

Lin Damien  
Clément Bosc  
M2/I3 DAI

# Chatbot & Generative AI Report

## The Workout Chatbot



## Table des matières

Introduction.....	4
Dataset .....	5
Chatbot developing.....	6
Functions.....	7
Youtube data API v3 .....	9
Streamlit Part.....	11
Streamlit .....	13
Innovation part.....	16
Conclusion : .....	17

## Introduction

During the gym subscription, they display different offers so you can choose the better suited option.

As students, we don't swim in gold coins and usually take the student option that is cheaper but has less advantages.

The people that did this project are aware of that because they also made the same mistake. They got this subscription, but they don't know what to do when going to gym. This is mainly because they had no coaches and nowhere to find exercise.

For the chatbot & gen ai class we had to do a project by creating a chatbot. We first thought about doing a chatbot that could play connect 4 with us but it was less interesting than doing a chatbot that could recommend workout exercises to people. This way we could help people to not feel lost during their gym sessions... At least less than us at the beginning of our subscription.

## Dataset

First task was finding a dataset containing different gym exercises on the internet.

We searched on huggingface and Kaggle because they're the most famous website for datasets and we found one on Kaggle which is not that big named "megaGymDataset"

It looked like this :

```
,Title,Desc,Type,BodyPart,Equipment,Level,Rating,RatingDesc
0,Partner plank band row,The partner plank band row is an abdominal exercise where two partners perform single-arm planks while pulling on the opposite ends of a resistance band.
1,Banded crunch isometric hold,The banded crunch isometric hold is an exercise targeting the abdominal muscles, particularly the rectus abdominis or "six-pack" muscles. The band is placed around the ankles and the hands are placed behind the head.
2,FYR Banded Plank Jack,The banded plank jack is a variation on the plank that involves moving the legs in and out for repetitions. Having a band around the ankles adds resistance.
3,Banded crunch,The banded crunch is an exercise targeting the abdominal muscles, particularly the rectus abdominis or "six-pack" muscles. The band is placed around the ankles and the hands are placed behind the head.
4,Crunch,The crunch is a popular core exercise targeting the rectus abdominis, or "six-pack" muscles, as well as the obliques. It has been the centerpiece of many fitness routines.
5,Decline band press sit-up,The decline band press sit-up is a weighted core exercise that works the rectus abdominis or "six pack" muscles, as well as the obliques.
6,FYR2 Banded Frog Pump,,Strength,Abdominals,Bands,Intermediate,,
7,Band low-to-high twist,The band low-to-high twist is a core exercise targeting the upper abdominals and the obliques. Take care to perform it with control.
8,Barbell roll-out,The barbell roll-out is an abdominal exercise that utilizes a barbell in the place of an ab roller. It is best performed with a barbell that has a wide grip.
9,Barbell Ab Rollout - On Knees,The barbell roll-out is an abdominal exercise that utilizes a barbell in the place of an ab roller. It is best performed with a barbell that has a wide grip.
10,Decline bar press sit-up,The decline bar press sit-up is a weighted core exercise targeting the abdominal muscles, particularly the lower abs. It also works the obliques.
11,Bench barbell roll-out,The bench barbell roll-out is a challenging exercise targeting the abdominals. It is similar to using an ab roller, but using a barbell.
12,Barbell Side Bend,,Strength,Abdominals,Barbell,Beginner,7.0,Average
13,Seated bar twist,The seated bar twist is a core exercise meant to strengthen the obliques. It is often performed for high reps with relatively light weights.
14,Single-arm landmine pull and press,The single-arm landmine pull and press is an explosive rotational movement using a barbell anchored in a landmine drill.
15,Barbell Ab Roll Out - Gethin Variation,The barbell roll-out is an abdominal exercise that utilizes a barbell in the place of an ab roller. It is best performed with a barbell that has a wide grip.
16,30 Barbell Floor Wiper,The barbell floor wiper is a core exercise in which the barbell is held in the locked-out position of a floor press, and the hands are moved in a circular motion.
17,30 Barbell Roll-Out,The barbell roll-out is an abdominal exercise that utilizes a barbell in the place of an ab roller. It is best performed with a barbell that has a wide grip.
18,Decline plate sit-up,The decline plate sit-up is a weighted core exercise that works the rectus abdominis or "six pack" muscles, as well as the obliques.
19,KV Barbell Hip Thrust,,Strength,Abdominals,Barbell,Intermediate,,
20,Advanced Kettlebell Windmill,,Strength,Abdominals,Kettlebells,Beginner,8.3,Average
21,Kettlebell Windmill,The single-kettlebell windmill is a dynamic kettlebell exercise emphasizing core strength and shoulder and hip mobility and stability.
22,Kettlebell Pass Between The Legs,,Strength,Abdominals,Kettlebells,Beginner,7.3,Average
23,Kettlebell 3-point leg extension,The kettlebell 3-point leg extension is a dynamic core exercise performed around and over a kettlebell or other short object.
24,Double Kettlebell Swing,,Strength,Abdominals,Kettlebells,Intermediate,,
25,Burpee over kettlebell,The burpee over kettlebell is a more advanced version of a popular conditioning exercise. If you're performing a circuit that includes burpees, this is a great variation.
26,Kettlebell toe-touch,The kettlebell toe-touch is a more difficult version of a highly popular abdominal movement usually done for high reps. The simple version involves reaching for the toes.
27,Kettlebell swing,The kettlebell swing is a popular lower-body exercise emphasizing the hamstrings, glutes, and back muscles. It is often used to train for power and endurance.
28,Two-way swing,The two-way kettlebell swing is an exercise that alternates a chest-height kettlebell swing (sometimes called a "Russian" or "hardstyle") with a full swing.
29,Kettlebell crab reach,The kettlebell crab reach is a kettlebell pressing exercise in which the lifter presses a kettlebell overhead while in a modified crab position.
30,Taylor Single-Arm Kettlebell Swing,,Strength,Abdominals,Kettlebells,Intermediate,,
```

So the features were :

- Title, containing the name of the exercise
- Desc, containing a description of the exercise, note that this isn't fully filled
- Type, containing the type of training we wanted to perform
- BodyPart, containing the body part we wanted to train
- Equipment, containing the necessary equipment we needed to train
- Level, The level of the exercise, if it was needed to be an expert to perform it
- Rating, the rating of the exercise
- RatingDesc, the rating of the description of the exercise

After downloading it we needed to preprocess it, We basically didn't need the rating part since we were not training a model or anything and we didn't want our recommendations be based on the rating, we wanted the user to discover different type of exercises and let them the will to remember what they liked.

## Chatbot developing

After downloading and processing the dataset we had, we wanted to create the chatbot part.

We needed to create a keyword dictionary containing the main keyword we wanted.

```
keywords_dict = {
    'greeting': [r'.*\bhell+o+\b.*|\.*\bhi+\b.*|\.*\bho+w+dy\b.*|\.*\bhul+lo+\b.*|\.*\bhey\b.*|\.*\bho+I+a+\b.*'],
    'bye': [r'.*\bby+e+\b.*|\.*\bgoo+db+ye\b.*|\.*\bsee you\b.*|\.*\btake care\b.*|\.*\bcy+a+\b.*|\.*have a great (night|da+y+).*'],
    'time_query': [r'.*\btime\b.*|\.*\btime\b.*|\.*\bwhat time is it\b.*|\.*\btell me the time\b.*|\.*do you have the time\b.*'],
    'name_query': [r'.*\byour name\b.*|\.*who are you\b.*|\.*what is your name\b.*'],
    'age_query': [r'.*\bhow old are you\b.*|\.*\bwhat is your age\b.*|\.*\bwhen were you born\b.*|\.*\bage\b.*'],
    'thanks': [r'.*\bthank you\b.*|\.*\bthanks\b.*|\.*\bappreciate\b.*|\.*\bgrateful\b.*|\.*\bok\b.*'],
    'unknown': [r'.*\?+\b.*|\.*\bhmm\b.*|\.*\bno idea\b.*'],
    'workout': [r'.*\bworkout\b.*|\.*\bexercise\b.*|\.*\bgym\b.*|\.*\bfitness\b.*|\.*\btraining\b.*|\.*\bphysical activity\b.*|\.*\bworout\b.*|\.*\bworkut\b.*'],
    'weather_query': [r'.*\bweather\b.*|\.*\bforecast\b.*|\.*\btemperature\b.*|\.*\bis it (cold|hot|raining)\b.*'],
    'joke_request': [r'.*\bjoke\b.*|\.*\bI want to laugh\b.*|\.*\bmake me laugh\b.*'],
    'advice_request': [r'.*\bgive me advice\b.*|\.*\bwhat should I do\b.*|\.*\bhelp me\b.*'],
    'compliment': [r'.*\bYou are (awesome|great|amazing|smart|cool)\b.*|\.*\bI love you\b.*'],
    'insult': [r'.*\bYou are (dumb|stupid|useless|bad)\b.*|\.*\bI hate you\b.*'],
    'mood_query': [r'.*\bhow are you\b.*|\.*\bhow do you feel\b.*|\.*\bhow\s it going\b.*'],
    'description': [r'.*\b(?:describe me|what is|tell me more about) (.+)*'],
    'fun_fact': [r'.*\btell me a fact\b.*|\.*\bgive me a fun fact\b.*|\.*\bI want to learn something new\b.*'],
    'workout': [r'.*\bworkout\b.*|\.*\bexercise\b.*|\.*\bgym\b.*|\.*\bfitness\b.*'],
    'food_query': [r'.*\bwhat should I eat\b.*|\.*\brecipe\b.*|\.*\bcook\b.*|\.*\bfavorite food\b.*'],
    'hobby_query': [r'.*\bwhat do you like to do\b.*|\.*\bany hobbies\b.*'],
    'bored_query': [r'.*\bI am bored\b.*|\.*\bwhat should I do\b.*|\.*\bentertain me\b.*'],
    'relationship_query': [r'.*\bdo you have a girlfriend\b.*|\.*\bare you single\b.*|\.*\bdo you love me\b.*'],
    'life_advice': [r'.*\bgive me life advice\b.*|\.*\bhow to be happy\b.*|\.*\bany tips for life\b.*'],
    'dreams_query': [r'.*\bdo you dream\b.*|\.*\bwhat do you dream about\b.*'],
    'money_query': [r'.*\bhow to make money\b.*|\.*\bget rich\b.*|\.*\bbest way to earn\b.*'],
    'daily_plan': [r'.*\bwhat should I do today\b.*|\.*\bplan my day\b.*'],
    'pet_query': [r'.*\bdo you like animals\b.*|\.*\bdo you have a pet\b.*|\.*\bfavorite animal\b.*'],
    'social_media_query': [r'.*\bare you on social media\b.*|\.*\bwhat\s trending\b.*|\.*\bviral\b.*'],
    'fashion_query': [r'.*\bwhat should I wear\b.*|\.*\bfashion advice\b.*'],
    'celebrity_query': [r'.*\bfavorite celebrity\b.*'],
    'agreement_query': [r'.*\bI agree\b.*|\.*\bI think so\b.*|\.*\bI believe\b.*'],
    'Quoi': [r'.*\bquoi\b.*|\.*\bwait what\b.*'],
}
```

And put it in a file keywords\_answers.py

Then we did the same but for the responses we wanted our chatbot to display.

```
responses_dict = {
    'greeting': [
        "Hello! How can I help you today?",
        "Hey there! What\s up?",
        "Hi! What\s on your mind?",
        "Howdy! Need anything?",
        "Hey! How\s your day going?"
    ],
    'bye': [
        "Goodbye! Have a great day!",
        "See you later! Take care!",
        "Bye-bye! Stay safe!",
        "Catch you later!",
        "Hope to chat again soon!"
    ],
    'time_query': [
        "It's time to get a watch!",
        "Right now, it's .. Nah idk don't have this part implemented",
        "Checking the time... U're gonna wait forever to get the answer",
        "Time flies!",
    ],
    'name_query': [
        "I'm your friendly chatbot!",
        "They call me Chatty!",
        "I'm just a bot, but you can call me your virtual assistant.",
        "I go by many names, but you can call me ChatBot.",
        "I'm here to help! No need for formalities."
    ],
    'age_query': [
        "I was born in the digital world, so I don't really age!",
        "Age is just a number... and I don't have one!",
        "I've been around as long as the internet has allowed me to exist!",
        "Let\s just say I'm young at heart!",
        "I don't celebrate birthdays, but thanks for asking!"
    ]
}
```

Now our chatbot had almost everything on hand to chat with our user. The only thing that he didn't have was the function to chat with users.

## Functions

We started by doing a function `recommend_workout` that could take the preferences of the users and choose randomly if they use any we can choose anything in the feature from the database.

```
def recommend_workout(preferences, quantity=5):
    """
    Recommend workouts based on user preferences.
    :param preferences: A dictionary with keys like 'Type', 'BodyPart', 'Equipment', and 'Level'.
    :return: A list of recommended workouts.
    """
    filtered_data = data.copy() # Create a copy of the data to avoid modifying the original

    for key, value in preferences.items():
        if value and value.lower() != "any":
            print(value)
            # Filter where the value matches OR where it's "None"
            filtered_data = filtered_data[filtered_data[key].str.contains(value, case=False, na=False)]
            if value == "Equipment":
                filtered_data = filtered_data[
                    filtered_data[key].str.contains(value, case=False, na=False) |
                    filtered_data[key].str.contains("Body Only", case=False, na=False)
                ]
            else:
                continue

    if filtered_data.empty:
        return ["Sorry, no workouts match your preferences. Try adjusting them!"]

    return filtered_data['Title'].sample(n=min(quantity, len(filtered_data))).tolist() # Return up to 5 random recommendations
```

We needed to match the pattern of our keyword dictionary

```
def matchPattern(user_msg):
    user_msg = user_msg.lower()
    for intent, patterns in keywords_dict.items():
        for pattern in patterns:
            match = re.search(pattern, user_msg)
            if match:
                return intent
    return 'unknown'
```

Then we coded the chatbot for it to display our responses.

```
def chatbot(user_input):
    url_list = []
    matched_intent = matchPattern(user_input)
    # Initialize session state variables
    if "chat_step" not in st.session_state:
        st.session_state.chat_step = 0 # Tracks which step we're on
        st.session_state.workout_preferences = {} # Stores user choices

    # Step-by-step conversation flow
    if st.session_state.chat_step == 0:
        if matched_intent == "workout":
            st.session_state.chat_step += 1
            return "Great! Let's find a workout for you. What type of workout are you looking for? (Strength, Cardio, etc.)", url_list
        elif matched_intent == "description":
            sport = description(user_input)
            if sport != 'unfound_desc':
                sport_desc = lower_data.loc[lower_data["Title"] == sport, "Desc"]
                if sport_desc.empty or pd.isna(sport_desc.values):
                    return random.choice(responses_dict["unfound_desc"]), url_list
                else:
                    return sport_desc.values[0], url_list
            else:
                return random.choice(responses_dict["unknown_sport"]), url_list
```

We added some features like if the user needs a description of the sport he can find the description in our dataset.

```
def description(request):
    request = request.lower()
    for pattern in keywords_dict["description"]:
        match = re.match(pattern, request)
        if match:
            sport = match.group(1).strip()
            return sport
    return 'unknown_sport'
```

We did a description function to retrieve the workout and give it to our chatbot so he can get the workout information.

We also created a chat\_step to meet the needs of the users in terms of workout.

```
def chatbot(user_input):
    elif st.session_state.chat_step == 1:
        user_input = st.session_state.messages[-1]["content"]
        types_lower = [str(x).lower() for x in data["Type"].unique()]
        if user_input.title().lower() in types_lower or user_input.title().lower() == "any":
            st.session_state.workout_preferences["Type"] = user_input.title()
            st.session_state.chat_step += 1
            return "Which body part do you want to target? (Abdominals, Biceps, Chest, Any etc.)", url_list
        else:
            return "Please enter a valid workout type: Strength, Plyometrics, Cardio, Stretching, Powerlifting, Strongman, Olympic Weightlifting, Any", url_list

    elif st.session_state.chat_step == 2:
        user_input = st.session_state.messages[-1]["content"]
        bodyp_lower = [str(x).lower() for x in data["BodyPart"].unique()]
        if user_input.title().lower() in bodyp_lower or user_input.title().lower() == "any":
            st.session_state.workout_preferences["BodyPart"] = user_input.title()
            st.session_state.chat_step += 1
            return "Any specific equipment? (Bands, Dumbbell, Barbell, Bands, Body Only, Any, etc.)", url_list
        else:
            return "Please enter a valid body part: Abdominals, Adductors, Abductors, Biceps, Calves, Chest, Forearms, Glutes, Hamstrings, Lats, Lower Back, Middle Back, Traps, Neck, C

    elif st.session_state.chat_step == 3:
        user_input = st.session_state.messages[-1]["content"]
        equip_lower = [str(x).lower() for x in data["Equipment"].unique()]
        if user_input.title().lower() in equip_lower or user_input.title().lower() == "any":
            st.session_state.workout_preferences["Equipment"] = user_input.title()
            st.session_state.chat_step += 1
            return "What difficulty level? (Beginner, Intermediate, Expert, Any)", url_list
        else:
            return "Please enter a valid equipment type: Bands, Barbell, Kettlebells, Dumbbell, Other, Cable, Machine, Body Only, Medicine Ball, Exercise Ball, Foam Roll, E-Z Curl Bar,

    elif st.session_state.chat_step == 4:
        user_input = st.session_state.messages[-1]["content"]
        level_lower = [str(x).lower() for x in data["Level"].unique()]
        if user_input.title().lower() in level_lower or user_input.title().lower() == "any":
            st.session_state.workout_preferences["Level"] = user_input.title()
            st.session_state.chat_step += 1

            # Now that we have all inputs, generate recommendations
            preferences = st.session_state.workout_preferences
            recommendations = recommend_workout(preferences)
            if recommendations == ["Sorry, no workouts match your preferences. Try adjusting them!"]:
                response = recommendations[0]
            else:
                response = "Here are some recommended workouts:\n"
                for workout in recommendations:
                    title, url = search_youtube_video(workout)
                    if title and url:
```

This way we get the Type, the bodypart, the equipment and the level of the workout the user wants.



```

# Now that we have all inputs, generate recommendations
preferences = st.session_state.workout_preferences
recommendations = recommend_workout(preferences)
if recommendations == ["Sorry, no workouts match your preferences. Try adjusting them!"]:
    response = recommendations[0]
else:
    response = "Here are some recommended workouts:\n"
    for workout in recommendations:
        title, url = search_youtube_video(workout)
        if title and url:
            response += f"- {workout}\n"
            url_list.append(url)
        else:
            response += f"- {workout}, but no video found.\n"

# Reset state after providing recommendations
st.session_state.chat_step = 0
st.session_state.workout_preferences = {}
response += {random.choice(responses_dict["video"])}
return response, url_list

else:
    return "Please enter a valid difficulty level: Beginner, Intermediate, Expert, Any.", url_list

```

Then when every information is retrieved we give it to our recommend\_workout function and display them in response. If user answers aren't in the dataset we want we ask them again to complete the step. If every steps are complete we go back to the step 0 with the normal chatbot. We also return the videos found about the workout on youtube thanks to the Youtube data API v3.

### Youtube data API v3

We give a query to the API to find out the video best describing the retrieved results from our chatbot

```

from googleapiclient.discovery import build
from Api_key import YOUTUBE_API_KEY

def search_youtube_video(query):
    """
    Search for a YouTube video using the YouTube Data API.
    :param query: The search query (e.g., workout title or keyword).
    :return: The title and URL of the first video found.
    """
    youtube = build('youtube', 'v3', developerKey=YOUTUBE_API_KEY)

    # Search for the video
    request = youtube.search().list(
        part='snippet',
        q=query,
        type='video',
        maxResults=1
    )
    response = request.execute()

    # Extract video information
    if response['items']:
        video = response['items'][0]
        title = video['snippet']['title']
        video_id = video['id']['videoId']
        video_url = f"https://www.youtube.com/watch?v={video_id}"
        return title, video_url
    else:
        return None, None

```

Here's the function; we of course will not display our youtube api key that is on  
Api\_key.py file

And the Api key file has only "YOUTUBE\_API\_KEY = "API\_key""

We also did a freeze to get a requirements.txt file containing all the libraries we used on  
our project and a readme file explaining our files and how to run our program.

## Streamlit Part

After all this we wanted a graphic interface, we first thought about discord since it could directly display the videos without any searches then we went on streamlit since it was allowed on streamlit and that it has a complete documentation to do a chatbot. We could also deploy it if we wanted.

To begin with streamlit we gave him a title “workout chatbot” we could have found better but we didn’t

```
# Streamlit user interface

st.title("Workout Chatbot")

if "messages" not in st.session_state:
    st.session_state.messages = []
    message = {"role": "assistant", "content": "Hello! I'm your chatbot assistant. How can I help you today?"}
    st.session_state.messages.append(message)
```

The first displayed message will be from the assistant (the chatbot) and the content displayed would be “Hello! I’m your chatbot assistant. How can I help you today?”

We decided to store the messages so we could have a chat history:

```
# Display chat messages from history on app rerun
for message in st.session_state.messages:
    with st.chat_message(message["role"]):
        st.markdown(message["content"])
```

Then we made the input for the users with prompt, and we added it to our history after that:

```
# React to user input
if prompt := st.chat_input("Say something..."):
    # Display user message in chat message container
    with st.chat_message("user"):
        st.markdown(prompt)
    # Add user message to chat history
    st.session_state.messages.append({"role": "user", "content": prompt})
```

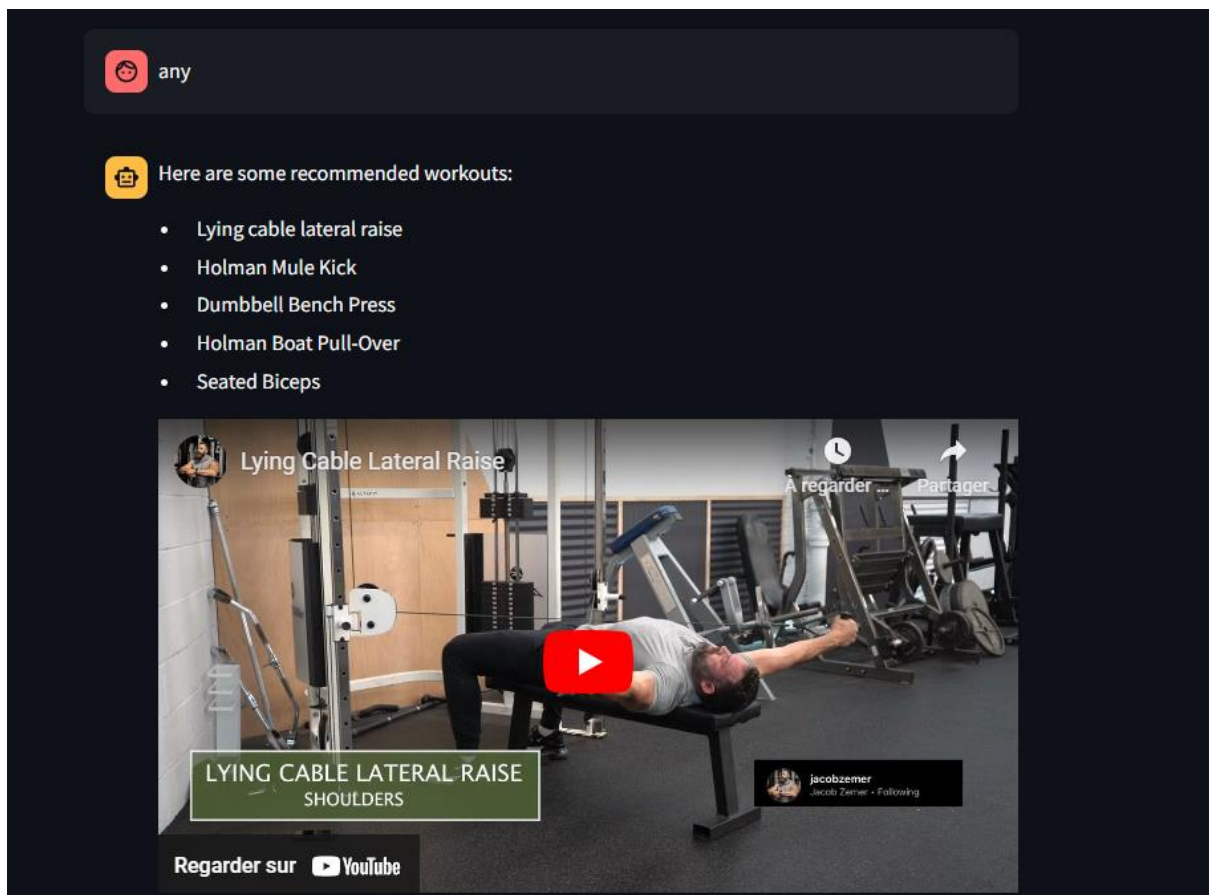
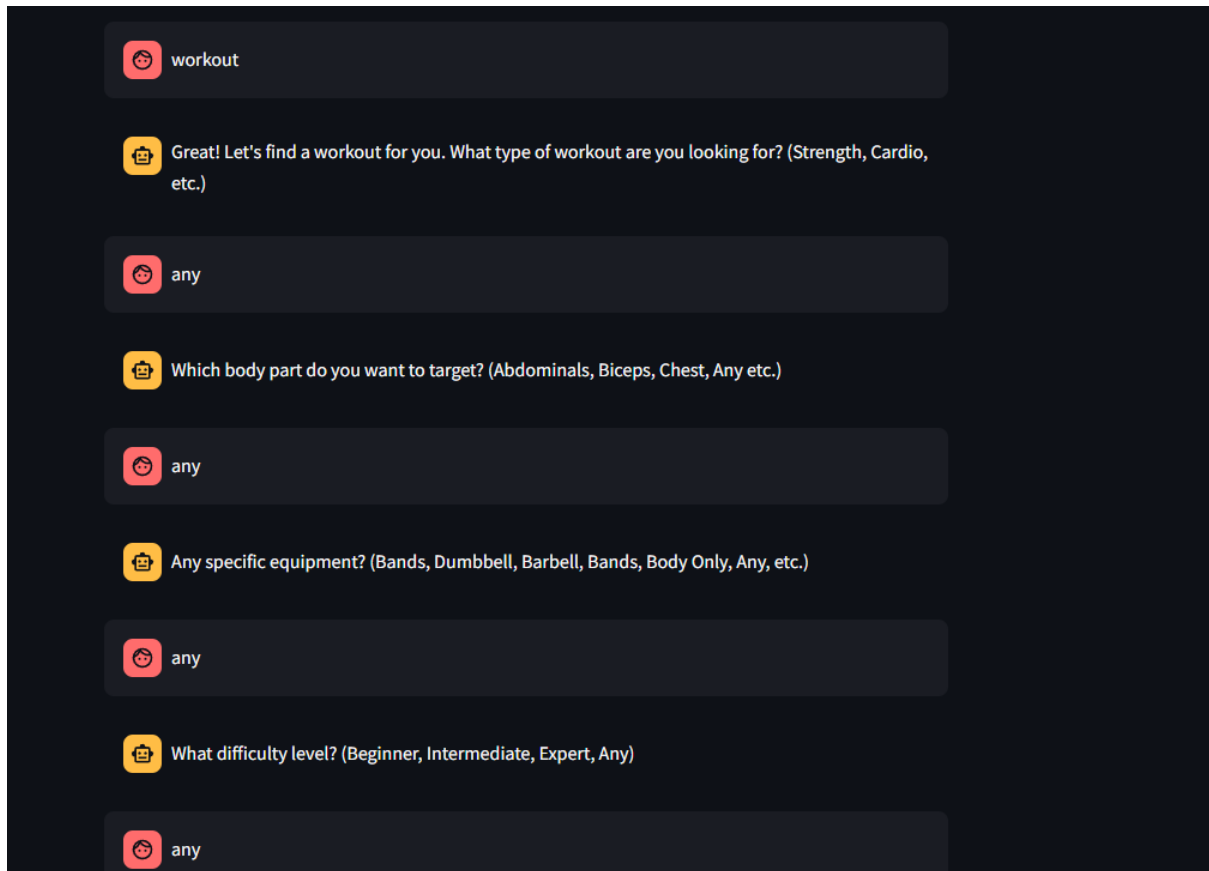
After doing the user’s prompt we had to display the answer of the chatbot :

```
# Generate a new response if last message is not from assistant
if st.session_state.messages[-1]["role"] != "assistant":
    with st.chat_message("assistant"):
        with st.spinner("Thinking..."):
            response, url_liste = chatbot(prompt)
            message = {"role": "assistant", "content": response}
            st.write(response)
            if url_liste is not None:
                for url in url_liste:
                    st.video(url)
    st.session_state.messages.append(message)
```

We made it so it could act after the user's prompt. It gives the prompt to our chatbot, and it displays the associated response. If it was a workout prompt the chatbot, at the end, displayed the video adapted to show the exercise. After this we added it to the history of the chat.

## Streamlit screens

Here's some screenshots of our chatbot :





## Vidéo non disponible

Le propriétaire de la vidéo a désactivé la lecture sur d'autres sites Web

[Regarder sur YouTube](#)



Best Dumbbell Bench Press Tutorial Ever Made




À regarder ...



Partager



BEST  
DB BENCH  
GUIDE

Regarder sur  YouTube



Hello



Hey there! What's up?



give me a joke



Why don't skeletons fight each other? Because they don't have the guts!



describe me Lying cable lateral raise



The lying cable lateral raise is an isolation exercise that targets the medial or middle deltoid muscle of the shoulders. It is most common in muscle-building shoulder or upper-body workouts, usually in higher rep ranges of at least 8-12 reps per set.

## Innovation part

We didn't really know what to implement on our chatbot that we didn't already see in TP's but we found out that using youtube API to find the adapted video was a good innovation.

We also used Streamlit since it was suited for our project and good for deployment even though the object creating process was hard.

These are the main innovations. The hardest part was to do the chat steps since we had to combine this with streamlit function. We first started to do the function to run it on visual studio Code but we had to change everything, so it can suit Streamlit.



## Conclusion :

This project was for us a real improvement in our knowledge. It gave us the opportunity to use what we learnt in class in our chat but also to look at ourselves at what could be done. It also requested our imagination to find out features and interesting stuff to add to it.

We also feel proud since it's a project that could help a lot of lost people like us at the start of our subscription.