique is deprecated¹⁵, but we tried to implemented in our way. The *MainDialog* class will dispatch based on the intent.

When the *MainDialog* is running, in the *self.intent_detection_step* step, we use LUIS Azure service to detect the intent and extract the entities. In order to use the service, you need to register and create a resource group. More on the LUIS service later, but for now you need to know that we used this service to detect the intent of the incoming message; Now using the *LuisHelper* Class we are able to distinct either the user is intending to either:

¹³ <https://docs.microsoft.com/en-us/azure/bot-service/bot-service-quickstart-create-bot?view=azure-bot-service-4.0&tabs=python%2Cvs>

¹⁴ <https://docs.microsoft.com/en-us/azure/bot-service/bot-builder-howto-qna?view=azure-bot-service-4.0&tabs=python>

¹⁵ <https://docs.microsoft.com/en-us/azure/bot-service/bot-builder-tutorial-dispatch?view=azure-bot-service-4.0&tabs=cs>

- set a meeting (CALENDARCREATECALENDARENTRY)
- show the ToDo list (TODOSHOWTODO)
- add a reminder (TODOADDTODO)
- None of the above, hence dispatching to the QnA service (knowledge base) or the GPT3 API

Now that we detected the intent and extracted the entities we will dispatch to the according dialog. The dispatch is done using the following function

```
step_context.begin_dialog ({Dialog_Name}, options=luis_result)
```

This will start the according dialog (Dialog_Name) with the according extracted entities (luis_results).

Let's briefly talk about each dialog.

Set Meeting Dialog

Set_meeting_dialog.py is the most complex dialog since it is initiated with 3 entities (start_date, start_time, end_time) and 7 steps.

Notice that not all the steps will be called; some steps will be called only if we have a missing entity. See figure 5-2.

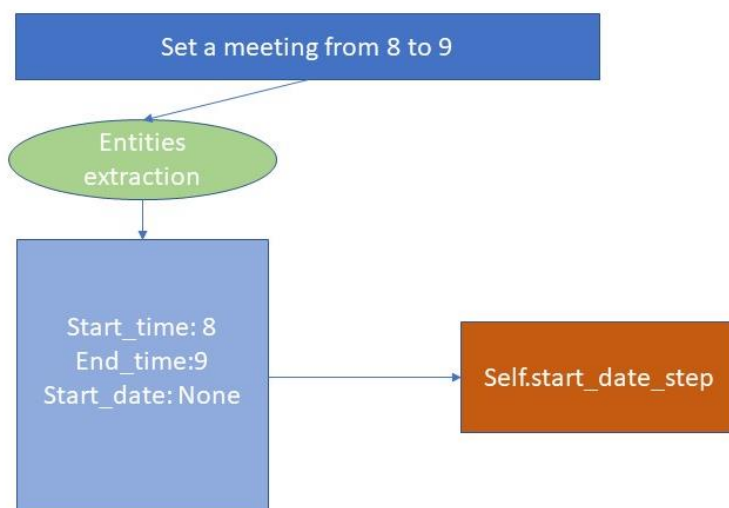


Figure 5-2- set meeting dialog; if needed, the bot will prompt the user to provide the missing entity

In this case the user provided the starting time, the ending time but no date was given. So only this step will be called. Notice that each step has a validation function that, if the response was not valid, will restart the step and send a re-prompt message.

Figure 5-3 shows all the steps in detail.

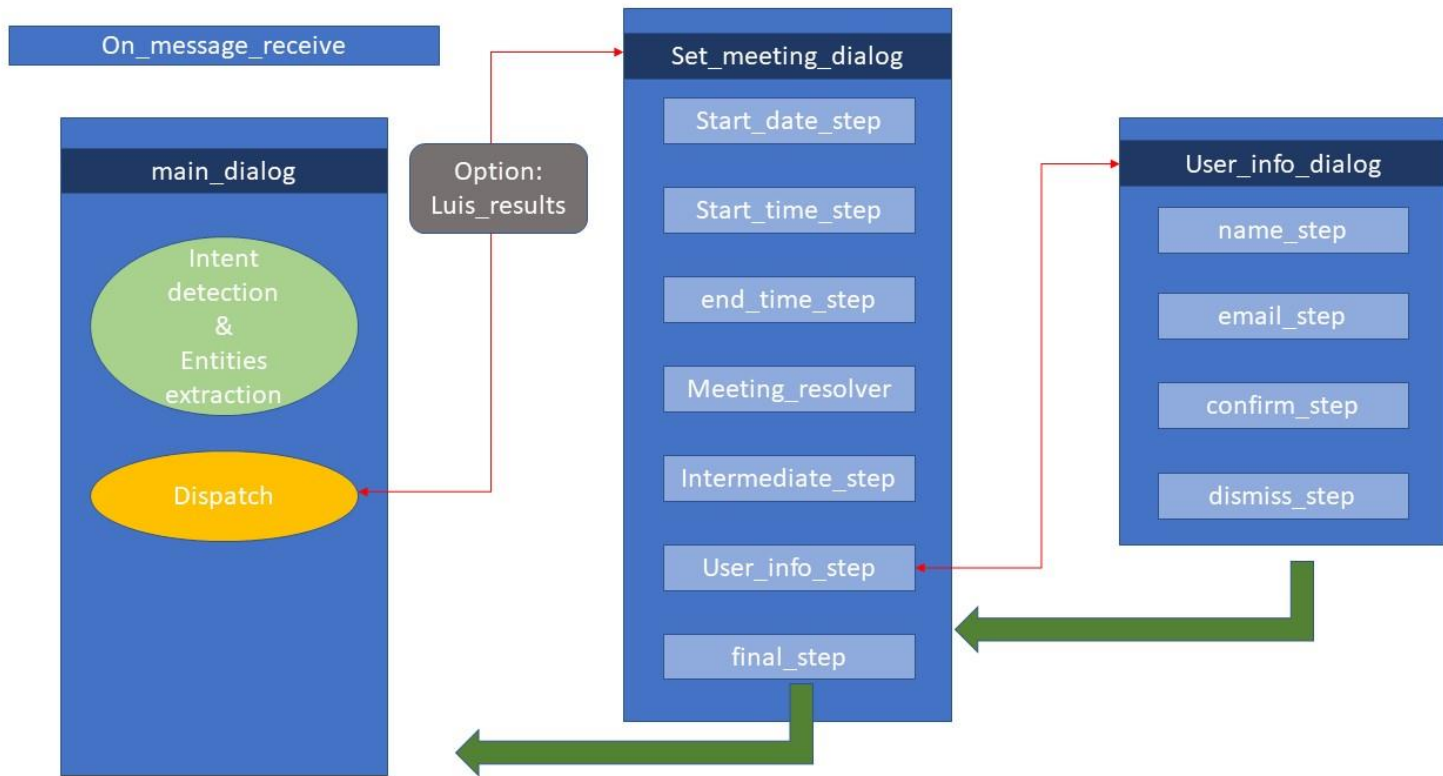


Figure 5-3- *set_meeting_dialog*; when the bot detect "ADD CALENDAR ENTRY" intent, the *set_meeting_dialog* will be initiated. the Luis results along with the extracted entities will be passed as option (parameter). it will go through the steps if needed.

When the LUIS recognizer, detect the *add_calendar_entry* intent, it will dispatch to the *set_meeting_dialog.py* dialog. Now at each step if the entity isn't detected it will prompt the user to provide it.; otherwise it will skip the step.

Now it is worth mentioning that the entity extracting on the dialog level is done by the Azure LUIS recognizer (*luis_helper.py*); but on the step level, the extraction is done by another nlp model, Spacy.

Spacy is an open-source software python library used in advanced natural language processing and machine learning. It will be used to build information extraction, natural language understanding systems, and to pre-process text for deep learning¹⁶

At the meeting resolver step, we will check the calendar availability regarding the given date and time. The *TimexResolutionHelper* class will convert the above entities into Timex format

¹⁶ <https://spacy.io/api/doc>

according to the **ISO**-8601 standard so it can be used over multiple APIs without any issue.

In addition to that, it will apply constraints (using the *apply_constraint* function) to be sure that the meeting is during working hours and within a week of the current day. In addition to that, it will call the google calendar API to see if there is no other event in this exact same time (using *check_google_calendar_availability* function).

If there is nothing else scheduled, it will proceed to collect user information through *user_info_dialog.py*.

The bot will prompt the user to enter his name and his email. It will also validate his email and check if it is deliverable using **AbstractAPI**¹⁷.

Now that the bot has all the information, it will make an API call to book the meeting on the calendar and send notification to both the owner and the user.

Then it will fall back to the main dialog.

ToDo Add ToDo Dialog

Similarly, when initiated, this dialog needs the *luis_result* option that hold all the entities needed.

Actually, there is only one entity in this case, which is the tasks title. If not detected by the LUIS the bot will prompt the user to add it otherwise, it will call the **NotionAPI** and add the task to the workspace under ToDo status.

Cancel And Help Dialog

This is not an actual dialog, but it actually implements the interrupt logic to the bot; so, at any time the user can cancel an ongoing dialog. More one interrupt in the following link¹⁸.

¹⁷ <https://www.abstractapi.com/>

¹⁸ <https://docs.microsoft.com/en-us/azure/bot-service/bot-builder-howto-handle-user-interrupt?view=azure-bot-service-4.0&tabs=csharp>

LUIS and QnA Azure services

In order to use any azure service, you need to register and create a resource group. Being a student, you get \$100 in credits. More on how to create an Azure Service in the following link¹⁹.

Once you create a service, you will be given an endpoint URL and a key that you should copy and paste in the corresponding place in the *config.py* file.

LUIS service

The azure LUIS service comes with prebuilt domain that can be used for intent detection. A look at the domain we used (table 3):

Table 3- Pre-built domains

Domain	Intent	Entities
Calendar	CreateCalendarEntry	start_time, end_time, start_date
ToDo	AddToDo	Content
ToDo	ShowToDo	None

QnA Service

Using the QnA will allow the bot to respond to general information of the owner. Technically speaking it's all the static information. For more information follow this link²⁰

Direct Line API

You can enable communication between your bot and your own client application by using the Direct Line API. This article introduces key concepts in Direct Line API 3.0 and provides information about relevant developer resources. You can build a

¹⁹ <https://azure.microsoft.com/en-us/products/>

²⁰ <https://docs.microsoft.com/en-us/azure/bot-service/bot-builder-howto-qna?view=azure-bot-service-4.0&tabs=cs>

client using the SDK, REST API, or Web Chat²¹.

While communicating with a client app, it needs some hints to know when the bot is expecting answer or any other state.

The bot will feed the expected state via the *Input Hint* hyper parameter.

The state diagram below (Figure 5-4) shows the functioning of the bot.

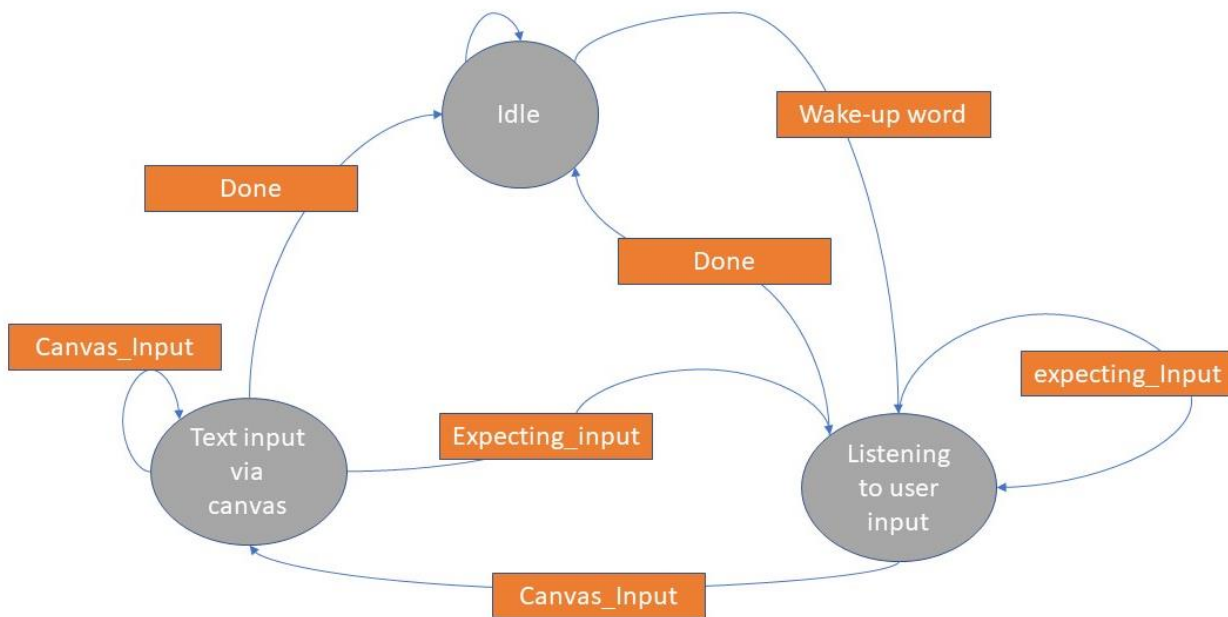


Figure 5-4 state machine representing the workflow of the client app

State: IDLE

During the IDLE state, the bot is waiting for the wake-up word, porcupine.

State: Listening to the user input

During this state, the bot is expecting the user to enter prompts. In this state the bot will be applying STT (speech to text) technologies. When ready, the bot will convert the generated response to voice format using TTS (text-to-speech).

State: Text input via canvas

Here the bot is expecting a text input. This is useful when an email address is required. There is no STT technology capable of doing this task.

²¹ <https://docs.microsoft.com/en-us/azure/bot-service/rest-api/bot-framework-rest-direct-line-3-0-api-reference?view=azure-bot-service-4.0>

Complete Design

Figure 5-5 shows a more complete diagram of the design.

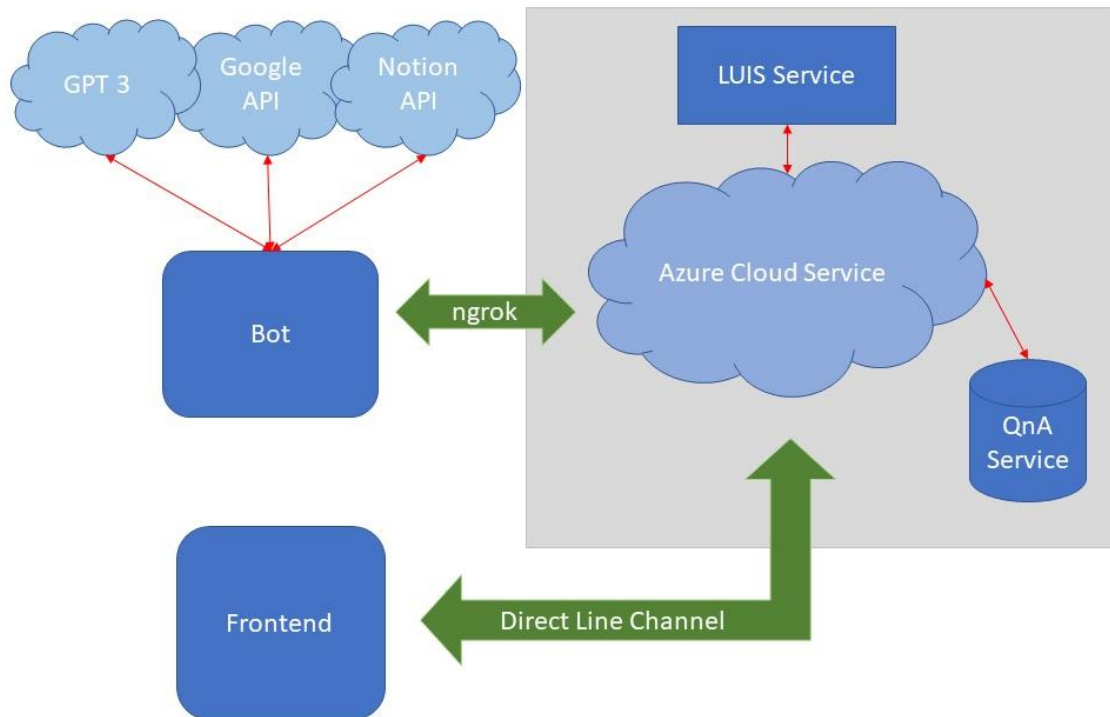


Figure 5-5 Complete architecture of the assistant

6 Evaluation/Validation/Results

Now the bot is able, to set a meeting on Google Calendar, show the owner's ToDo list, remind the owner of a given task via Notion API and answer general question using GPT3 API.

Set a meeting on Google Calendar

First, let's take a deeper look at the following scenario. The user needs to set a meeting with the owner. See Figure 6-1.

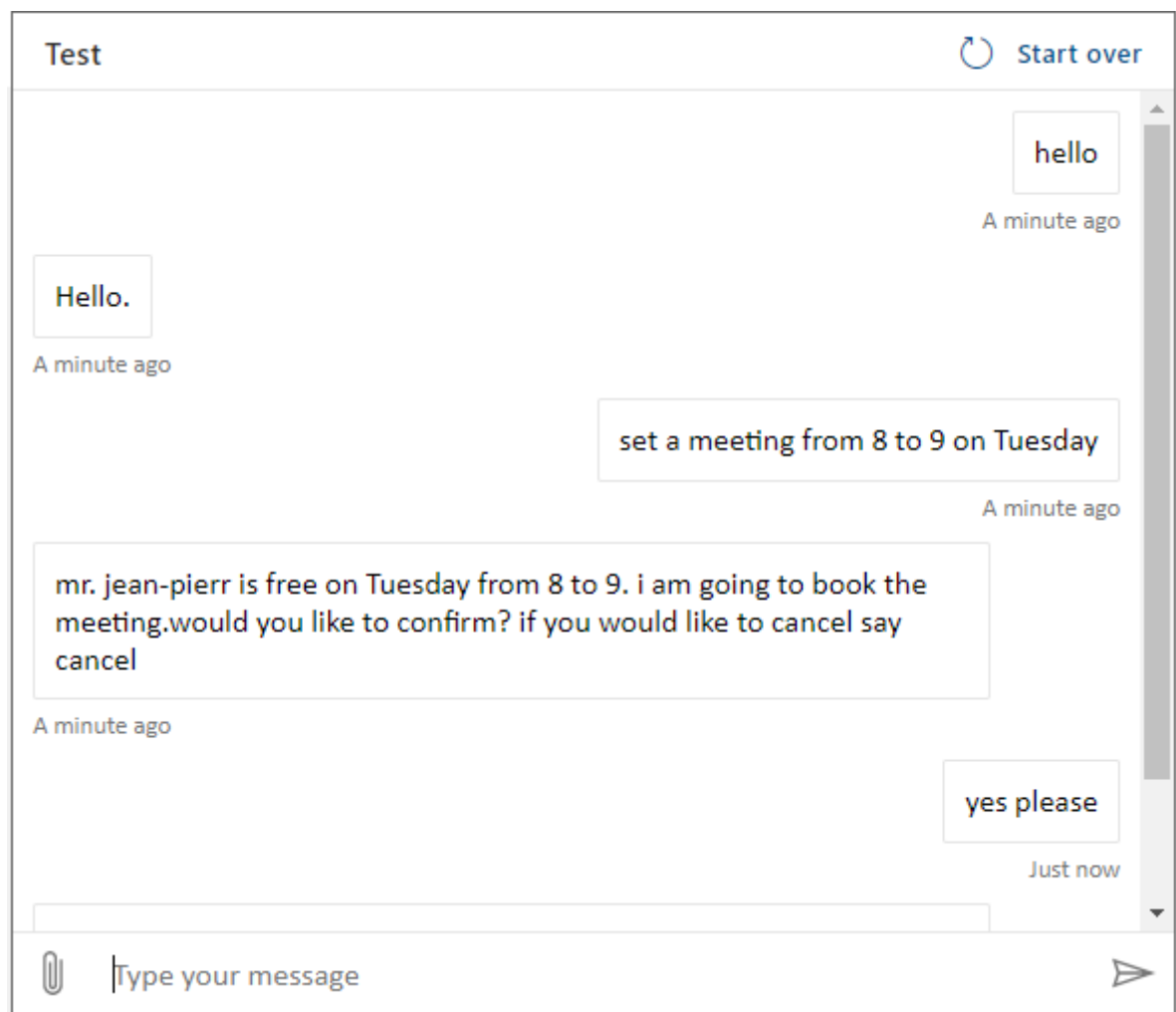


Figure 6-1 Setting a meeting scenario given that all the entities are provided

The bot will check if the owner is free at this time of the day; in this case the bot will check the availability from 8 o'clock to 9 o'clock on Tuesday. And since no other event is set on the same time the bot will proceed with the procedure and ask the user for additional information.

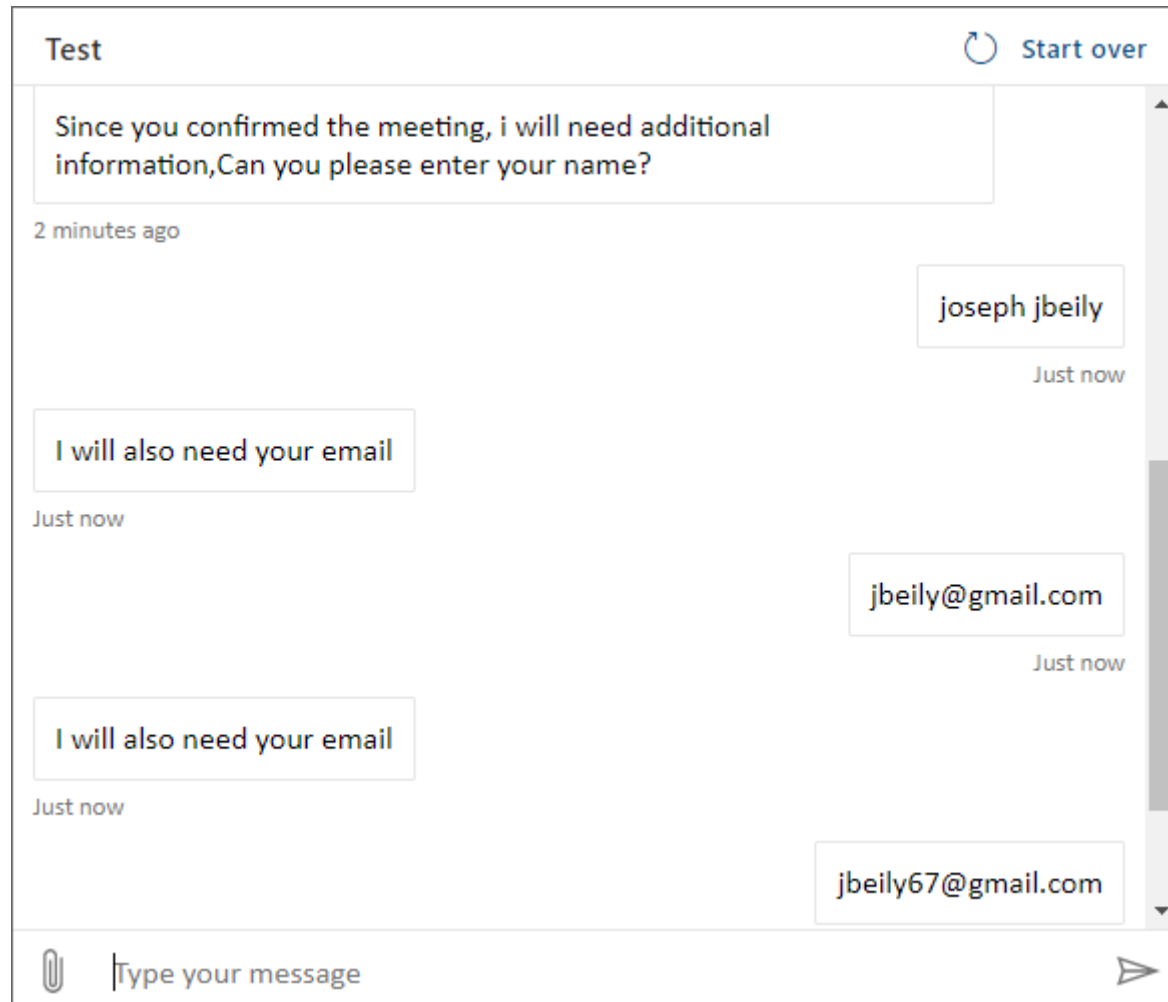


Figure 6-2 Collecting user information

Now the bot will ask for the name and the email of the user. The bot will verify the deliverability of the email via the Abstract API. It will keep asking the user for the email until the validation is complete. The user can cancel any time if desired.

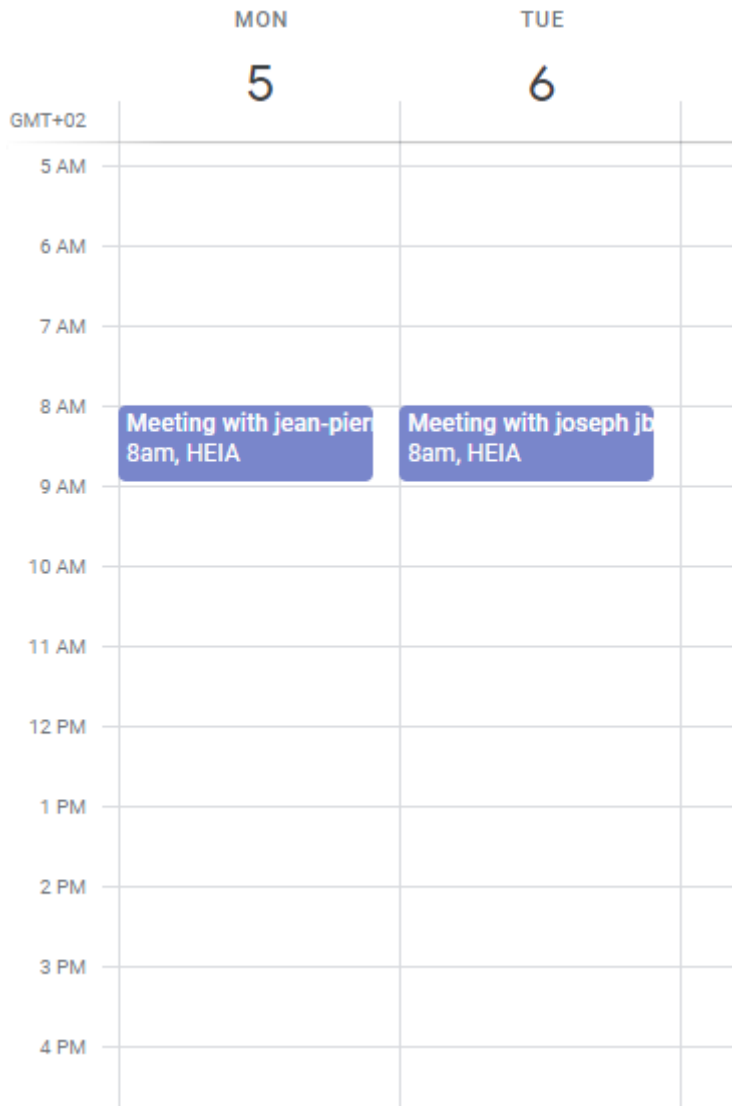


Figure 6-3 adding the event on Google Calendar

Once the user provides all his information, the event will be visible on the google calendar. Both the owner and the user will receive notification 15min prior to the set starting time.

The user information will also be saved in NoSQL Database (Cosmos DB). For this project, the database is implemented using an emulator. See Figure 6-4.

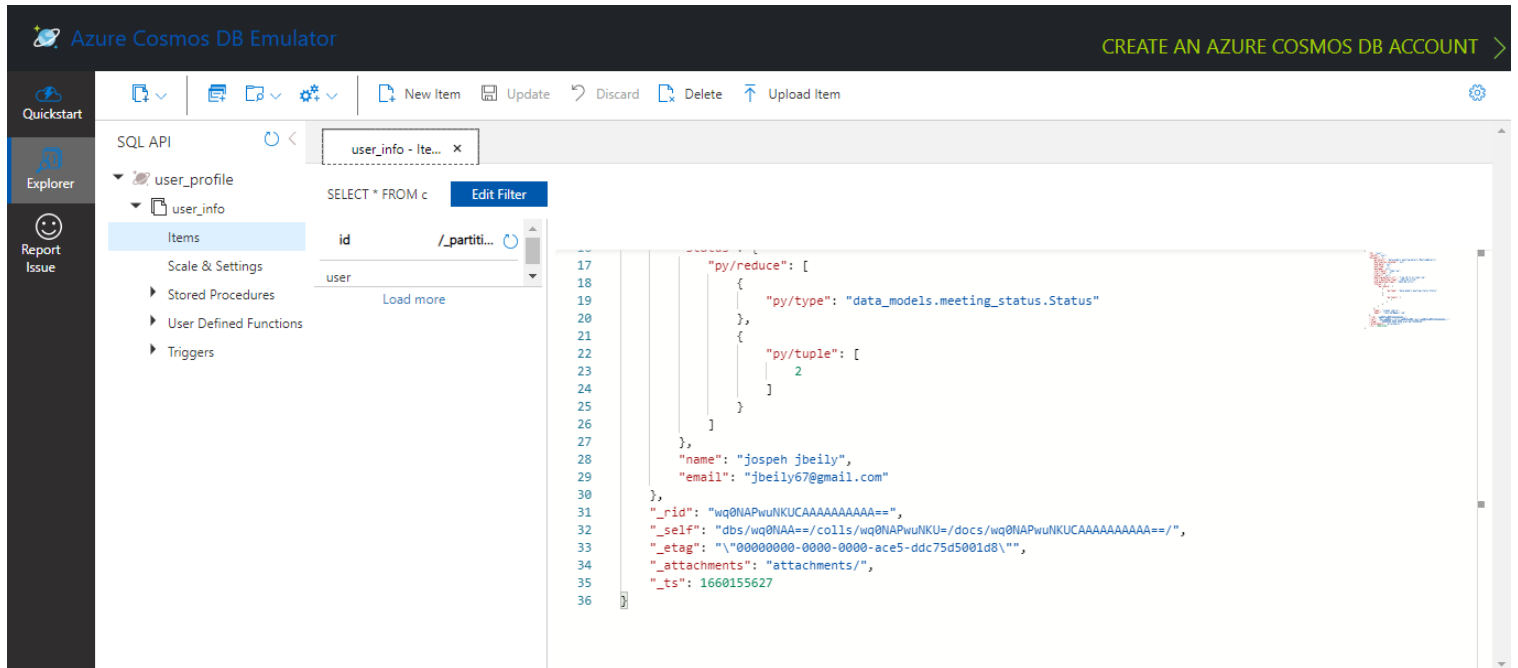


Figure 6-4 Saving the user info in Cosmos DB

In case the user doesn't provide the sufficient information to set a meeting, or the LUIS isn't able to extract all the entities, the bot will ask for each missing entity individually. The following example shows the worst-case scenario. See figure 6-5.

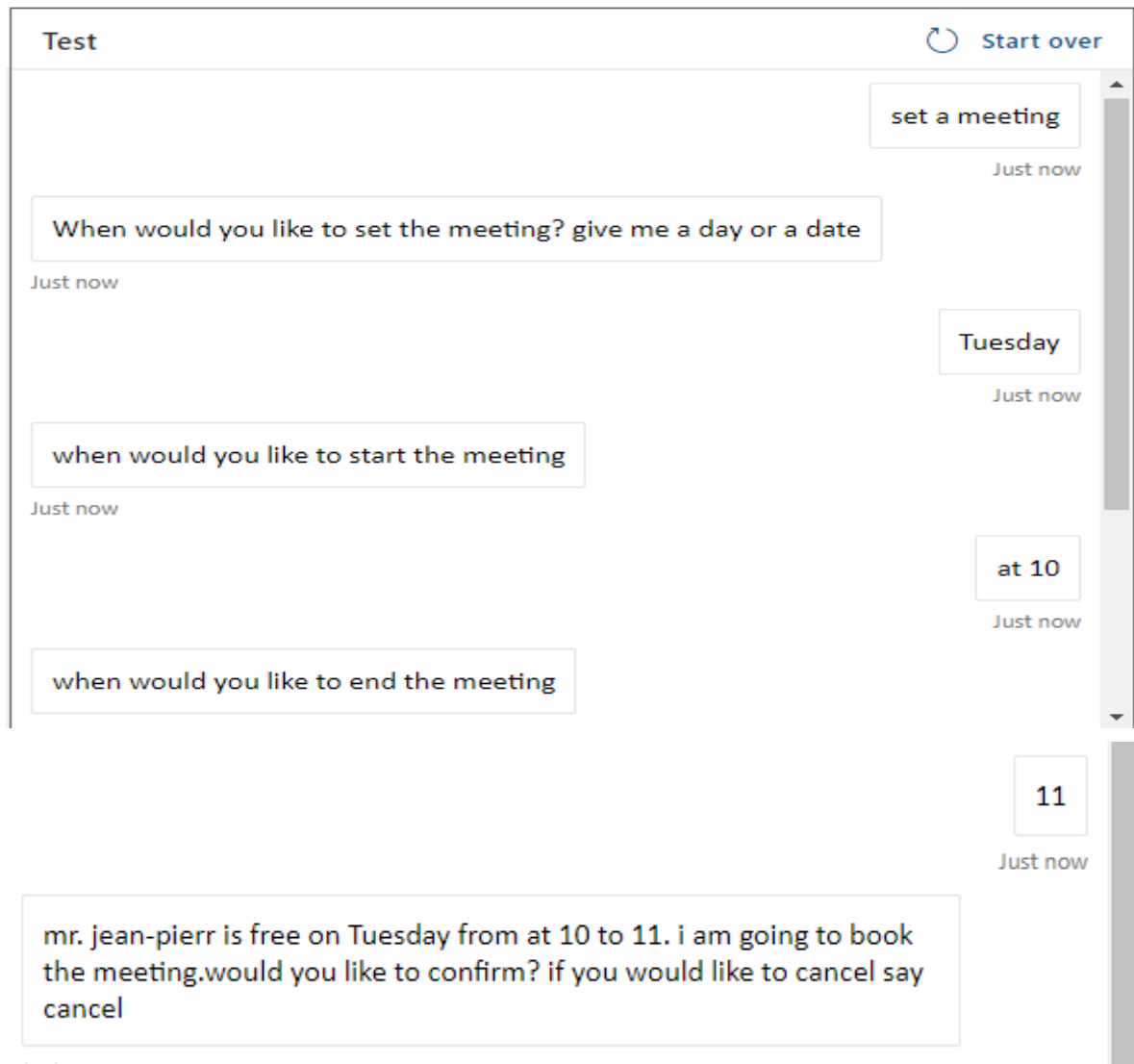


Figure 6-5 Setting a meeting with no entities given (worst-case Scenario)

Here the intent is detected correctly but no entity was extracted. So, the bot prompts the user for the required entities

Now if the owner isn't available at a given time, the bot will propose a different starting time the same day. See figure 6-6.

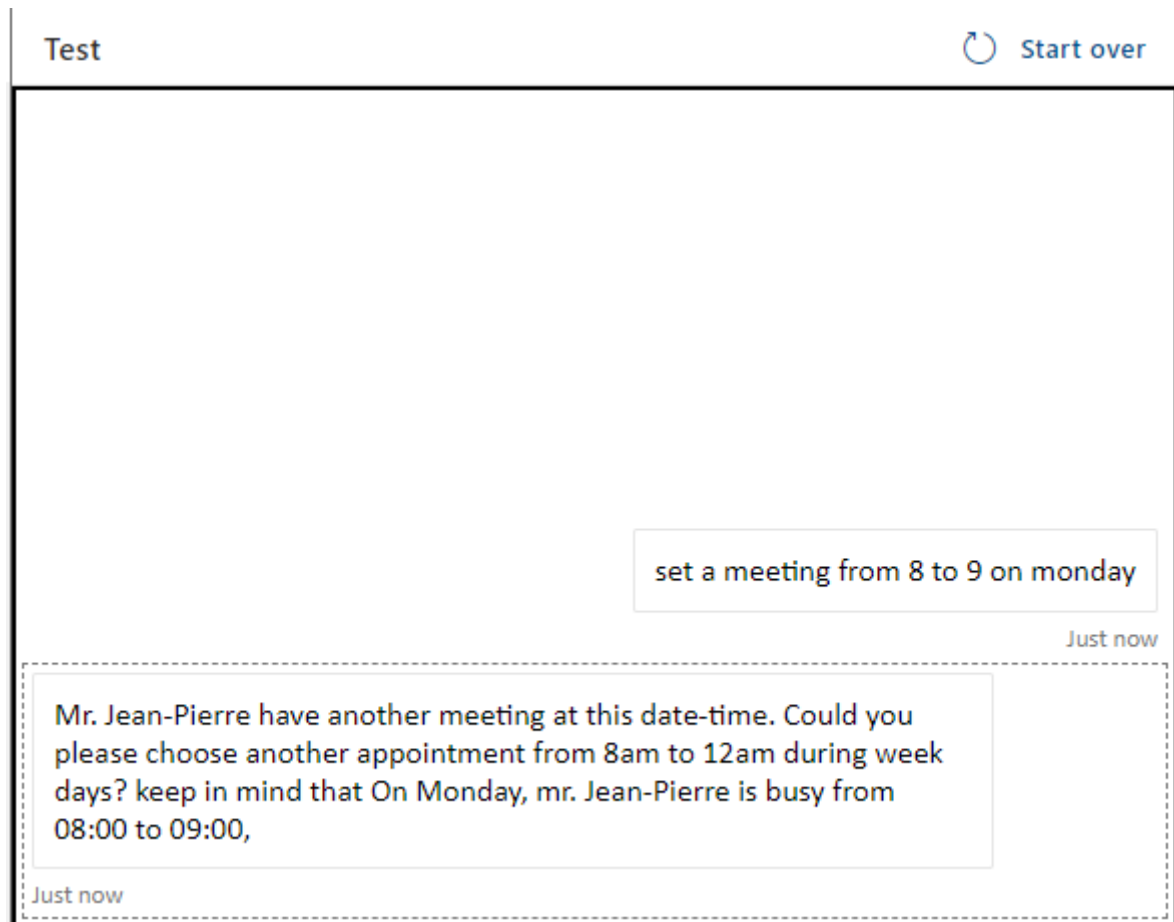


Figure 6-6 Setting a meeting when the owner is busy

The user can also ask the bot when is the owner available. If the user asks for a specific day, the bot will list the owner's schedule on this given day. By default, the bot will check the schedule of the current day. Additional development is needed in this case. (Figure 6-7)

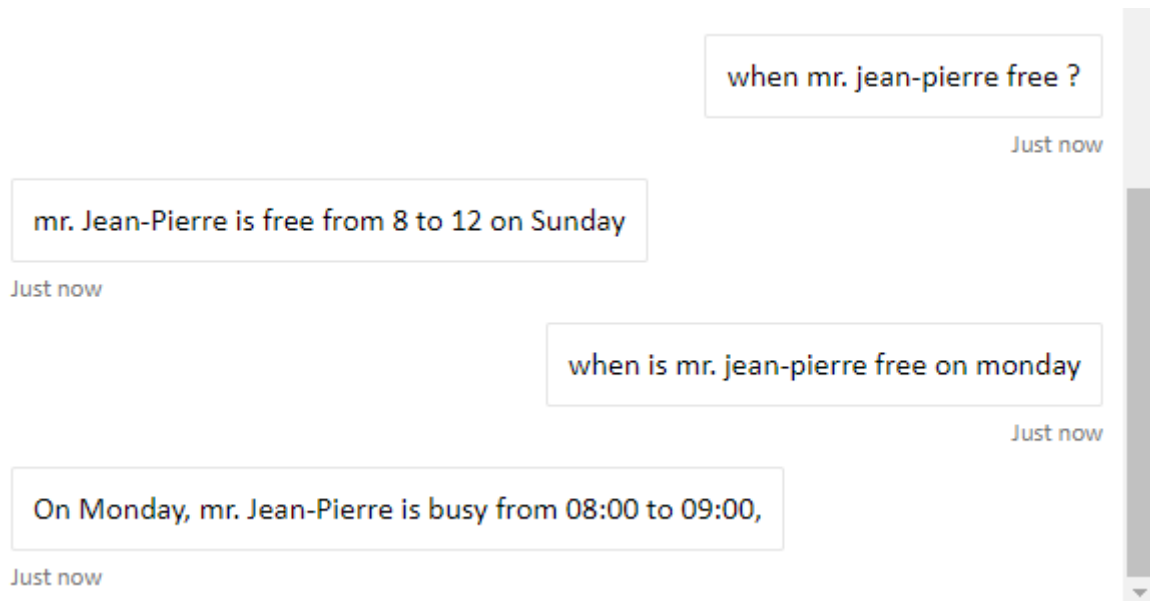


Figure 6-7 Checking availability

The bot can also delete a meeting with a given person. In case there is multiple meeting with the same person in the span of the current week it will delete the first meeting. For future implementation, we could extract not only the Person name, but also the starting date of the meeting. In this case we could delete a meeting with a specific person on a specific day.

Note that in the following example (Figure 6-8), if the name of the person is not provided, the bot will prompt the user to provide the name.

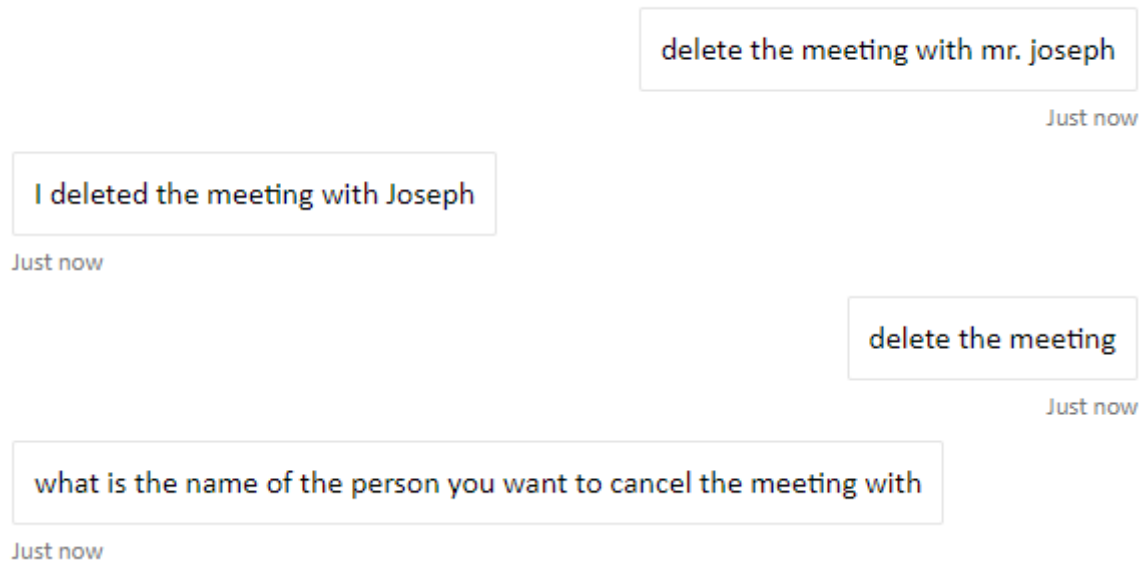


Figure 6-8 Deleting a meeting

Remind the owner of a Task on Notion and query Task List

The bot should be able to keep the user up-to-date on the progress of a project. We used Notion API to provide the user with the To-do list of the owner.

Take a look at a task list on Notion API in the Figure 6-9.

✓ Task List

Board View +

Filter Sort 🔍 ... New ▾

+ Add filter Reset Save for everyone ▾

To Do 2

To call the doctor

To Do

August 18, 2022 5:37 AM

Write the reports

To Do

August 31, 2022 1:18 AM

+ New

Doing 1

Take Fig on a walk

Doing

August 16, 2022 5:08 AM

+ New

Done 🏆 1

fix DateTime resolver dialog

Done 🏆

August 18, 2022 6:19 AM

+ New

+ Hidden groups

📁 No Status 2

Figure 6-9 Notion Task List

Now if prompted the bot will list the to-do section of this page.

what have you been up to

Just now

I will list the To DO list of Mr. Jean-Pierre: Write the reports, To call the doctor

Just now

Figure 6-10 Querying a Notion Task List

The bot can also remind the user of a certain task. See figure 6-11.

A minute ago

remind mr. jean-pierre to write the pvs

Just now

I will remind Mr. Jean-Pierre to Write the pvs. is it ok?

Just now

ok

Just now

That's great.

Just now

Figure 6-11 Adding a task

Now a new task titled “Write the PVs” will be added to the Notion Task List. See Figure 6-12.

To Do 3

To call the doctor

To Do

August 18, 2022 5:37 AM

Write the pvs

To Do

September 4, 2022 2:29 AM

Write the reports

To Do

August 31, 2022 1:18 AM

+ New

Figure 6-12 Adding a Task under to-do list

Query a knowledge base and respond to general information

The bot can also answer about personal information about the user and itself. It will query the knowledge base (present in the QnA maker). (Figure 6-13)

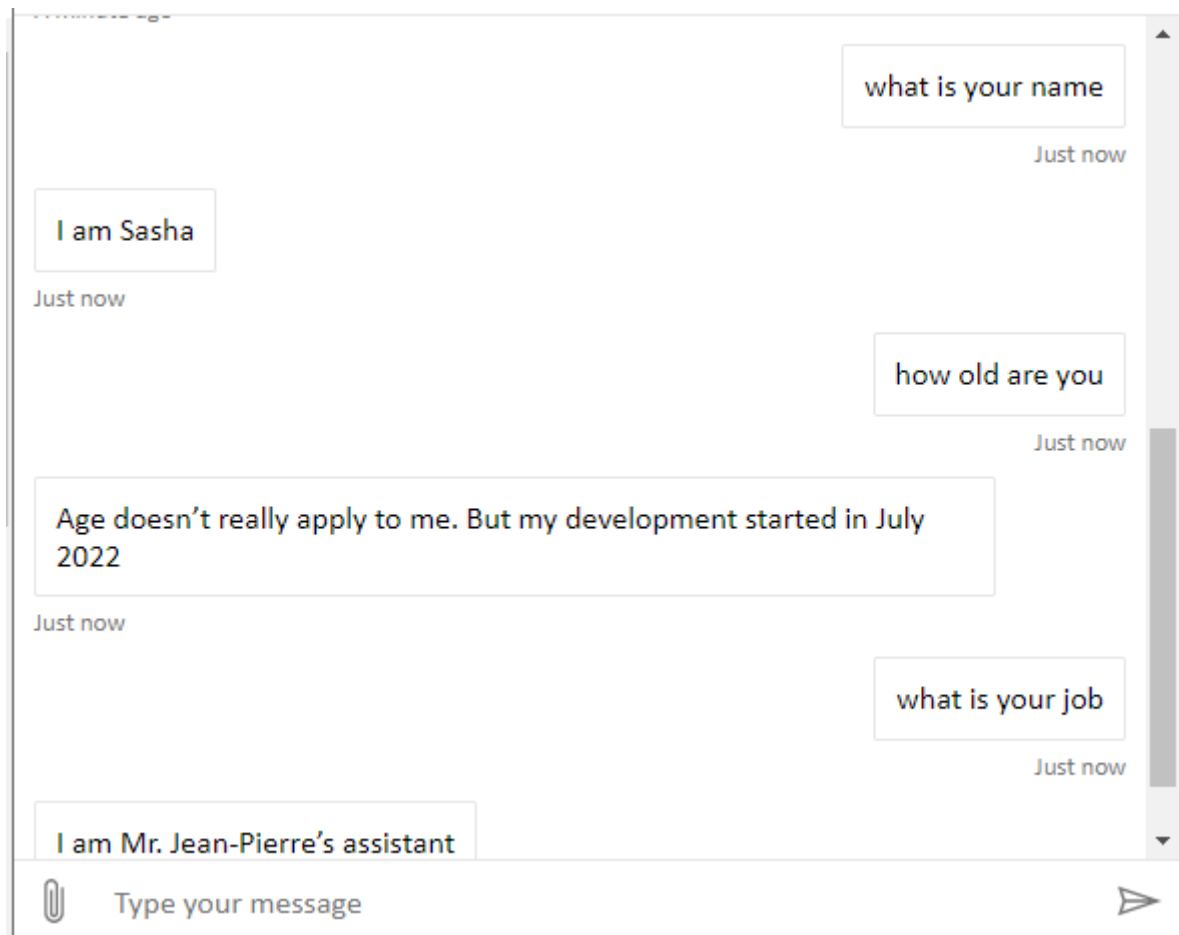


Figure 6-13 Querying the Knowledge Base

If the information is not found in the knowledge base, it will call the GPT3 API. See figure 6-14.

what is global warming

A minute ago

Global warming is a gradual increase in the Earth's average surface temperature, typically caused by the emission of greenhouse gases such as carbon dioxide and methane.

The term "global warming" can also refer to the greenhouse effect.

Just now

how old is the sun

Just now

?

The sun is approximately 4.5 billion years old.

Just now

Figure 6-14 Calling GTP3 API

7 Conclusion

7.1 Interpretation of results

We are confident to say that, if the owner needed to attend a meeting just to talk about his work progress so far or to set another meeting, the bot could easily replace him. This will increase the owner's work efficiency and allow for additional free time

7.2 Limitations and Further work

One limitation, is when the bot needs to prompt the user for additional information about the meeting, it uses the Timex resolution NLP model. In some cases, it will extract entities properly but sometimes, it will fail to extract those entities and keeps re-prompting the user for information. This could be solved by using the orchestrator instead of the dispatch. The orchestrator will allow to use the LUIS model over all the dialogs. The problem is that the orchestrator is only available in C# and JavaScript.

Later, we could make the bot prompt for available free time on the schedule when the user doesn't ask for a specific starting date and starting time

7.3 Personal conclusion

During this project, I enhanced my knowledge in python programming. I acquired skills that would be useful for future work.

8 Glossary

STT Speech-to-text

TTS Text-to-speech

LUIS Language Understanding Service

QnA Questions and Answers

References

- ¹ <https://docs.microsoft.com/en-us/azure/bot-service/?view=azure-bot-service-4.0>
- ² <https://rasa.com/docs>
- ³ <https://cloud.google.com/dialogflow/docs>
- ⁴ <https://docs.readyplayer.me/ready-player-me/>
- ⁵ <https://docs.microsoft.com/en-us/azure/cognitive-services/luis/what-is-luis>
- ⁶ <https://docs.microsoft.com/en-us/azure/cognitive-services/qnamaker/overview/overview>
- ⁷ <https://developers.google.com/docs/api/reference/rest>
- ⁸ <https://developers.notion.com/>
- ⁹ <https://beta.openai.com/docs/>
- ¹⁰ <https://docs.microsoft.com/en-us/azure/bot-service/index-bf-sdk?view=azure-bot-service-4.0>
- ¹¹ <https://docs.microsoft.com/en-us/azure/bot-service/rest-api/bot-framework-rest-direct-line-3-0-api-reference?view=azure-bot-service-4.0>
- ¹² <https://docs.microsoft.com/en-us/azure/bot-service/index-bf-sdk?view=azure-bot-service-4.0>
- ¹³ <https://docs.microsoft.com/en-us/azure/bot-service/bot-service-quickstart-create-bot?view=azure-bot-service-4.0&tabs=python%2Cvs>
- ¹⁴ <https://docs.microsoft.com/en-us/azure/bot-service/bot-builder-howto-qna?view=azure-bot-service-4.0&tabs=python>
- ¹⁵ <https://docs.microsoft.com/en-us/azure/bot-service/bot-builder-tutorial-dispatch?view=azure-bot-service-4.0&tabs=cs>
- ¹⁶ <https://spacy.io/api/doc>
- ¹⁷ <https://www.abstractapi.com/>
- ¹⁸ <https://docs.microsoft.com/en-us/azure/bot-service/bot-builder-howto-handle-user-interrupt?view=azure-bot-service-4.0&tabs=csharp>
- ¹⁹ <https://azure.microsoft.com/en-us/products/>
- ²⁰ <https://docs.microsoft.com/en-us/azure/bot-service/bot-builder-howto-qna?view=azure-bot-service-4.0&tabs=cs>
- ²¹ <https://docs.microsoft.com/en-us/azure/bot-service/rest-api/bot-framework-rest-direct-line-3-0-api-reference?view=azure-bot-service-4.0>