Chatbot for a fictive company specialized in health data

During this project, I wanted to design a chatbot that would answer a user's questions about a fictive company's products. This was a very interesting project, as it covered the development of a product from start to finish, in this case deciding and writing the specs, creating the chatbot, developing the backend and then building a lightweight frontend.

This project has been inspired from codebasics (https://codebasics.io/), who built a chatbot for food delivery.

You can directly access to the chatbot following this link: https://clement1103.github.io/frontend/, but I would strongly advise you to read the whole document.

1. The fictive company

First, it was necessary to define the context in which the chatbot would be employed. It would be used by a fictitive company specializing in healthcare AI, offering three flagship services:

MedIAg

An AI tool that analyzes medical data to assist doctors in making accurate diagnoses while improving over time

SyntheMedix

A tool which generates synthetic healthcare data that mimics real data patterns to protect patient privacy and support research.

ForecastMed

A predictive analytics tool that helps healthcare providers anticipate patient outcomes and optimize care strategies using medical data and machine learning.

Logos generated by Dall-E

These products would be proposed to other companies specialized in AI for health, or medical institutions in order to improve their efficiency.

Also, the company would offer custom services that meet the specific needs of other collaborators.

2. Definition of the specs

It was also necessary to clearly define the requirements that the chatbot would have to meet. It's obvious that I couldn't develop a chatbot that could understand several languages, that would have a very wide range of answers to suit all the questions it was asked.

After analyzing and prioritizing the initial requirements, I decided that the chatbot would have to meet the following criteria:
- Present the different services available,
- Capture the user's interest following each presentation,
- Obtain and confirm user information so that an expert can contact them again,
- Redirect the user to an expert if a complex question is asked
- Schedule a meeting with an expert.

Here are 3 simplified conversation flows that the chatbot should be able to handle:

①
BOT: Welcome. Would you like a presentation of the services or to schedule a meeting?
USER: Presentation.
BOT: Here are our products […] Do you want to know more about one?
USER: Tell me more about SyntheMedix.
BOT: [Presentation of the product] Are you interested?
USER: Yes.
BOT: Do you want to discover another service ?
USER: Yes, tell me more about ForecastMed.
BOT: [Presentation of the product] Are you interested?
USER: No.
BOT: Do you want to discover another service ?
USER: No.
BOT: What are your email/phone number ?
USER: contact1@example.com, 0123456789
BOT: Are these information correct?
USER: Yes
BOT: Someone will contact you.

②
BOT: Welcome. Would you like a presentation of the services or to schedule a meeting?
USER: Schedule a meeting.
BOT: What are your email/phone number ?
USER: contact1@example.com, 0123456789
BOT: Are these information correct?
USER: Yes.
BOT: Someone will contact you.

③
…
BOT: [Presentation of the product] Are you interested?
USER: How precisely do you think it would behave in this specific case?
BOT: I don't have the answer. Do you want an expert to get in touch with you to answer this specific question?
USER: Yes please.
BOT: What are your email/phone number ?
USER: contact1@example.com, 0123456789
BOT: Are these information correct?
USER: Yes.
BOT: Someone will contact you.

Then, the useful information from the chat (user coordinates, products of interest, date of the chat) would be extracted and host in a SQL database.

Finally, a lightweight frontend will be developped in order to complete the production launch of the chatbot.

3. Chatbot creation using dialogflow

During this project, I will use Dialogflow which is a NLP platform developed by Google, allowing developers to create conversational interfaces such as chatbots and voice assistants. It makes it easy to understand user intentions and manage dialogues, while integrating machine learning features to improve interactions over time.

To understand the development of the chatbot, there are some key concepts that need to be presented.

- Contexts: Used to manage conversation status and help keep track of the information needed to respond appropriately, enabling more natural and continuous dialogues.
  In our case, here are what the contexts could be like (we'll see later that it has slightly been modified):

Welcome context → Ongoing presentation → Capure of interest → Get coordinates
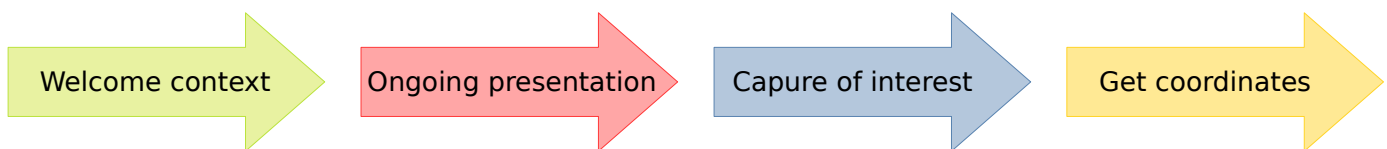
- Entities (Named Entity Recognition): Extract specific information from user input, such as names, dates, places or other relevant terms. For example, in the sentence "Book a table for two people in Paris", "two people" and "Paris" would be entities. One can also define custom entities, specific to the use case. This is what I did to define the services:

product_name   SAVE ⋮

☑ Define synonyms ❓   ☐ Regexp entity ❓   ☐ Allow automated expansion   ☑ Fuzzy matching ❓

ForecastMed	ForecastMed, forecastmed, forecast assistant, forecast tool, forecastMed
SyntheMedix	SyntheMedix, synthemedix, generated data tool, syntheMedix, SYNTHEMEDIX
MedIAg	MedIAg, mediag, assisted diagnostic tool, med diagnostic tool, MedIag, MedIAG, medIAg
Click here to edit entry	

For each entity, we define a name, and its corresponding synonyms.
It allows the chatbot to identify the different services even if the entity is misspelled, called with another name or with lower/upper case.

forecastmed
forecastMed
ForecastMed
Forecast tool
Forecast assistant

synthemedix
syntheMedix
SyntheMedix
generated data tool
SYNTHEMEDIX

Assisted diagnostic tool
mediag
MedIAg
medIAg
MedIag
MedIAG
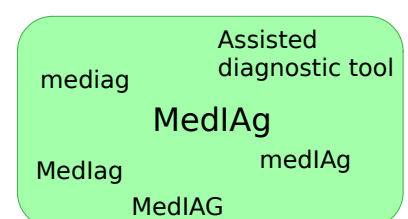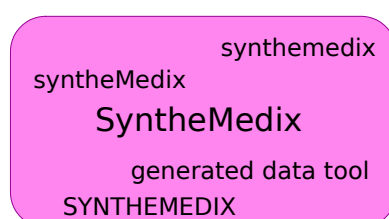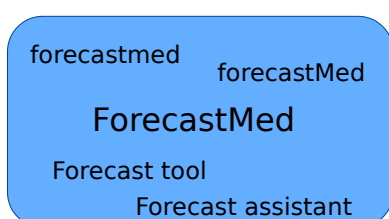
The entity recognition will also be useful when obtaining the email adress and the phone number of the user.

- Intents: Represent user intentions. Each intent is associated with example inputs that the user might say, helping Dialogflow to understand what the user wants to do. An intent is defined by training phrases and a response.

  The training phrases are sentences that the user might use, which will trigger the intent. For each intent, it is advisable to define several training phrases, in order to cover the full range of user response. However, the user phrases don't need to be exactly as the training ones, because Dialogflow uses NLP algorithms to detect sentences sharing the same meaning.
  Here is an example of training phrases used to trigger the intent aiming to present the services of the company:

Training phrases ❓   Search training phrases 🔍 ∧

⚠ Template phrases are deprecated and will be ignored in training time. More details here.

When a user says something similar to a training phrase, Dialogflow matches it to the intent. You don't have to create an exhaustive list. Dialogflow will fill out the list with similar expressions. To extract parameter values, use annotations with available system or custom entity types.

| " | Add user expression |

"	Yes sure, I'd love to hear about the products.
"	What services do you offer ?
"	Product presentation
"	What are the services of FictiveCompany ?
"	What are the products of FictiveCompany ?
"	What are the products ?
"	What do you offer ?
"	Present the products plz
"	Can you please present the services ?
"	I'd like to know more about the services

1 OF 2 →

With this set of training phrases, if the user type something like « I would like a presentation of the products your offer please. », the intent will be triggered even if this sentence is not in the training set.

Then, for each intent, we associate one response or more.

Text Response

1	The company offers several services:
	- MedIAg: a diagnostic assistant
	- SyntheMedix: a synthetic data generator
	- ForecastMed: a predictive analysis tool for health data
	Are you interested in one of them?
2	Enter a text response variant

Custom Payload

```json
{
  "richContent": [
    [
      {
        "text": [
          "We have three main solutions, covering several areas:",
          "- MedIAg, a diagnostic assistant",
          "- SyntheMedix, a synthetic data generation tool",
          "- ForecastMed, allowing predictive analytics for healthcare",
          "Which one are you interested in?"
        ],
        "type": "description"
      },
      {
        "options": [
          {
            "text": "MedIAg"
          },
          {
            "text": "SyntheMedix"
          },
          {
            "text": "ForecastMed"
          }
        ],
        "type": "chips"
      }
    ]
  ]
}
```

It can be a simple text response...

... or a more complex one, called custom payload, allowing to display different answer styles such as multiple choices.

To make the triggering of the desired intent easier, one can define an input and an output contexts during the creation of the intent.

• Fulfillment: Allows to execute actions outside Dialogflow, such as calling APIs or interacting with databases, to generate dynamic responses based on user input. This will be very useful for gathering and storing user information in our database. To link our chatbot with the backend, it is necessary to activate the fulfillment on the intent definition (but not every intent requires to access to the backend).

Fulfillment ❓

🔵 Enable webhook call for this intent

⚪ Enable webhook call for slot filling

• Fallback intents: Triggers when the chatbot can't categorize the user input into any intent.

Finally, here is the list of the intents I defined:

- capture.of.disinterest - context: ongoing presentation
- capture.of.interest - context: ongoing presentation
- check.coordinates
- complete.presentation - context: interest-capture
- coordinates.correct
- coordinates.incorrect
- Default Fallback Intent
- Default Welcome Intent
- fallback   Contexts: ongoing-presentation
- fallback2   Contexts: interest-capture
- no.appointment - context: fallback-presentation
- presentation.ForecastMed - context: interest-capture
- presentation.ForecastMed - context: ongoing-presentation
- presentation.MedIAg - context: interest-capture
- presentation.MedIAg - context: ongoing-presentation
- presentation.SyntheMedix - context: interest-capture
- presentation.SyntheMedix - context: ongoing-presentation
- product.presentation - choices
- set.appointment
- set.appointment - context: fallback-presentation

Sometimes, an intent is defined twice, for different contexts.

For instance, the intent « presentation SyntheMedix – context: ongoing-presentation » triggers at the beginning of the chat, when the user wants to know more about SyntheMedix after requesting a presentation of the different products, whereas « presentation SyntheMedix – context: interest-capture » triggers once the presentation of a product has already be done, the user has declared is (dis)interest in the product, and wants to discover another one.

4. Backend development

As mentioned above, the backend's role will be to enter the list of products that the customer is interested in, obtain and then verify the customer's contact details and add them to an SQL database.

To implement this, we need to create a server using fastapi and uvicorn, to manage the requests sent by the chatbot asynchronously. The server deployed by uvicorn is located at http://localhost:8000, but Dialogflow only accepts secure URLs. It was therefore necessary to use a tunnel that generates an https URL.

Then, to extract the relevant information from the Dialogflow responses, all I had to do is study their JSON syntax and filter out what's useful and what's not. Here is what a an extract of fulfillment request look like, with added comments:

```
{
  "responseId": "98eb2642-eea6-4f74-9495-5a2c60118907-47793ac9",
  "queryResult": {
    "queryText": "my phone number is 0123456789 and my email address is contact@example.com",
    "parameters": {
      "phone-number": "0123456789",
      "email": "contact@example.com" jj #Extraction of important features from the chat
    },
    ...
      "text": {
        "text": [
          "Are the following information correct: \nEmail address: contact@example.com\nPhone number: 0123456789"
        ]
      }
    }
  ],
  "outputContexts": [
    {
      "name": "projects/chatbot-health-wmdg/agent/sessions/214c489a-e9f7-bea0-5a90-e12e7c46e8c8/contexts/check-coordinates", #Extraction of the context, useful to handle multiple sessions
      "lifespanCount": 5,
      "parameters": {
        "email": "contact@example.com",
        "phone-number": "0123456789",
        "email.original": "contact@example.com",
        "phone-number.original": "0123456789"
      } ...
    ],
    "intent": {
      "name": "projects/chatbot-health-wmdg/agent/intents/d79dc514-8034-4fe9-a7f5-e554b53319bc",
      "displayName": "check.coordinates" #Extraction of the intent name
    },
    "intentDetectionConfidence": 1,
    "diagnosticInfo": {
      "webhook_latency_ms": 143
    },
    "languageCode": "en"
  } ...
```

An important step of the development of the backend is handling the multi-session: when several users connect to the chatbot simultaneously, each session needs to maintain its own context and data, ensuring personalized and accurate responses without interference from other users.To do so, I created a dictionnary and associate a key for each session ID.
This isolation guarantees a smooth and secure experience, even with numerous interactions running in parallel.

Finally, the last step of the backend was to create a SQL database and establish a connexion with the python script.

The database is really simple. It contains the three following tables:

#	product_id	product_name
1	1	MedIAg
2	2	SyntheMedix
3	3	ForecastMed
*	NULL	NULL

products:
A unique ID (primary key) is associated to each product

#	customer_id	phone_number	email_address
1	1	0123456789	contact1@example.com
2	2	1234567890	contact2@example.com
3	3	2345678901	contact3@example.com
4	4		contact1@test.com
5	5	1234567890	contact2@test.com

customers:
A unique ID (primary key) is associated to each customer, with their coordinates

| #	customer_id	interests	date
1	4	1	05-11-2024 10:42:17
2	5	3	05-11-2024 11:06:02
3	6	\	05-11-2024 11:12:34
4	7	\	05-11-2024 11:16:35

interests:
For each customer (called by their ID as a foreign key from the previous table), we enter the interests and the date of the chat. The column interest contains a str with the ID of each product of interest, or '\' if none has been mentioned

Thanks to the library mysql.connector, the database is automatically completed at the end of a chat with a user.

5. Frontend development

Finally, I wanted to implement a lightweight frontend to support my chatbot. I mainly used ChatGPT to generate index.html and style.css files, resulting in the following interface:
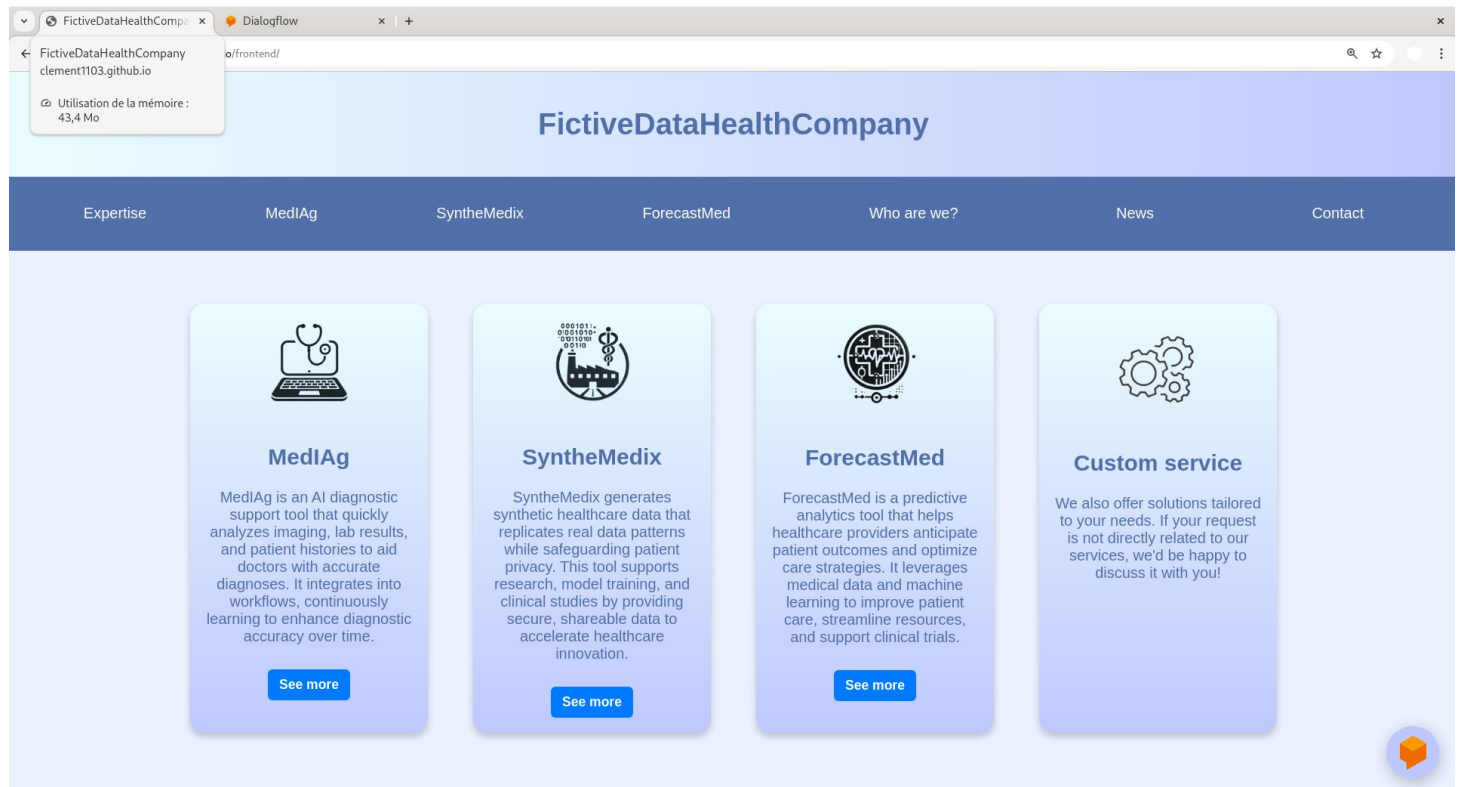
There is a brief presentation of each product, and a navigation banner where the user can click in order to be redirected to other pages (the redirections are not functional).

At the bottom right, there is an icon allowing to open and close the chatbot window.

6. Deployment and tests

The final step of this project was hosting the frontend, the backend scripts and the SQL database online.

- To host the frontend, I used Github pages which allows to host static websites directly from a GitHub repository, making HTML, CSS and JavaScript files accessible online without server configuration.

- To host the backend, I chose Heroku, a cloud platform that makes it easy to deploy, manage and scale web applications by running their backend and associated services.

Note: The backend and the frontend codes are present in the repository */data_science_projects/DL_projects/chatbot_health* but they are deployed on another github repository.

- For the SQL database, I chose the free plan of AWS RDS which hosts and manages SQL databases in the cloud, automating maintenance tasks such as backups and security. For safety reasons, the database identifiers written in the shared python scripts are not the correct ones.

You can access to the chatbot following this link:
https://clement1103.github.io/frontend/

Note: The program works perfectly locally, accessing the database stored on AWS, but I've noticed that the backend sometimes malfunctions when deployed on heroku, due to poor management of the variable corresponding to the different sessions. In fact, the "sessions" variable is stored in volatile memory and is sometimes reset during the chat. I intended to use Redis to improve the storage of variables, but there is no free plan available.
Moreover, the hosting on heroku may be deprecated after a certain time. In this case, the chat is going to end the conversation after the user type his details.

Here is an example of conversation flow:

chatbot_health

Hello and welcome to FictiveDataHealth! I'm here to guide you through our products or help you schedule a meeting with an expert. You can type something like "Product presentation" or "Meeting with expert".

> Hi, I would like a presentation of your services please.

The user asks for a presentation of the products

We have three main solutions, covering several areas:
- MedIAg, a diagnostic assistant
- SyntheMedix, a synthetic data generation tool
- ForecastMed, allowing predictive analytics for healthcare

Which one are you interested in?

> ForecastMed

The user asks for a more detailed presentation of ForecastMed

ForecastMed

ForecastMed is a state-of-the-art predictive analytics tool that empowers healthcare organizations to anticipate patient outcomes and optimize care strategies. By analyzing large-scale medical data, ForecastMed forecasts key trends such as disease progression, patient risk factors, and hospital readmission rates. With its

Ask something...

---

chatbot_health

helps you stay ahead by making data-driven decisions that lead to better patient care and operational efficiency.

You can read more at https://fictivelink_for_fictivehealthcompany.com/ForecastMed/.

Are you interested in the product?

> Yes

The user confirms the interest for ForecastMed

All right! Your interest has been taken into account. Would you like to know more about another service?

(As a reminder, we offer the following services: predictive analysis with ForecastMed, synthetic data generation with SyntheMedix and diagnosis assistance with MedIAg.)

> SyntheMedix

The user then asks to discover SyntheMedix

SyntheMedix

SyntheMedix is an innovative solution specialized in the generation of synthetic data for the healthcare sector. Using advanced artificial intelligence algorithms, SyntheMedix recreates medical datasets that faithfully mimic the characteristics of real data, while preserving patient confidentiality. This technology enables researchers, hospitals and healthcare companies to access valuable information to

Ask something...

---

chatbot_health

You can read more at https://fictivelink_for_fictivehealthcompany.com/SyntheMedix/.

Are you interested in the product?

[Yes] [No]

> No

The user is not interested in SyntheMedix

I am sorry that the service does not suit your needs. Do you want to discover another product?

(As a reminder, we offer the following services: predictive analysis with ForecastMed, synthetic data generation with SyntheMedix and diagnosis assistance with MedIAg.)

[MedIAg] [SyntheMedix]

[ForecastMed] [No, thank you]

> No I am good, thanks

The user doesn't want to discover other products

Understood. Please, type your e-mail address or phone number so that an expert contact you in the next 24

Ask something...

---

chatbot_health

[MedIAg] [SyntheMedix]

[ForecastMed] [No, thank you]

> No I am good, thanks

The user doesn't want to discover other products

Understood. Please, type your e-mail address or phone number so that an expert contact you in the next 24 hours to discuss about the products which drew your attention.

> My email address is contact1@report.com and my phone number is 0123456789

The user gives his coordinates

Are the following information correct:
Email address:
contact1@report.com
Phone number: 0123456789

> Yes, they are.

The user confirms his coordinates

Great! An expert will get in touch with you within the next 24 hours. Thanks for using the chatbot!

Ask something...

Following this chat, here are the rows added to the database:

#	customer_id	phone_number	email_address
1	21	0123456789	contact1@report.com

customers table

#	customer_id	interests	date
1	21	3	05-11-2024 21:19:45

interests table
(3 is the index of ForecastMed)

Here are some other extracts of other flows, illustrating that the chatbot is polyvalent.

chatbot_health

to be generated in record time, accelerating

Presentation of SyntheMedix

Are you interested in the product?

Yes   No

What are the main algorithms used here?

I apologize, I don't have enough information about your specific problem at this stage. Would you like me to put you in touch with an expert for more details?

Ask something...

Please, type your e-mail address or phone number so that an expert contact you in the next 24 hours in order to answer to all your interrogations.

The user gives wrong details.

My phone number is wrongphonenb and my email address is wrongemailaddress.

The chatbot detects that the given details are not correct.

Please, enter correct pieces of information.

The user asks a specific question the chatbot is not able to answer

The chatbot detects the complex question and offers a redirection toward an expert

7. Conclusion

In the end, this project proved to be highly enriching, through an immersion into the end-to-end design and deployment of a chatbot system. Not only did it allow me to discover the full process of tool development, from the initial specifications to the final deployment, but it also enhanced my understanding of the many challenges inherent in building an interactive and responsive application.

During this project, I faced challenges in requirement scoping, which led me to refine the chatbot's functionality to meet the needs of its intended users. Developing both the backend and frontend components required integrating multiple technologies, fostering my ability to seamlessly connect Dialogflow with custom server-side logic and a user-friendly client interface. Additionally, I managed a SQL database to support efficient data storage and retrieval, ensuring that the chatbot could interact dynamically with stored information.

Each stage of this project presented unique technical and problem-solving challenges, from architecting scalable backend logic to designing a responsive user interface. The final deployment phase underscored the importance of rigorous testing and optimization to deliver a stable and effective online solution.

In summary, this project allowed me to consolidate a diverse range of technical skills while deepening my understanding of software development workflows and deployment strategies.