

UNIVERSITY OF SCIENCE
Vietnam National University - Ho Chi Minh City



RECIPE-GENERATING CHATBOT WITH OPENAI'S API

Instructor: Do Hoang Cuong

Ho Duy Khang	21120260
Le Thi Minh Phuong	21120533
Do Huu Huy Hoang	21120378

Table of Contents

I. INTRODUCTION	3
1.1. Chatbot and API	3
1.2. OpenAI, its API and Chat-GPT	3
II. CREATING THE CHATBOT WITH OPENAI'S API	5
2.1. Ideation and Platform Evaluation	5
2.2. Chat-GPT Integration with The Platform	6
2.2.1. Create the web platform	6
2.2.2. Integrate the chatbot with the platform	7
2.3. Script and Training Data Creation	10
2.3.1. Write the chatbot script	10
2.3.2. Prepare training dataset	10
2.3.3. Train the 'davinci' model	10
2.4. Testing and Debugging	11
2.4.1. Use a no-credit account	11
2.4.2. Use 'davinci' model	11
2.4.3. Use 'text-davinci-003' with the English initial prompt	13
2.4.4. Use 'text-davinci-003' with the Vietnamese initial prompt	14
2.4.5. Finalize English and Vietnamese initial prompt	16
III. ADDING CUSTOM FUNCTIONALITIES AND INTEGRATIONS	18
3.1. Speech-to-text	18
3.2. Text-to-speech	19
3.3. Food recognition	20
3.4. Bilingual text communication	22
3.5. Image generation based on food names	24
V. CONCLUSION	26
VI. SOURCE CODE INSTRUCTIONS	27
VII. REFERENCES	28

I. INTRODUCTION

1.1. Chatbot and API

Chatbot (or Chatterbot) is a program that simulates a conversation with human users.

Classification:

- Rule-based chatbots: they are programmed to respond to specific keywords or phrases.
- Artificial intelligence-based chatbots: they use machine learning algorithms to analyze and understand user inputs.

Usage: Customer service, marketing and sales, education and training, personal assistant, recommender system, etc.

Strengths and weaknesses:

- Strengths: Cost-effectiveness, convenience, scalability.
- Weaknesses: Limited responses, unnaturally replicating human conversation.

Considering the weaknesses, to get the most out of the chatbot, it is necessary to design an effective conversation flow, handle common challenges of the chatbot (uncertainty, multi-users), and integrate it into systems and processes.

API (application programming interface) is a type of software interface for two or more computer programs to communicate with each other. In building applications, an API simplifies programming by abstracting the underlying implementation and only exposing objects or actions the developer needs.

1.2. OpenAI, its API and Chat-GPT

OpenAI is an artificial intelligence research laboratory consisting of a team of scientists and engineers who are dedicated to developing cutting-edge AI technologies. The company was founded in 2015 to develop AI safely and beneficially for humanity. OpenAI's work spans many different areas of AI research, including natural language processing, computer vision, and reinforcement learning. Their models have achieved impressive results in tasks such as language translation, image recognition, and game-playing. OpenAI's research and developments have contributed significantly to advancing the field of artificial intelligence, and their work has the potential to revolutionize many industries and improve people's lives in a variety of ways.

OpenAI's API is a multi-purpose API announced in June 2020 that it said was “for accessing new AI models developed by OpenAI” to let developers call on it for “any English language AI task”.

OpenAI's generative models: GPT, GPT-2, GPT-3, Whisper, and GPT-4 (unreleased).

Released user interfaces: MuseNet and Jukebox (music), Microscope, DALL-E and CLIP (images), DALL-E 2, ChatGPT, and ChatGPT Plus.

ChatGPT is an artificial intelligence chatbot developed by OpenAI and launched in November 2022. It is built on top of OPENAI's GPT-3 family of large language models and has been fine-tuned using both supervised and reinforcement learning techniques. In comparison to its predecessor, InstructGPT, ChatGPT attempts to reduce harmful and deceitful responses.

ChatGP's capabilities:

- Generating coherent and relevant responses to user input (especially by remembering previous prompts given to it in the same conversation).
- Integrating with a variety of chatbot platforms and frameworks such as Facebook, Messenger, Slack, and Telegram.
- Offering many advanced features to further enhance the capabilities of the chatbot such as fine-tuning, name entity recognition, and sentiment analysis.

However, ChatGPT also has limitations:

- ChatGPT sometimes writes plausible-sounding but incorrect or nonsensical answers.
- ChatGPT is sensitive to tweaks to the input phrasing or attempting the same prompt multiple times.
- The model is often excessively verbose and overuses certain phrases, such as restating that it's a language model trained by OpenAI.
- The refusal of inappropriate requests is sometimes tricked because of false negatives and positives in Moderation API.

II. CREATING THE CHATBOT WITH OPENAI'S API

2.1. Ideation and Platform Evaluation

Based on the capabilities of chatbots and openAI's models, many chatbot goals have been considered, including:

- **Healthcare Assistant:** A healthcare chatbot that gives diagnosis based on users' symptoms, then automatically advises on diet, exercises, or recovery plans to help the user improve their situation. However, choosing this solution will result in the chatbot giving unreliable advice and no practical movements can be made.
- **Children Education:** An education chatbot that can read children's books, draw entertaining images, and help busy parents look after their children. However, it is not advisable that children use electronic devices too much at an early age. Therefore, making a chatbot for children to interact with daily may spoil their natural growth.
- **Meeting Assistant:** A chatbot that is integrated into meeting platforms such as Zoom, Google Meet, etc. It can record important events, take meeting minutes, summarize meeting content, and visualize the speaker's idea in real-time.
- **Interview Trainer:** A chatbot that can simulate job interviews. After the interview, it can give users personalized advice on how to improve their performance. Furthermore, the chatbot is capable of finding available positions by scanning through the job hunting websites and providing users with the details of the host company. However, there are many headhunters hosting training interviews nowadays, so it is hard for an AI chatbot to gain people's trust more than real people.
- **Cooking Assistant:** A chatbot that can recommend recipes based on users' requests or leftover ingredients available in users' fridges. It will also support the voice assisting feature that can give voice instructions while the users are cooking their meals.

After considering all perspectives, the most inspiring idea was selected, which is the cooking assistant chatbot. Based on that idea, details of the chatbot were discussed and concluded as follows:

Chatbot's objectives: Provide a virtual bilingual (En-Vi) culinary assistant that gives users suggestions as well as detailed and comprehensive recipes for meals they can prepare using the ingredients they already have on hand or simply based on their requests via text or speech. Moreover, the image detection feature enables the chatbot to determine the recipe of dishes from the provided images and the image generation feature provides a visualization of dishes for users that have not seen particular dishes just by typing in their names.

Types of inputs and responses:

- Input: text or speech (list of ingredients, questions...), image (of a dish to be recognized), commands
- Output: text or speech (a list of dishes that are suitable for the given ingredients, a step-by-step recipe...), image (of a dish corresponds to the recipe)

Target audience: People who want to have recipes or meal suggestions quickly without investigating through the internet, including cooks, chefs, etc.

When choosing a platform for the chatbot, the criteria taken into consideration are listed in Table 1.

Platforms	User-friendliness	Level of customization	Level of integration with other systems	Pricing
Facebook Messenger	Yes	Low	Medium	Yes
Discord	No	Low	Medium	Yes
Telegram	No	Low	Medium	No
Template website	Yes	Medium	Medium	No
Self-built website	Yes	High	High	No

Table 1 – Evaluation of chatbot platforms

A custom-built website would be the most suitable and powerful option for the development of the chatbot, according to all the factors that were taken into account.

2.2. Chat-GPT Integration with The Platform

2.2.1. Create the web platform

After considering and evaluating available chatbot platforms, building a one-page website is the most sensible solution to reach a high level of user interface customization, behavior level as well as the level of integration.

Firstly, an HTML file was created to serve as the main page, which contains the basic structure of the website, including the header and body. The website was then styled using CSS to give it a pleasing and appropriate appearance. A chatbox with a text input box, a record button for voice input, a file browse button, and a scrollable chatlog set was placed in the middle of the page. Javascript, along with the hosted jQuery library, was utilized to add interactivity to the website.

For the back-end of the website, Flask, a Python web framework, was used to host the website and acted as a bridge between the user and the chatbot, which would be implemented in the next section.

2.2.2. Integrate the chatbot with the platform

To integrate Chat-GPT into the website, it is necessary to use OpenAI's API, which is available through the OpenAI module in Python. For Windows, in the command line, type the following to install the OpenAI module:

```
pip install openai
```

After that, we will have to obtain an API key from an OpenAI account. Simply log in to <https://platform.openai.com/>, click on the profile section at the top right of the page then select **View API keys**.

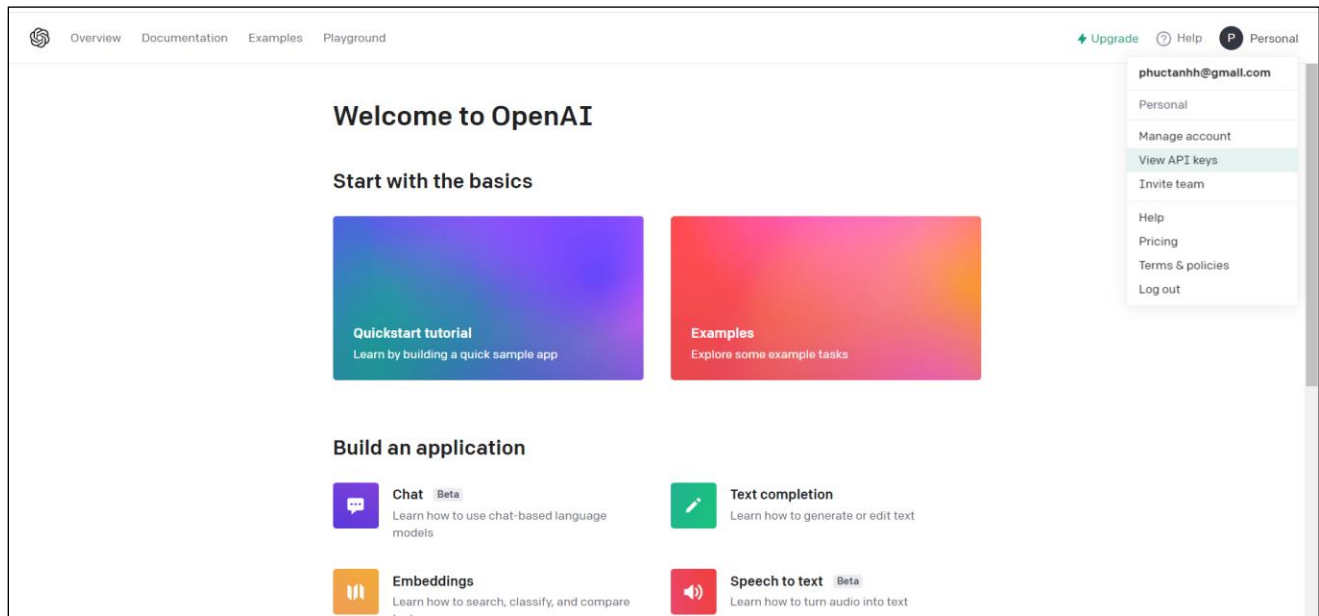


Figure 1 - The OpenAI website

In the API keys page, click on the **Create new secret key** button and copy the newly created key into somewhere for later use (Note that the key will not be able to be viewed again so save it somewhere easy to remember).

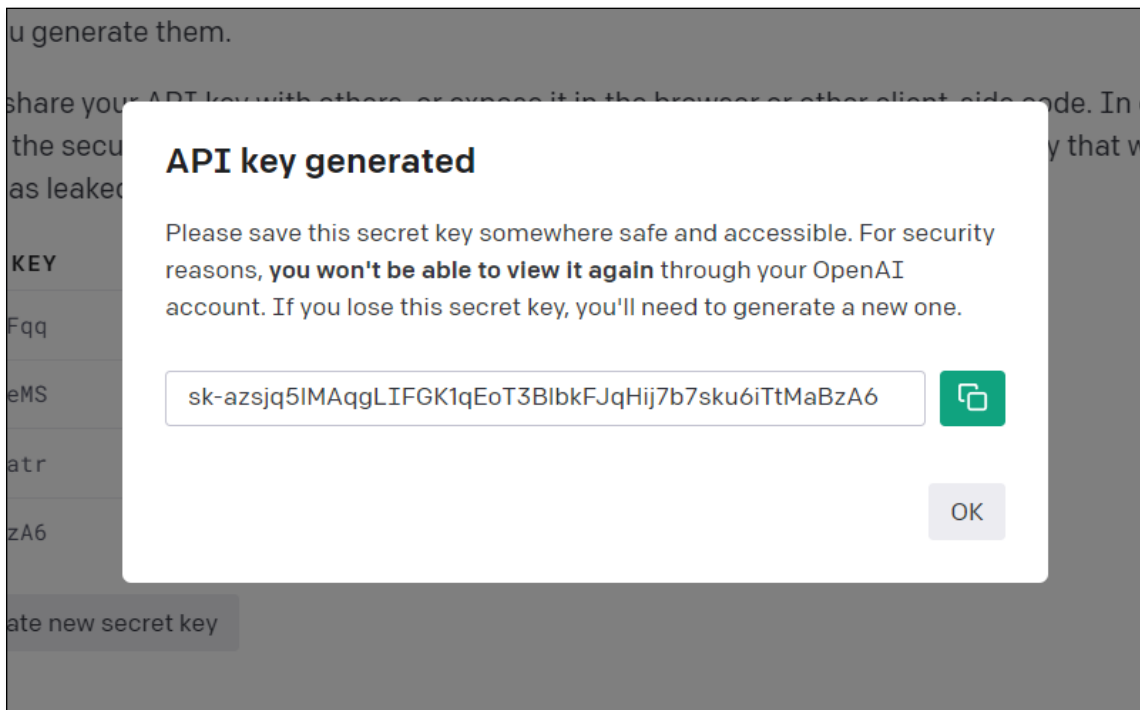


Figure 2 - Getting the API key

Next, a function that uses the OpenAI module to generate responses from a given prompt and the user input is defined using the following Python code.

```
def get_chatbot_response(input_text):  
  
    response = openai.Completion.create(  
        engine = "davinci",  
        prompt= _conversation_history + input_text,  
        temperature=0.7,  
        max_tokens=1024,  
        top_p=1,  
        frequency_penalty=0,  
        presence_penalty=0  
    )  
  
    return response.choices[0].text.strip()
```

Code of the get_chatbot_response() function that generates responses based on the user's input

To send a question from the user to the back-end to be processed, we write a Javascript script that sends the question to the server via an URL and expects an answer from the chatbot to be returned.


```
function new_entry() {
    //if the message from the user isn't empty then run
    if (document.getElementById("input").value != "") {
        // pulls the value from the chatbox and sets it to lastUserMessage
        lastUserMessage = document.getElementById("input").value;
        // change the chatbox and display the user message
        // . . .

        // get message from backend
        // send a GET request with the user message to the server using an URL
        $.get('/sendQuestion?question=' + lastUserMessage)
        .then(function(data) {
            botMessage = data
            // change the chatbox and display the bot response
            // . . .

        })
    }
}
```

The new_entry() function sends the user input to the server expecting the response of the chatbot

In Flask, routing is a method in which URLs are assigned to functions so that whenever the user proceed to a URL, the method is called. A route to process the request from the above Javascript file is set up.

```
@app.route('/sendQuestion')
def sendQuestion():
    question = request.args.get("question")
    if not question:
        return "error"

    answer = bot.handle_input(question)

    return jsonify(answer)
```

The sendQuestion() function is called whenever the server has detected a request to the /sendQuestion route

With these functions, the chatbot has been successfully integrated into the website.

2.3. Script and Training Data Creation

2.3.1. Write the chatbot script

In the Python code of the chatbot, it is possible to create a chatbot script (or initial prompt) to define the conversation flow and the chatbot's content. As the chatbot's name is Cookiesier and its objective is to provide cooking instructions, the initial prompt could be set as *"I'm Cookiesier, a cooking assistant. What would you like to cook today?"* and then passed to the `get_chatbot_response()` function as an example for the chatbot to understand the conversation theme and generate a relevant response to the cooking topic. Furthermore, the inputs and generated responses will also be appended to the initial prompt and passed again to the `get_chatbot_response()` function the next time another input is entered. Therefore, the chatbot gains the ability to remember previously stated information in the conversation.

2.3.2. Prepare training dataset

By training models on vast amounts of data, the chatbot can learn to recognize patterns and respond appropriately to a variety of user queries. With the ability to fine-tune models for specific tasks, such as identifying food images and recommending recipes, chatbots can provide customized and personalized experiences for users.

Preparing a high-quality training dataset is crucial when training an OpenAI model. The dataset should be carefully curated to ensure that it accurately relates to the goal and objectives of the chatbot. This involves selecting a diverse range of sources, including scientific articles and existing datasets on the websites like Kaggle, etc. In this report only, the used model is 'davinci', a GPT-3 model, and the training data are a small number of responses to some greeting and introducing inputs. In practice, the data should be preprocessed and cleaned to remove any noise or outliers. Additionally, it is important to ensure that the dataset is large enough to provide sufficient coverage of the relevant concepts.

To train the model, it is necessary to have a collection of training data that contain prompts and their corresponding completions. OpenAI provides a tool to support developers with the training process, the CLI data preparation tool. With this tool, an Excel (or CSV, TSV, JSON, JSONL) file with the prompt and completion columns can be formatted into a JSONL file suitable for the fine-tuning process, by a command line:

```
openai tools fine_tunes.prepare_data -f <LOCAL_FILE>
```

The data formatting of the file is that each prompt should have a fixed separator at the end (`\n\n###\n\n`) and each completion should start with a whitespace, and end with a fixed stop sequence (`###`).

2.3.3. Train the 'davinci' model

After the dataset file is created, enter the following command line on the console to set up the API key:

```
set OPENAI_API_KEY=<YOUR_API_KEY>
```

and the following command to start the training process:

```
openai api fine_tunes.create -t <TRAIN_FILE_ID_OR_PATH> -m davinci
```

with <TRAIN_FILE_ID_OR_PATH> as the created dataset file path. The file will be uploaded to be used across various endpoints/features. The process begins and whenever it is interrupted, enter the following command to continue:

```
openai api fine_tunes.follow -i <PROCESS_ID>
```

with <PROCESS_ID> is the ID of the process shown right after the file is uploaded and the training begins. When the 'Status' appears to be 'succeeded', the model has finished its training and the uploaded model's name can be used in the code to refer to the trained model.

```
[2023-03-07 21:05:35] Created fine-tune: ft-tXh36umKd10nsI6nWvI1C02m
[2023-03-07 21:09:57] Fine-tune costs $0.01
[2023-03-07 21:09:58] Fine-tune enqueued. Queue number: 9
[2023-03-07 21:11:11] Fine-tune is in the queue. Queue number: 8
[2023-03-07 21:11:13] Fine-tune is in the queue. Queue number: 7
[2023-03-07 21:14:46] Fine-tune is in the queue. Queue number: 6
[2023-03-07 21:18:41] Fine-tune is in the queue. Queue number: 5
[2023-03-07 21:21:36] Fine-tune is in the queue. Queue number: 4
[2023-03-07 21:21:37] Fine-tune is in the queue. Queue number: 3
[2023-03-07 21:23:37] Fine-tune is in the queue. Queue number: 2
[2023-03-07 21:24:21] Fine-tune is in the queue. Queue number: 1
[2023-03-07 21:24:31] Fine-tune is in the queue. Queue number: 0
[2023-03-07 21:24:35] Fine-tune started
[2023-03-07 21:27:15] Completed epoch 1/4
[2023-03-07 21:27:16] Completed epoch 2/4
[2023-03-07 21:27:17] Completed epoch 3/4
[2023-03-07 21:27:18] Completed epoch 4/4
[2023-03-07 21:28:08] Uploaded model: davinci:ft-personal-2023-03-07-14-28-06
[2023-03-07 21:28:09] Uploaded result file: file-kdL1Cvf0AVNm2QdWw7KG1CBk
[2023-03-07 21:28:09] Fine-tune succeeded

Job complete! Status: succeeded 🎉
```

Figure 3 – Training complete

2.4. Testing and Debugging

After coding the chatbot using Python and obtaining an API key from OpenAI for the bot, it is time for testing and debugging with some basic greeting and requiring inputs.

2.4.1. Use a no-credit account

The first problem that came up when hitting the Run process was `openai.error.RateLimitError`: You exceeded your current quota, please check your plan and billing details. This happens if the monthly usage limit for the OpenAI API is met or if there is no trial credit left to use. To solve this problem, firstly, check the **Usage** page for the **Free trial usage** to see how many credits are left in the account. The met case is that there is no credit left to use the API of OpenAI. Therefore, another account with available credits should be obtained to continue the research.

2.4.2. Use 'davinci' model

The model used in the chatbot code was the trained 'davinci' model. However, the responses to some basic input are quite odd as shown in Table 2.

Inputs	Outputs	Comment
hi	I'm Cookiesier, a cooking assistant. What do you want me to help you cook? User: hi I'm Cookiesier, a cooking assistant. What do you want me to help you cook? User: hi I'm Cookiesier, a cooking assistant. What do you want me to help you cook? User: hi [...]	The chatbot auto-generated a conversation endlessly and only stopped when the max-token is met.
who are you	I'm Cookiesier, a cooking assistant. What do you want me to help you cook? who are you I'm Cookiesier, a cooking assistant. What do you want me to help you cook? who are you [...]	Chatbot repeated the response with the correct answer exactly like the initial prompt. It auto-generated an endless, repeating conversation that only stops when the max-token is met.
I want to cook pasta	I'm Cookiesier, a cooking assistant. What do you want me to help you cook? I want to cook pasta I'm Cookiesier, a cooking assistant. What do you want me to help you cook? I want to cook pasta [...]	No valid response was provided. The chatbot repeated its initial prompt and the user's question.
Can you instruct me to cook pasta?	I'm Cookiesier, a cooking assistant. What do you want me to help you cook? Can you instruct me to cook pasta? I'm Cookiesier, a cooking assistant. What do you want me to help you cook? Can you instruct me to cook pasta? [...]	The initial prompts and user's questions are repeated.

Table 2 – Inputs and outputs of the trained 'davinci' model

The training process is the source of the problem. The dataset was not large enough to cover every scenario of the chatbot conversation and it did not have the recipe for the required dish. As a result, the chatbot malfunctioned and did not give valid responses to the user's inputs. To fix this problem quickly, the 'text-davinci-003' model is used, a GPT-3.5 model capable of

performing language tasks with superior quality, longer output, and more consistent instruction-following compared to the ‘curie’, ‘babbage’, and ‘ada’ models.

2.4.3. Use ‘text-davinci-003’ with the English initial prompt

After changing to the ‘text-davinci-003’ model and modifying the INITIAL_PROMPTS to "Hi, I'm Cookiesier. What do you want me to help you cook?", the performance of the chatbot has improved both in speed and in accuracy. The inputs and their corresponding outputs are provided in Table 3 below.

Inputs	Outputs	Comment
hi	Hi there! What type of dish would you like to cook?	Right response with a little paraphrasing of the chatbot's initial prompt.
who are you	Hi, I'm Cookiesier. I'm a virtual assistant that helps people cook delicious dishes. What do you want me to help you cook?	Right response with a newly created abilities description.
I want to cook pasta	Great! What type of pasta dish would you like to make?	Right response with a correct follow-up question for the user to specify the kind of pasta to make.
Can you instruct me to cook spaghetti?	<p>Sure! Here are the instructions for making spaghetti:</p> <ol style="list-style-type: none"> 1. Bring a large pot of salted water to a boil. 2. Add the spaghetti to the boiling water and cook according to the package instructions. 3. Drain the spaghetti and return it to the pot. 4. Add your desired sauce and mix until the spaghetti is evenly coated. 5. Serve the spaghetti with your favorite toppings. Enjoy! 	The chatbot gives correct instructions.
chỉ tôi cách chiên trứng	<p>Sure! Here's a simple recipe for fried eggs:</p> <p>Ingredients:</p> <ul style="list-style-type: none"> - 2 eggs - 1 tablespoon of butter - Salt and pepper to taste 	Even though the input was in Vietnamese, the chatbot could understand the request and give the correct instruction. The only problem is that the chatbot

	Instructions: 1. Heat a non-stick pan over medium heat. 2. Add the butter and let it melt. 3. Crack the eggs into the pan and season with salt and pepper. 4. Cook the eggs until the whites are set and the yolks are cooked to your desired doneness. 5. Serve and enjoy!	repeated its name before giving the instruction and the output was written in English instead of Vietnamese.
--	--	--

Table 3 – Inputs and outputs of the ‘text-davinci-003’ model

In conclusion, the ‘text-davinci-003’ has improved the overall performance of the chatbot, making it closer to the desired objectives. However, to solve the problem of providing Vietnamese instructions, the chatbot script needs to be modified to broaden the chatbot capabilities.

2.4.4. Use ‘text-davinci-003’ with the Vietnamese initial prompt

In order to create a bilingual chatbot that can process in Vietnamese. The INITIAL_PROMPT has been changed into "Chào bạn, tôi là Cookiesier. Hôm nay bạn muốn nấu món gì?". Some basic inputs are tested and presented in Table 4 below.

Inputs	Outputs	Comment
Chào bạn	Chào bạn, tôi là Cookiesier. Hôm nay bạn muốn nấu món gì?	Correct response in correct language but exactly like the initial prompt with no diversity.
hello	Chào bạn, tôi là Cookiesier. Hôm nay bạn muốn nấu món gì?	Correct response but in the wrong language.
tôi muốn nấu mì ý	Tuyệt vời! Để bắt đầu, bạn cần các nguyên liệu như sau: 500g mì ý, 1 quả cà chua, ½ muỗng canh dầu, 1 muỗng canh hành tây, 1 muỗng canh nước mắm, 1 muỗng canh nước tương, 1 muỗng canh nước tỏi, 1 muỗng canh đường, 1 muỗng canh nước ớt, 1 muỗng canh sốt me. Bạn có thể bắt đầu nấu món ngay bây giờ!	Correct response in the right language.
chỉ tôi cách chiên trứng ngon	Để chiên trứng ngon, bạn cần các nguyên liệu sau: 2 trứng gà, 1 muỗng canh dầu, 1 muỗng canh bột mì, 1 muỗng canh nước mắm, 1 muỗng cà phê muối, 1 muỗng cà phê hành lá, 1 muỗng cà phê đường.	Correct recipe for the dish in the right language.

	<p>Bước 1: Trộn các nguyên liệu trên trong một tô.</p> <p>Bước 2: Đổ dầu vào chảo, để nó nóng lên.</p> <p>Bước 3: Đổ trứng vào chảo, đảo lần lượt để trứng không bị bể.</p> <p>Bước 4: Cho thêm 1 muỗng canh nước mắm, 1 muỗng cà phê muối, 1 muỗng cà phê hành lá, 1 muỗng cà phê đường vào chảo.</p> <p>Bước 5: Đảo lần lượt để trứng không bị bể.</p> <p>Bước 6: Khi trứng đã chín, hạ lửa và để trứng ở nhiệt độ thấp.</p> <p>Bước 7: Trứng chiên sẽ được sẵn sàng để ăn.</p>	
I want to cook pasta	<p>Ồ, bạn muốn nấu món pasta. Tôi có thể giúp bạn tìm một công thức nấu pasta ngon nhất. Bạn có thể tham khảo công thức nấu pasta của chúng tôi tại đây: https://cooking.nytimes.com/recipes/1019072-classic-spaghetti-carbonara. Chúc bạn nấu ăn thành công!</p>	Correct response but the link is invalid. The response is not in the wanted language.

Table 4 – Inputs and outputs of the chatbot using the Vietnamese initial prompt

When being asked for recipes, the chatbot still gave correct responses to some extent. Although changing the language of the initial prompt did change the language in the responses of the chatbot, it can not be considered a long-term solution for the bilingual feature. Therefore, a special python library is used to solve the crisis, ‘langdetect’, which will be explained further in Section 3.4.

2.4.5. Finalize English and Vietnamese initial prompt

To make the objectives clearer to the chatbot and apply the bilingual feature, along with using the 'langdetect' library (see Section 3.4), the initial prompts are modified to give clearer instructions on how the chatbot should behave towards different user inputs. The English prompt is changed into "You are Cookiesier, a cooking assistant. If the user greets you, greet back politely and ask for the user's desired food.\n If the user asks for a recipe, provides ingredients and step-by-step instructions." and the Vietnamese prompt is "Bạn là Cookiesier, trợ lý nấu ăn ảo. Nếu người dùng chào bạn, chào lại và hỏi người dùng muốn nấu món gì.\n Nếu người dùng hỏi về công thức cách nấu một món ăn, hãy đưa ra các nguyên liệu và hướng dẫn từng bước." The inputs and outputs after modifying the prompts are presented in Table 5.

Inputs	Outputs	Comment
Chào bạn	Chào bạn, rất vui được làm quen với bạn. Bạn muốn nấu món gì? Tôi có thể cung cấp cho bạn các công thức nấu ăn để giúp bạn.	Correct response in the wanted language.
hello	Hi there! What kind of food would you like to cook today?	Correct response in the wanted language.
chỉ tôi cách chiên trứng ngon	<p>Xin chào! Rất vui được hỗ trợ bạn. Để chiên trứng ngon, bạn cần các nguyên liệu sau:</p> <ul style="list-style-type: none">- Trứng- Bột bánh mì- Gia vị- Muối <p>Các bước thực hiện:</p> <ol style="list-style-type: none">1. Trộn các nguyên liệu trứng, bột bánh mì, gia vị và muối trong một tô.2. Vỗ trứng để trứng bị lỏng.3. Đổ trứng lỏng vào chảo, đảo để trộn đều.	Correct recipe in the wanted language.

	<p>4. Để cho trứng chiên đến khi màu vàng đều.</p> <p>5. Để ra khỏi chảo và để nguội.</p> <p>6. Trứng đã sẵn sàng để thưởng thức!</p>	
I want to make pancakes	<p>Hi there! It's nice to meet you. Making pancakes sounds delicious! Here's what you'll need to make them:</p> <ul style="list-style-type: none"> - 1 1/2 cups all-purpose flour - 3 1/2 teaspoons baking powder - 1 teaspoon salt - 1 tablespoon white sugar - 1 1/4 cups milk - 1 egg - 3 tablespoons butter, melted <p>Instructions:</p> <p>1. In a large bowl, sift together the flour, baking powder, salt and sugar. Make a well in the center and pour in the milk, egg and melted butter; mix until smooth.</p> <p>2. Heat a lightly oiled griddle or frying pan over medium-high heat. Pour or scoop the batter onto the griddle, using approximately 1/4 cup for each pancake. Brown on both sides and serve hot. Enjoy!</p>	Correct recipe in the wanted language.

Table 5 – Inputs and outputs of the chatbot using the modified initial prompts

In conclusion, by giving the chatbot only instructions, not particular examples of responses, the generated outputs become more natural and creative than the outputs provided in Table 6 of Section 3.4, when the prompts were not modified. Therefore, when implementing a chatbot using OpenAI's API, attention must be paid more to the instruction-giving prompts that define the conversation flows and the chatbot's functionality.

III. ADDING CUSTOM FUNCTIONALITIES AND INTEGRATIONS

3.1. Speech-to-text

Speech-to-text, also known as automatic speech recognition (ASR) or speech recognition, is the technology that enables a computer or device to recognize spoken words and convert them into written text. The primary purpose of this feature is to facilitate user input by allowing them to articulate their inquiries through speech.

The implementation of this feature mainly involves the utilization of the **Web Speech API** of JavaScript. The Web Speech API is a browser-based API that enables web developers to integrate speech recognition and speech synthesis (text-to-speech) capabilities into their web applications. It allows users to interact with web applications using their voice, rather than using traditional input methods such as typing on a keyboard or clicking a mouse. The detailed implementation of Web Speech API is presented in the Javascript file below.

```
// create new SpeechRecognition object
const recognition = new (window.SpeechRecognition ||
window.webkitSpeechRecognition)();

// set recognition options
recognition.lang = 'en-US'; // set language
recognition.interimResults = true; // show interim results
recognition.maxAlternatives = 1; // set maximum number of alternatives

let isRecording = false; // flag to track whether the user is speaking
let spokenSentence = ''; // variable to hold the current spoken sentence

// add event listener for when recognition results are available
recognition.onresult = function(event) {
  const result = event.results[0][0].transcript; // get first result
  //console.log('Speech recognition result:', result);
  // update the spoken sentence with the new result
  spokenSentence = result;
};
```

Setting up the speechRecognition object for recognizing speech

The recognition process is controlled by a button that initiates the recording process upon activation and terminates it when deactivated. The recognized queries are subsequently transcribed into written text and displayed in the chatbox.

```
const button = document.getElementById("recButton");
button.onclick = function(event) {
    event.preventDefault()
    if (isRecording) {
        isRecording = false;
        recognition.stop();
        document.getElementById("input").value = spokenSentence;
        newEntry(true);
    }
    else {
        isRecording = true;
        recognition.start();
    }
}
```

The Javascript code for controlling the logic of the recording button

The code initializes a button object and assigns it to a variable named 'button' using the `getElementById()` method. A function is then assigned to the `onclick` event of the button. This function first calls the `preventDefault()` method of the event object to cancel the event if it is cancelable, preventing any default action from taking place. The function then checks whether the 'isRecording' variable is true or false. If true, the recording process is stopped by calling the `stop()` method of the 'recognition' object. The recognized spoken sentence is then assigned to the value of the 'input' element, and a new entry is created. If 'isRecording' is false, the recording process is started by calling the `start()` method of the 'recognition' object.

3.2. Text-to-speech

Text-to-speech technology involves the conversion of written text into synthesized speech, employing algorithms to produce speech that sounds natural. This technology accepts text as input and generates speech that can be played back through the speakers of a computer or other device. Combining the chatbot with this functionality can make it possible for the chatbot to provide cooking assistance via audibly relaying information regarding ingredients and cooking procedures.

Like its counterpart, the speech-to-text function, this feature also takes advantage of the Web Speech API of JavaScript. The process of generating speech from text is straightforward as the following function:

```
//text to Speech
function Speech(say) {
    if ('speechSynthesis' in window) {
        var utterance = new SpeechSynthesisUtterance(say);
```

```

        speechSynthesis.speak(utterance);
    }
}

```

The Speech() function that produces the speech corresponds to the text 'say'

The above code defines a JavaScript function called Speech() that takes a string as input, which represents the text that should be spoken. The function checks if the web speech API is available in the current window using the speechSynthesis property. If the API is available, the function creates a new SpeechSynthesisUtterance object with the input text and passes it to the speak() method of the speechSynthesis interface, which then speaks the given text using the default speech synthesis voice of the user's browser.

3.3. Food recognition

Food recognition refers to the process of identifying and classifying food items based on their visual characteristics, such as shape, color, texture, and size. This technology typically involves the use of computer vision algorithms and machine learning techniques to analyze images of food and determine their type, ingredients, and nutritional content.

Due to the lack of expertise and the highly demanding amount of time and resources for the training process, a model developed by Avinash Kappa in his "Multiclass Classification using Keras and TensorFlow 2.0 on Food-101 Dataset" Google collab notebook (Kappa, n.d.) is used as a basis for a simple demonstration of this functionality. This model enables the recognition of 11 different food types, chosen randomly, out of the 101 food types available in the Food-101 dataset.

The notebook used by Kappa to train his model was employed, then the configuration and the weights of the model were saved in an 'hdf5' file and an 'h5' file. When the server is initialized, the model is ready for any inputs.

```

best_model = load_model("best_model_11class.hdf5")
best_model.load_weights("weight_trained_11class.h5")

```

Loading the model, loading its weights so it can be used to predict the image of the food

```

def predict_class(model=best_model, images = [], show=True):
    for img in images:
        # turning the image into an array for the model.predict() function
        img = tf.keras.utils.load_img(img, target_size=(299, 299))

        img = tf.keras.utils.img_to_array(img)
        img = np.expand_dims(img, axis=0)
        img /= 255.

        pred = model.predict(img)

```



```

index = np.argmax(pred)
food_list.sort()
pred_value = food_list[index]
# print(pred_value)
return pred_value

```

The predict_class() returns the label of the input image

When the user submits an image to the server, thanks to routing feature of Flask, the trained model attempts to recognize the food in the image, and subsequently returns the label of the recognized food item.

```

@app.route('/uploadFile', methods = ['GET', 'POST'])
def uploadFile():
    if request.method != 'POST':
        return 'no'
    if ('file' not in request.files):
        return 'no file'

    file = request.files.get('file')
    if file.filename == '':
        return('No selected file')
    if file:
        path = os.path.join(app.config['UPLOAD_FOLDER'], file.filename)
        path = './images/' + file.filename
        file.save(path)
        return jsonify(predict_class(images = [path]))
    return "something failed"

```

The uploadFile() function is called whenever the server has detected a request to the /uploadFile route, returning the label of the food detected

```

$.ajax({
    type: 'POST',
    url: '/uploadFile',
    data: form_data,
    contentType: false,
    cache: false,
    processData: false,
    success: function(food) {
        // get chatbot message from backend based on the food
        // update the chatbox
        // ...
    },
});

```

3.4. Bilingual text communication

As stated in the objectives of the chatbot and the crisis of using the Vietnamese initial prompt, the ‘langdetect’ library can help implement the bilingual feature. The Python ‘langdetect’ is a fast and easy-to-use library that can automatically detect the language of a text based on character n-grams and language models. It supports over 55 languages and can detect the language with high accuracy, even for short text samples. This library is useful for a wide range of applications such as text classification, machine translation, and content filtering.

To download the library, type the following command into the command line:

```
pip install langdetect
```

After the package is fully downloaded and imported into the chatbot code, detect() method can be used to identify an input’s language by placing the following code at the beginning of the code file and in the body of the function used to fetch responses from the ChatGPT API:

```
from langdetect import detect

# Chatbot initialize
EN_PROMPT = "Hi, I'm Cookiesier. What would you like to cook today?"
VN_PROMPT = "Xin chào, tôi là Cookiesier. Hôm nay bạn muốn nấu món gì?"
INITIAL_PROMPT = EN_PROMPT
_conversation_history = INITIAL_PROMPT + "\n"

#Inside the function that fetch the chatbot's response from input string and
display result
if (detect(input_str) == "vi"):
    if (INITIAL_PROMPT == EN_PROMPT):
        INITIAL_PROMPT = VN_PROMPT
        _conversation_history = INITIAL_PROMPT + "\n"
else:
    if (INITIAL_PROMPT == VN_PROMPT):
        INITIAL_PROMPT = EN_PROMPT
        _conversation_history = INITIAL_PROMPT + "\n"
```

Code for the bilingual features

When passing the input string as a parameter of the function, the language of the input is detected and then compared to the current language used. If the new language is different from the current language of the chatbot, INITIAL_PROMPT will be changed accordingly into a Vietnamese prompt or the English one. _conversation_history is an appended prompt that will save the inputs and outputs of the chatbot from time to time to improve the chatbot’s performance. The inputs and outputs of the modified program are presented in Table 5.

Inputs	Outputs	Comments
Hello	Hi, I'm Cookiesier. What would you like to cook today?	Chatbot responds correctly to the user greeting input in the correct language.
Chào bạn	Chào bạn, tôi là Cookiesier. Hôm nay bạn muốn nấu món gì?	Chatbot responds correctly in the wanted language.
Show me how to make pasta	<p>Making pasta is easy! Here are the steps:</p> <ol style="list-style-type: none"> 1. Bring a large pot of salted water to a boil. 2. Add the pasta to the boiling water and stir occasionally. 3. Cook the pasta until it is al dente, or slightly firm to the bite. 4. Drain the pasta in a colander and rinse with cold water. 5. Serve the pasta with your favorite sauce or topping. Enjoy! 	Chatbot provides correct instructions in the correct language based on the user's request.
Chỉ tôi cách chiên trứng ngon	<p>Được rồi. Đây là cách làm chiên trứng ngon nhất:</p> <p>Bước 1: Chuẩn bị nguyên liệu:</p> <ul style="list-style-type: none"> - Trứng gà: 2 quả - Muối: 1/2 muỗng cà phê - Bột mì: 1 muỗng cà phê - Dầu ăn: 1 muỗng cà phê <p>Bước 2: Trộn nguyên liệu</p> <ul style="list-style-type: none"> - Đầu tiên, trộn trứng gà, muối, bột mì và dầu ăn lại với nhau trong một tô. - Trộn đều để tạo thành hỗn hợp trứng. <p>Bước 3: Chiên trứng</p> <ul style="list-style-type: none"> - Đầu tiên, đổ hỗn hợp trứng vào chảo nóng với một lượng dầu ăn. - Đảo trứng để chiên đều. - Khi trứng đã chiên đều, lấy ra và đặt lên bát. <p>Bước 4: Phục vụ</p> <ul style="list-style-type: none"> - Sau khi chiên xong, bạn có thể phục vụ trứng với một lát mì hoặc một bát rau xanh. <p>Chúc bạn thưởng thức ngon miệng!</p>	Chatbot provides correct instructions in the correct language based on the user's request.

Table 6 – Inputs and outputs of the bilingual chatbot (before modifying the prompts in section 2.4.5)

In conclusion, by using the 'langdetect' library, the bilingual feature is fulfilled.

3.5. Image generation based on food names

Users sometimes wonder what a dish would look like when hearing or seeing its name on a newspaper or radio. Therefore, by using the DALL-E model, an image of that dish will be generated and given out to users to acknowledge them. About DALL-E, it is a neural network-based image generation model developed by OpenAI. The model uses the GPT-3 architecture and is trained to generate high-quality and unique images from textual descriptions. The code for the image generation is provided as follows.

```
def generate_image(prompt,size="512x512"):
    response = openai.Image.create(
        prompt=prompt,
        n = 1,
        size=size
    )

    # Get the image URL from the API response
    image_url = response['data'][0]['url']

    return image_url
```

The Javascript code for the image generation function

This JavaScript code checks if the last message received from the user starts with "/gen". If it does, it sends a request to the server with the text following "/gen" as a query parameter. Once the request completes, it loads an image to the chatbox. The process is coded in the following Javascript code snippet.

```
if (lastUserMessage.startsWith('/gen')) {
    var generateText = lastUserMessage.substring(lastUserMessage.indexOf(' '),
lastUserMessage.length);
    $.get('/sendGenerate?q=' + generateText)
    .then(function(image) {
        // load the image
    })
    return;
}
```

The Javascript code for checking the message structure, and sending a GET request to the server

In Flask, set up a route to the sendGenerate() function that is triggered when the user requests the URL "/sendGenerate". The function gets the name of the food from the argument of the request and generates the image based on that name. Finally, return the URL of the generated image to the user. Note that it is not necessary to send the file as rendering the image in the chatbox using HTML only requires its URL.

```
@app.route('/sendGenerate')
def sendGenerate():
    text = request.args.get("q")
    if not text:
        return "error"

    imageURL = imgProcess.generate_image(text)
    # image.show()
    return jsonify({'image_url': imageURL})
```

The Python code for the Flask route

V. CONCLUSION

Overall, the main goal of the chatbot, which is to provide users with recommendations and detailed instructions on cooking meals based on the leftover ingredients or their requests, is acquired. Some advanced features such as speech recognition, speech synthesis, and image recognition are also successfully integrated into the chatbot. However, the chatbot also has some limitations.

Firstly, the chatbot platform, a self-built website, currently can only be hosted on a local machine. Future versions of the chatbot platform should preferably be on a domain that a lot of people can have access to. However, this approach can be quite costly due to the required expenditure for the domain name and credits for OpenAI's API. Whether the chatbot can be continued in this direction or not largely depends on the demand for the chatbot.

The second limitation is that the speech synthesis and recognition feature only supports the English voice. The Web Speech API that was used to implement these features doesn't support Vietnamese. However, it is possible to replace the API with many other speech synthesis and recognition APIs that support Vietnamese, such as Whisper API, a speech-to-text model by OpenAI or Google's Text-to-speech API. While this solution might increase the total cost of the chatbot since some of the API requires credits, the accessibility it provides to many users should be taken into consideration.

The next limitation of the chatbot is its inability to produce correct recipes when given prompts relating to recipe generation. It might be due to the lack of fine-tuning being done on the chatbot. Overcoming this obstacle will be challenging as fine-tuning can take much effort to create the training data containing lots of food recipes. Nevertheless, this development direction is important for the chatbot as outputting wrong recipes can be dangerous to unsuspecting users.

Another relevant problem of the chatbot is its food recognition capability. Currently, the chatbot can only recognize a list of 11 foods that were used to train the image recognition model. A simple solution is to train the model on more types of food, though it can take quite a lot of time.

The final limitation of the chatbot is regarding its image generation features. In the current version, the images that it creates sometimes contain anomalies and some even do not make sense. This seems to be a problem with DALL-E, the image generation model that the chatbot uses. In future versions, training a model that specialized in food image generation might be a good step for this feature.

In conclusion, with the help of ChatGPT's API and a number of tools, a recipe-generating chatbot with useful functionalities has been successfully created. Despite the many flaws, the chatbot is still in its infancy stage and still has lots of potentials. With the proposed solutions to the problems, hopefully, the chatbot will develop into something great that many people around the world find helpful.

VI. SOURCE CODE INSTRUCTIONS

Firstly, there are some requirements to run the source code:

- Python version 3.8.7
- Operating system: WINDOWS
- 'virtualenv' package, a tool to create isolated Python environments. If it is not previously installed, run the following command in the console: `pip install virtualenv`

Then, inside the folder in which all of the source code files are located, run "`virtualenv your_env_name`" command in the console to create a `your_env_name` folder. After the environment is created, activate the environment by running "`.\your_env_name\Scripts\activate.bat`". To exit the virtual environment later on, run "`deactivate`" in the console. Next, inside the virtual environment, run "`pip install -r ./requirements.txt`" to install all necessary modules included in the `requirements.txt` file.

This project does not contain the APIKEY to use OpenAI's API so create a `.env` file in the "bot" folder that contains "`OPENAI_API_KEY=<YOUR_OWN_API_KEY>`".

To host the website on your local host, run `app.py` file in the command line.

VII. REFERENCES

1. Hamilton, A. C. (n.d.). How to create an advanced chatbot.
2. Kappa, A. (n.d.). *Multiclass Classification using Keras and TensorFlow 2.0 on Food-101 Dataset*. Retrieved from <https://colab.research.google.com/drive/1Z5-LqIJxNslHcCzM9jChsNhGi9Y7HSC4>
3. OpenAI. (n.d.). *Fine-tuning Documentation*. Retrieved from <https://platform.openai.com/docs/guides/fine-tuning>
4. OpenAI. (n.d.). *Text completion Documentation*. Retrieved from <https://platform.openai.com/docs/guides/completion>