

Use ChatGPT to build a web-based system that can answer questions about a website

Special Topics: Generative AI-Driven Intelligent Apps Development

Ademilton Marcelo da Cruz Nunes (19679)

Links

Github:

[https://github.com/ademiltonnunes/Machine-Learning/tree/main/ChatGPT/Use%20ChatGPT%20to%20create%20customer%20support%20website%20\(data%20source%20web%20pages\)](https://github.com/ademiltonnunes/Machine-Learning/tree/main/ChatGPT/Use%20ChatGPT%20to%20create%20customer%20support%20website%20(data%20source%20web%20pages))

Table of Content

- Introduction
- Tutorial Analysis
- Tutorial Analysis - Setting up a web crawler
- Tutorial Analysis - Building embedding indexes
- Tutorial Analysis - Building a question-answer system with embedding indexes
- Jupyter Notebook
- Python on Ubuntu
- Python Flask - Quick-Start
- Python Flask
- Node.js - Quick-Start
- Node.js
- Conclusion

Introduction

This project aims to implement a customer support system using ChatGPT to build a web-based system that can answer questions about a website. We are using as a basis the tutorial available on the open website: OpenAI document - [Website Q&A with Embeddings](#).

In order to increase learning, I will be documenting the development process in a few steps, which will be the development of the system on Jupyter Notebook, Python on Ubuntu, Web Service based in Python Flask and Node.js.

Tutorial Analysis

The tutorial [Website Q&A with Embeddings](#) to build an web-based system that can answer questions about your website is separated into three sections:

- Setting up a web crawler
- Building embedding indexes
- Building a question-answer system with embedding indexes

Tutorial Analysis - Setting up a web crawler

The data acquisition process involves crawling a website to extract information in text format. Specifically, the project focuses on the sfbu.edu website. The crawling mechanism retrieves text from the homepage and systematically explores all links and pages within the website. Subsequently, the gathered data is indexed and organized into embedding indexes for further utilization.

Tutorial Analysis - Building embedding indexes

Utilizing the crawled data results, a tokenization process is initiated, breaking down sentences and words. This procedure standardizes input sizes in preparation for the subsequent embedding process. Subsequently, the text is segmented into smaller chunks, and a straightforward request is dispatched to the OpenAI API, specifically indicating the adoption of the text-embedding-ada-002 model to generate embeddings.

The generated embeddings are then indexed, enabling ChatGPT to leverage this data for responding to questions.

Tutorial Analysis - Building a question-answer system with embedding indexes

Once the embeddings are prepared, the system becomes primed to respond to straightforward questions. By comparing existing embedding indexes, the system effectively retrieves answers for users.

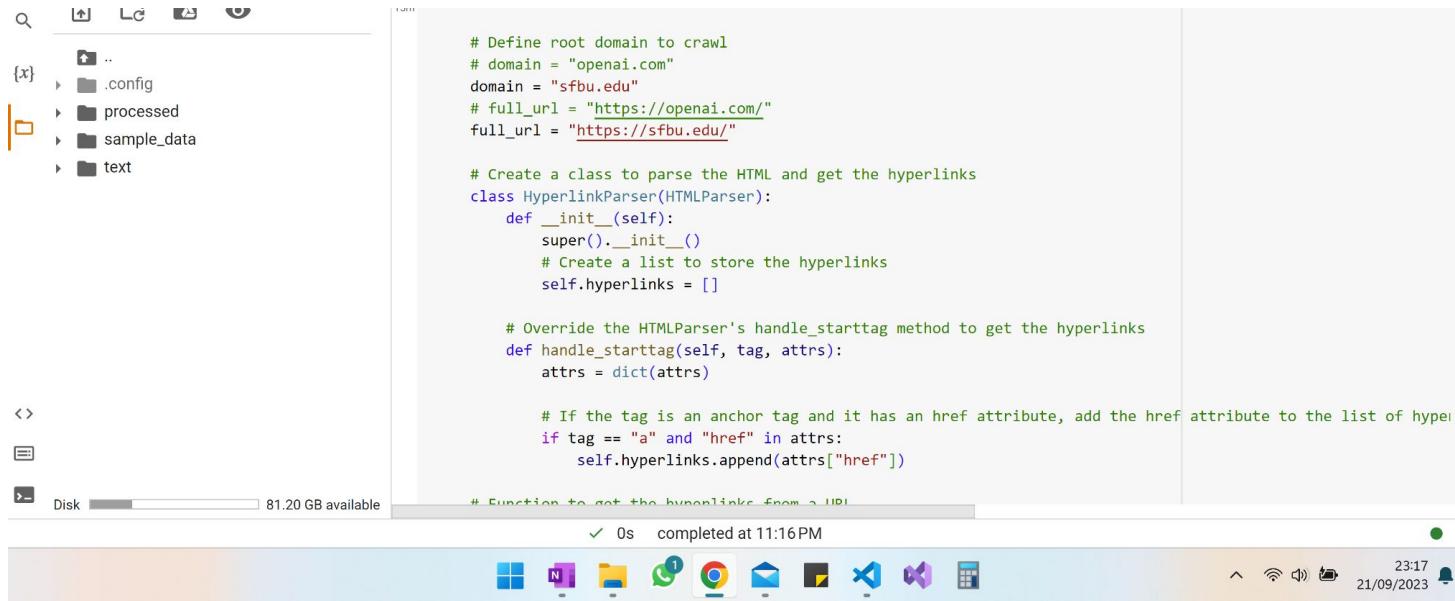
The chosen model for this project is the gpt-3.5-turbo-instruct, which generates responses in a natural and coherent manner, leveraging the information retrieved from the embeddings.

Jupyter Notebook

First step to developing the OpenAI document - Website Q&A with Embeddings tutorial where I implement a customer support system using ChatGPT to build a web-based system that can answer questions about a website is using executing it by a **Jupyter Notebook**. Next slides, it will show details of the code execution:

Jupyter Notebook

The image below shows the crawling process where I used the sfbu.edu website



A screenshot of a Jupyter Notebook interface. On the left, there's a file tree with a folder named 'x' containing '.config', 'processed', 'sample_data', and 'text'. The main notebook area contains the following Python code:

```
# Define root domain to crawl
# domain = "openai.com"
domain = "sfbu.edu"
# full_url = "https://openai.com/"
full_url = "https://sfbu.edu/"

# Create a class to parse the HTML and get the hyperlinks
class HyperlinkParser(HTMLParser):
    def __init__(self):
        super().__init__()
        # Create a list to store the hyperlinks
        self.hyperlinks = []

    # Override the HTMLParser's handle_starttag method to get the hyperlinks
    def handle_starttag(self, tag, attrs):
        attrs = dict(attrs)

        # If the tag is an anchor tag and it has an href attribute, add the href attribute to the list of hyperlinks
        if tag == "a" and "href" in attrs:
            self.hyperlinks.append(attrs["href"])

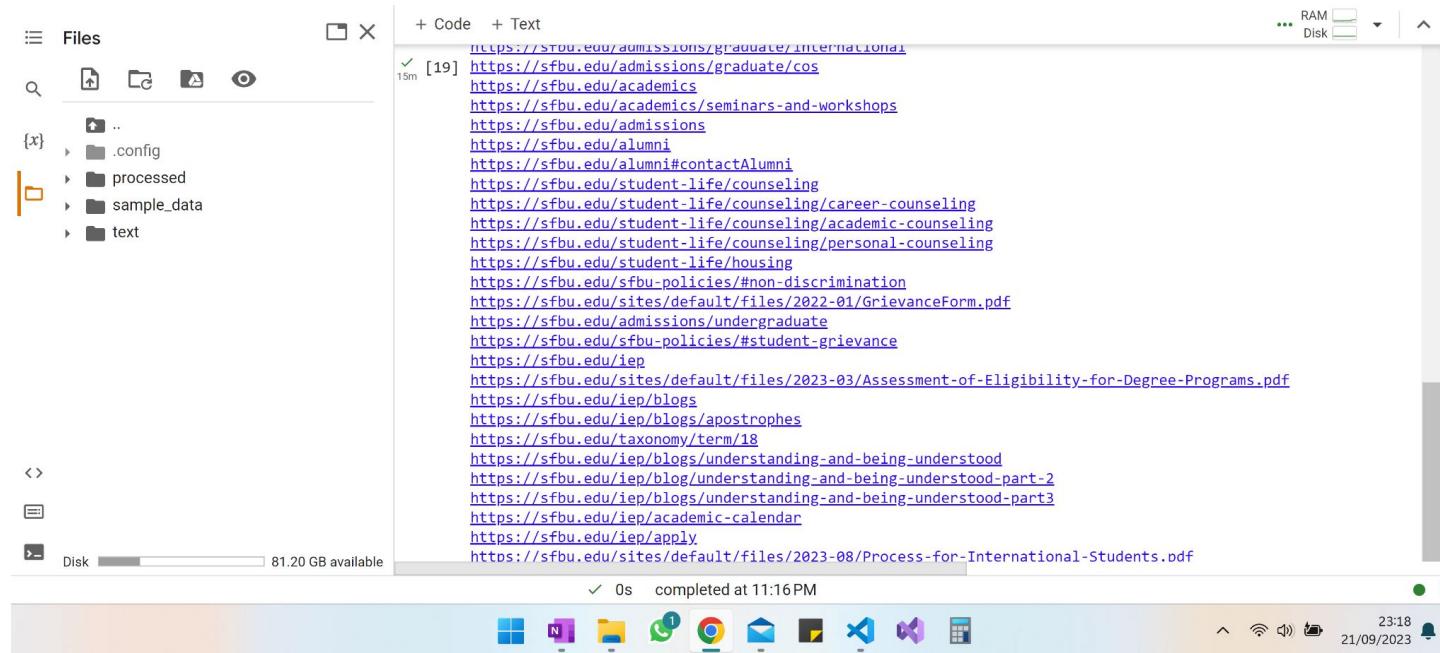
# Function to get the hyperlinks from a URL

```

The status bar at the bottom shows '0s completed at 11:16 PM'. The system tray at the bottom right includes icons for signal strength, battery, date (21/09/2023), and time (23:17).

Jupyter Notebook

During the crawling process, other links and urls on the website are visited in addition to the homepage



The screenshot shows a Jupyter Notebook interface with a sidebar on the left containing a 'Files' section with sub-directories like '.config', 'processed', 'sample_data', and 'text'. The main area displays a list of URLs under a cell labeled '[19]'. The URLs are all from the domain <https://sfbu.edu> and include various pages such as admissions, academic programs, student life, and policies. The status bar at the bottom indicates the task was completed at 11:16 PM.

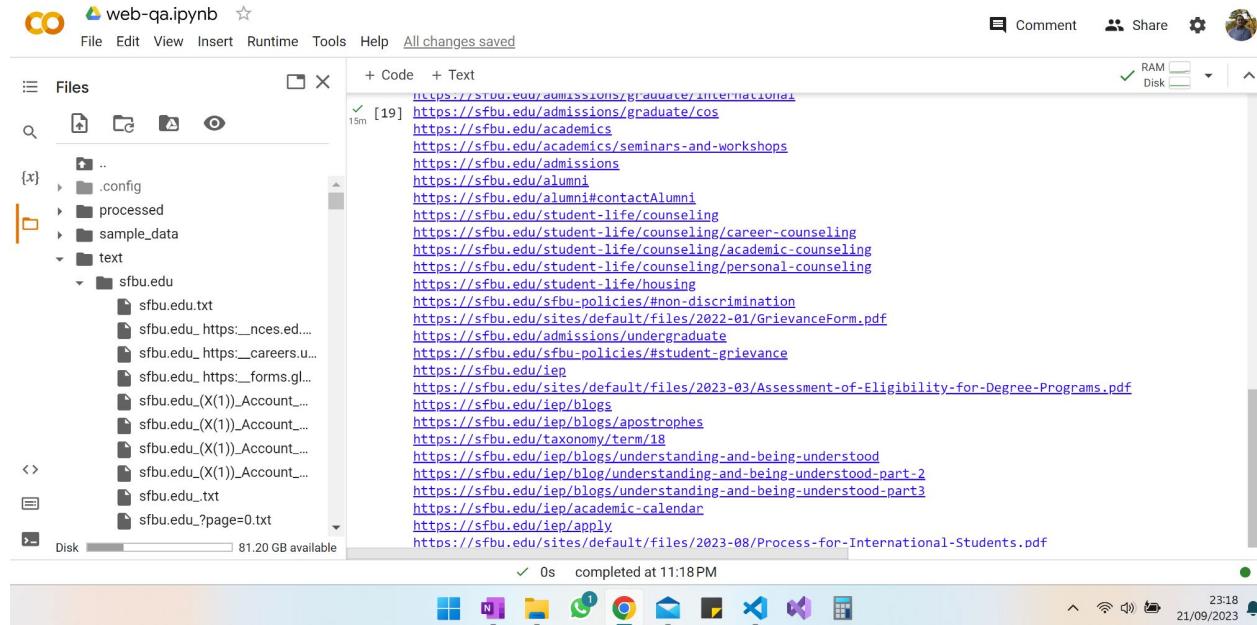
```
+ Code + Text  
15m [19] https://sfbu.edu/admissions/graduate/international  
https://sfbu.edu/admissions/graduate/cos  
https://sfbu.edu/academics  
https://sfbu.edu/seminars-and-workshops  
https://sfbu.edu/admissions  
https://sfbu.edu/alumni  
https://sfbu.edu/alumni#contactAlumni  
https://sfbu.edu/student-life/counseling  
https://sfbu.edu/student-life/counseling/career-counseling  
https://sfbu.edu/student-life/counseling/academic-counseling  
https://sfbu.edu/student-life/counseling/personal-counseling  
https://sfbu.edu/student-life/housing  
https://sfbu.edu/sfbu-policies/#non-discrimination  
https://sfbu.edu/sites/default/files/2022-01/GrievanceForm.pdf  
https://sfbu.edu/admissions/undergraduate  
https://sfbu.edu/sfbu-policies/#student-grievance  
https://sfbu.edu/iep  
https://sfbu.edu/sites/default/files/2023-03/Assessment-of-Eligibility-for-Degree-Programs.pdf  
https://sfbu.edu/iep/blogs  
https://sfbu.edu/iep/blogs/apostrophes  
https://sfbu.edu/taxonomy/term/18  
https://sfbu.edu/iep/blogs/understanding-and-being-understood  
https://sfbu.edu/iep/blog/understanding-and-being-understood-part-2  
https://sfbu.edu/iep/blogs/understanding-and-being-understood-part3  
https://sfbu.edu/iep/academic-calendar  
https://sfbu.edu/iep/apply  
https://sfbu.edu/sites/default/files/2023-08/Process-for-International-Students.pdf
```

✓ 0s completed at 11:16 PM

RAM Disk 23:18 21/09/2023

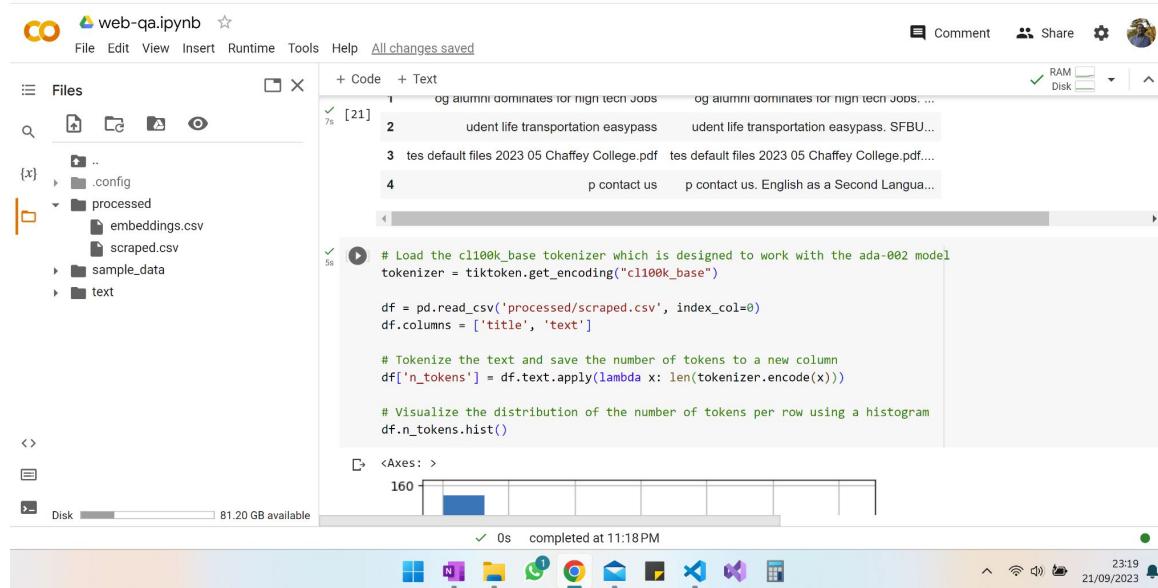
Jupyter Notebook

After visiting links, the texts are saved in txt files



Jupyter Notebook

After this, the tokenization process is carried out on embedding indexes



The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** web-qa.ipynb
- File Explorer:** Shows a directory structure with folders like {x}, processed, sample_data, and text, and files embeddings.csv and scraped.csv.
- Code Cell:** Contains Python code for tokenization and visualization:

```
# Load the cl100k_base tokenizer which is designed to work with the ada-002 model
tokenizer = tiktoken.get_encoding("cl100k_base")

df = pd.read_csv('processed/scraped.csv', index_col=0)
df.columns = ['title', 'text']

# Tokenize the text and save the number of tokens to a new column
df['_tokens'] = df.text.apply(lambda x: len(tokenizer.encode(x)))

# Visualize the distribution of the number of tokens per row using a histogram
df.n_tokens.hist()
```
- Output Cell:** Displays a histogram titled "Axes: >" with the x-axis labeled "0s completed at 11:18PM". The histogram has a maximum value of 160, with several bars visible across the range.
- System Status:** Shows RAM and Disk usage, with RAM being checked.
- Taskbar:** Includes icons for various applications like File Explorer, Task View, and Microsoft Edge.
- System Tray:** Shows the date (21/09/2023), time (23:19), and battery status.

Jupyter Notebook

After the crawling and embeddings process, the system is ready to answer questions related to the website used. If the system does not know the answer, it says: "I don't know":

Jupyter Notebook

Python on Ubuntu

The Jupyter notebook content was exported in Python code. For this to work it was necessary to install some modules in Python. These modules are:

- pip install beautifulsoup4
- pip install tiktoken
- pip install openai
- pip install pandas
- pip install matplotlib
- pip install plotly
- pip install scipy
- pip install scikit-learn

Python on Ubuntu

With the Python code, this was refactored to a more object-oriented pattern. In this format, the code will be more adapted to future changes. Classes were created with their independent methods, as you can see in the next image:

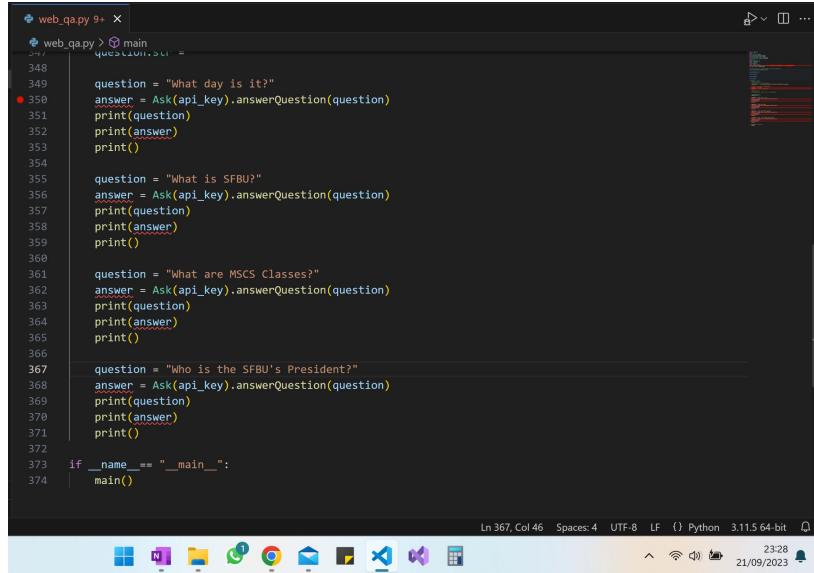
Python on Ubuntu

```
web_qa.py 9+ X
web_qa.py > main
1 import requests
2 import re
3 import urllib.request
4 from bs4 import BeautifulSoup
5 from collections import deque
6 from html.parser import HTMLParser
7 from urllib.parse import urlparse
8 import os
9 import pandas as pd
10 import tiktoken
11 import openai
12 import numpy as np
13 from openai.embeddings_utils import distances_from_embeddings, cosine_similarity
14 from ast import literal_eval
15
16 # Create a class to parse the HTML and get the hyperlinks
17 > class HyperlinkParser(HTMLParser): ...
30
31 > class HyperLink(): ...
91
92 > class Crawl(): ...
146
147 > class Embed(): ...
260
261 > class Ask(): ...
333
334 def main() -> None:
335     # Define API Key
336     # api_key:str = '<Your API Key>'
```

Python on Ubuntu

An initial test was carried out in the console, asking the same questions asked in the Jupyter notebook, so that answers and behavior could be compared.

Python on Ubuntu



A screenshot of a dark-themed code editor window titled "web_qa.py 9+". The code is a Python script that uses an API key to ask four questions: "What day is it?", "What is SFBU?", "What are MSCS Classes?", and "Who is the SFBU's President?". It prints the responses from the API. The code editor shows syntax highlighting and line numbers from 348 to 374. The status bar at the bottom indicates "Ln 367, Col 46" and "Python 3.11.5 64-bit".

```
web_qa.py 9+
web_qa.py > main
question = "What day is it?"
answer = Ask(api_key).answerQuestion(question)
print(question)
print(answer)
print()

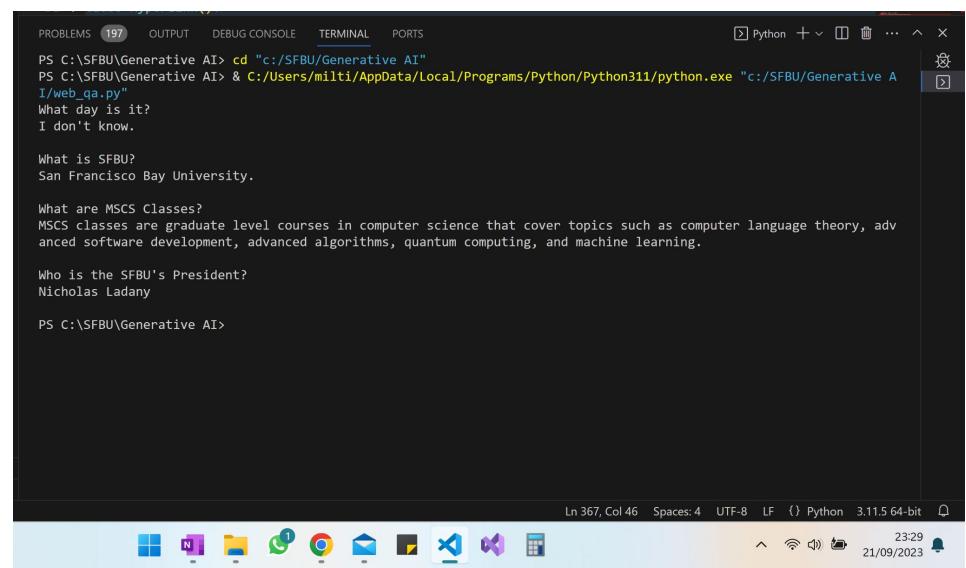
question = "What is SFBU?"
answer = Ask(api_key).answerQuestion(question)
print(question)
print(answer)
print()

question = "What are MSCS Classes?"
answer = Ask(api_key).answerQuestion(question)
print(question)
print(answer)
print()

question = "Who is the SFBU's President?"
answer = Ask(api_key).answerQuestion(question)
print(question)
print(answer)
print()

if __name__ == "__main__":
    main()

Ln 367, Col 46  Spaces:4  UTF-8  LF  () Python 3.11.5 64-bit  23:28  21/09/2023
```



A screenshot of a terminal window titled "Terminal" showing the execution of the Python script "web_qa.py". The terminal shows the command "cd "c:/SFBU/Generative AI"" and the file "I/web_qa.py". The script asks four questions and prints their answers. The terminal also shows the user's name "Nicholas Ladany". The status bar at the bottom indicates "Ln 367, Col 46" and "Python 3.11.5 64-bit".

```
PROBLEMS 197 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\SFBU\Generative AI> cd "c:/SFBU/Generative AI"
PS C:\SFBU\Generative AI> & C:/Users/milti/AppData/Local/Programs/Python/Python311/python.exe "c:/SFBU/Generative AI/web_qa.py"
What day is it?
I don't know.

What is SFBU?
San Francisco Bay University.

What are MSCS Classes?
MSCS classes are graduate level courses in computer science that cover topics such as computer language theory, advanced software development, advanced algorithms, quantum computing, and machine learning.

Who is the SFBU's President?
Nicholas Ladany

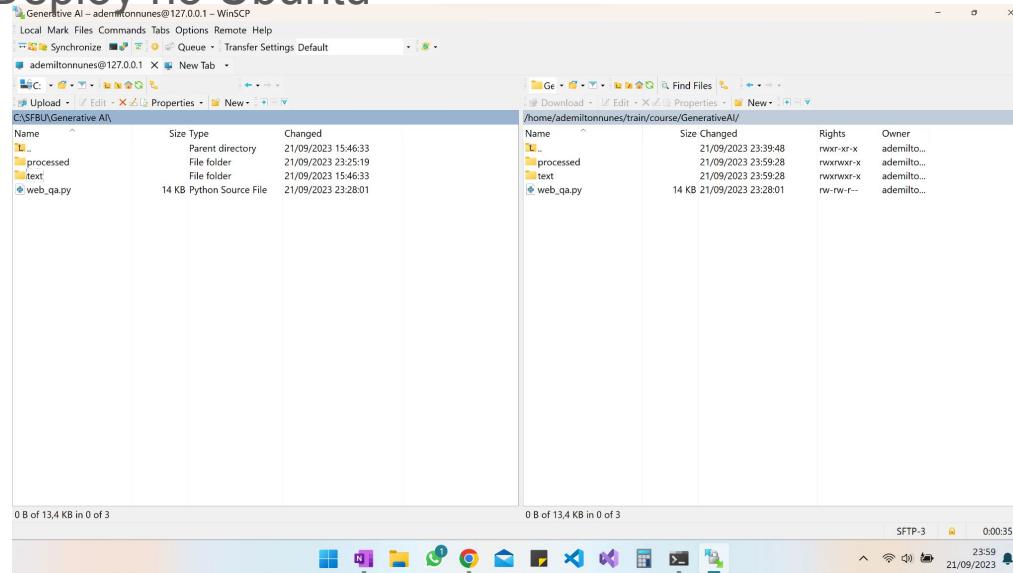
PS C:\SFBU\Generative AI>

Ln 367, Col 46  Spaces:4  UTF-8  LF  () Python 3.11.5 64-bit  23:29  21/09/2023
```

Python on Ubuntu

Running the project on an Ubuntu server, the same questions were asked to carry out the test.

Deploy no Ubuntu



Python on Ubuntu

```
ademilthonnunes@Ademilton- ~ /train/course/GenerativeAI
$ python3 web_qa.py
What day is it?
I don't know.

What is SFBU?
San Francisco Bay University.

What are MSCS Classes?
MSCS classes are graduate level courses in computer science that
cover topics such as computer language theory, advanced software
development, advanced algorithms, quantum computing, and machine
learning.

Who is the SFBU's President?
Nicholas Ladany

ademilthonnunes@Ademilton-Marcelos-PC:~/train/course/GenerativeAI
$ |
```



Python Flask - Quick-Start

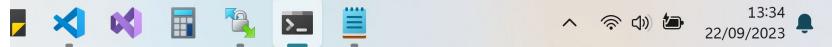
To transform this project into a web-based system (web service), I initially acquired an [OpenAI-quickstart Python Flask tutorial](#). This tutorial served as a practical example of utilizing Python Flask with ChatGPT to suggest pet names. By employing this Python Flask Quick-Start guide, I gained insights into constructing a website and subsequently adapted our project into a web-based system capable of answering questions about a website using Flask.

To implement the Python Flask quickstart project, it is imperative to download the corresponding repository and establish a development environment.

Python Flask - Quick-Start



```
ademiltnunes@Ademilton-  X  Windows PowerShell  X  +  v
ademiltnunes@Ademilton-Marcelos-PC:~/train/course/GenerativeAI$ mkdir quickstart_python|
```

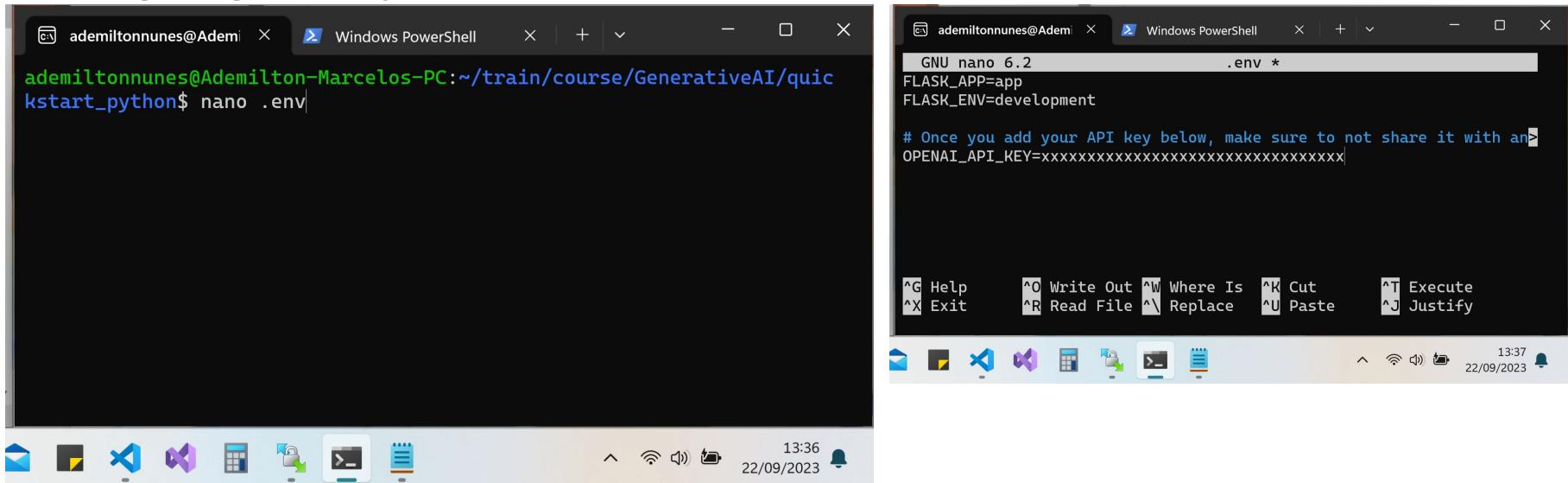


```
ademiltnunes@Ademilton-  X  Windows PowerShell  X  +  v  -  □  ×
ademiltnunes@Ademilton-Marcelos-PC:~/train/course/GenerativeAI$ git clone https://github.com/openai/openai-quickstart-python.git
```

Downloading the tutorial project

Python Flask - Quick-Start

Configuring API keys



The image shows a Windows desktop environment with two terminal windows and a taskbar.

Terminal Window 1 (Left): A Windows PowerShell window titled "ademiltonnunes@Adem". The command entered is:

```
ademiltonnunes@Ademilton-Marcelos-PC:~/train/course/GenerativeAI/quic  
kstart_python$ nano .env
```

Terminal Window 2 (Right): A Windows PowerShell window titled "ademiltonnunes@Adem". The command entered is:

```
GNU nano 6.2 .env *
```

```
FLASK_APP=app  
FLASK_ENV=development
```

```
# Once you add your API key below, make sure to not share it with an  
OPENAI_API_KEY=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx|
```

Taskbar Icons: The taskbar includes icons for Mail, File Explorer, Visual Studio Code, Task View, Task Manager, Calculator, File History, Task Scheduler, and Task Sequence Editor. The system tray shows the date (22/09/2023), time (13:36), battery status, signal strength, and a notification bell.

Python Flask - Quick-Start

Creating and activating the development environment.



ademiltnunes@Ademilton-Marcelos-PC:~/train/course/GenerativeAI/quic
kstart_python\$ sudo apt install python3.10-venv
[sudo] password for ademiltnunes:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
 python3-pip-whl python3-setuptools-whl
The following NEW packages will be installed:
 python3-pip-whl python3-setuptools-whl python3.10-venv
0 upgraded, 3 newly installed, 0 to remove and 61 not upgraded.
Need to get 2473 kB of archives.
After this operation, 2882 kB of additional disk space will be used.
Do you want to continue? [Y/n] |

13:45 22/09/2023

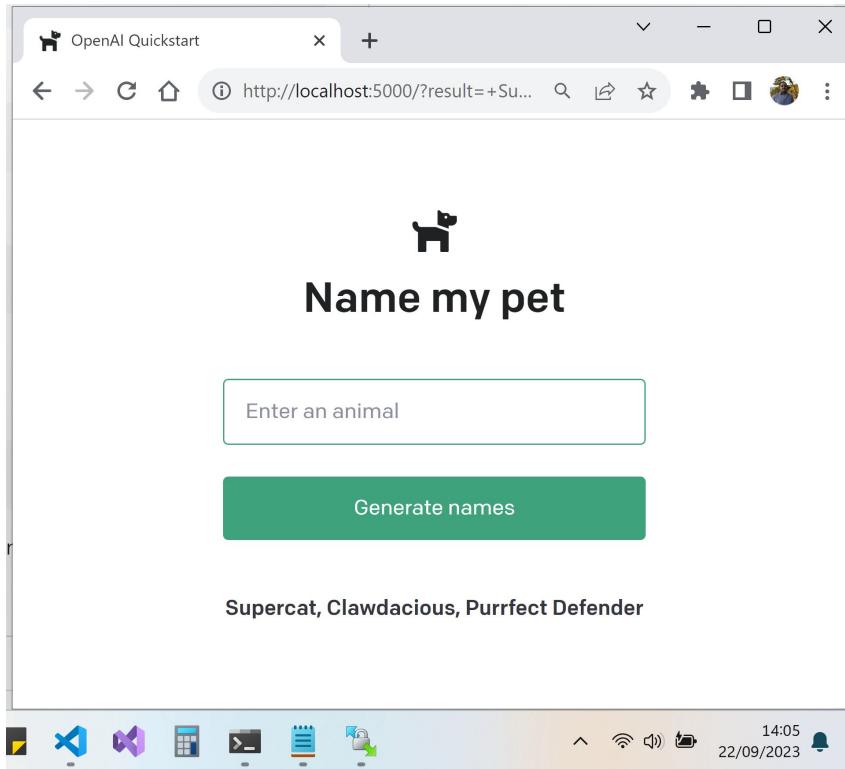


ademiltnunes@Ademilton-Marcelos-PC:~/train/course/GenerativeAI/quic
kstart_python\$ python3 -m venv venv
ademiltnunes@Ademilton-Marcelos-PC:~/train/course/GenerativeAI/quic
kstart_python\$. venv/bin/activate
(venv) ademiltnunes@Ademilton-Marcelos-PC:~/train/course/GenerativeAI/quic
kstart_python\$ |

13:47 22/09/2023

Python Flask - Quick-Start

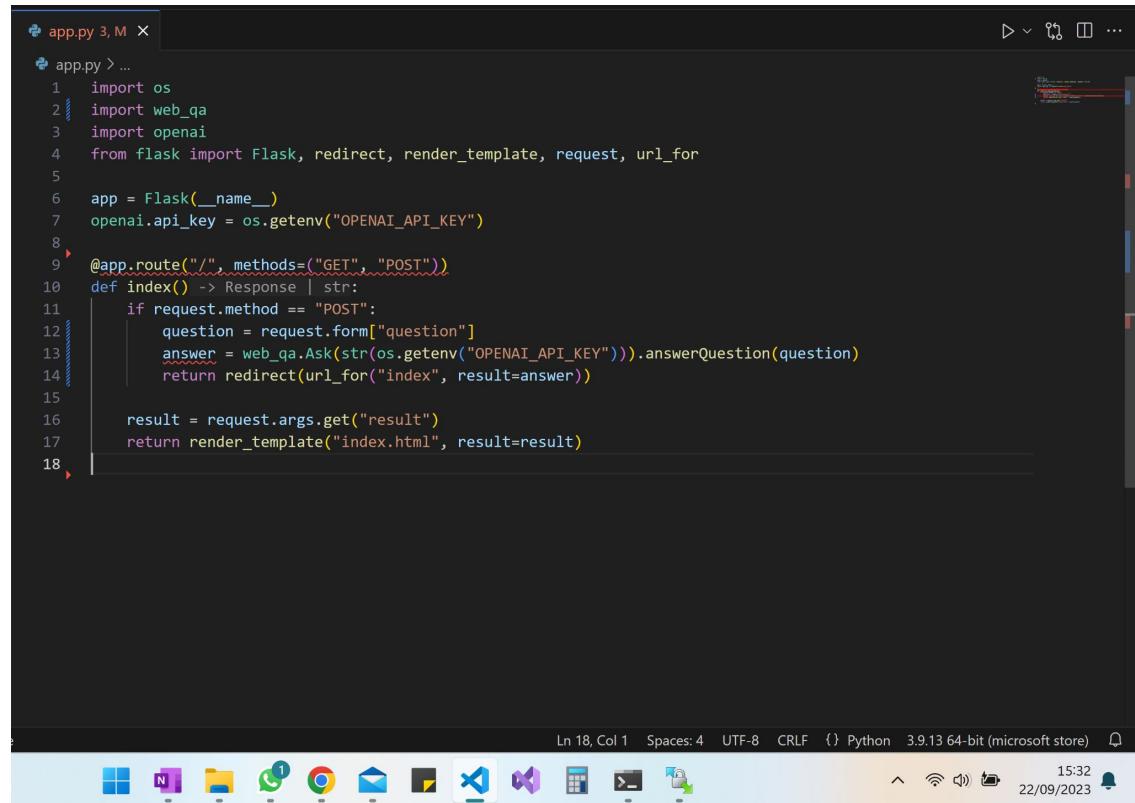
Executing the quick-start project. This project suggests pet names according to the type of pet you have, for example: cats and dogs



Python Flask

Taking the quick-start project as a reference, I integrated it with our project of answering questions on Python.

Because I refactored the project into a more object-oriented format, this integration was very simple. It was only necessary to import the class that receives and responds to questions and the system in Python was ready to work to answer questions.

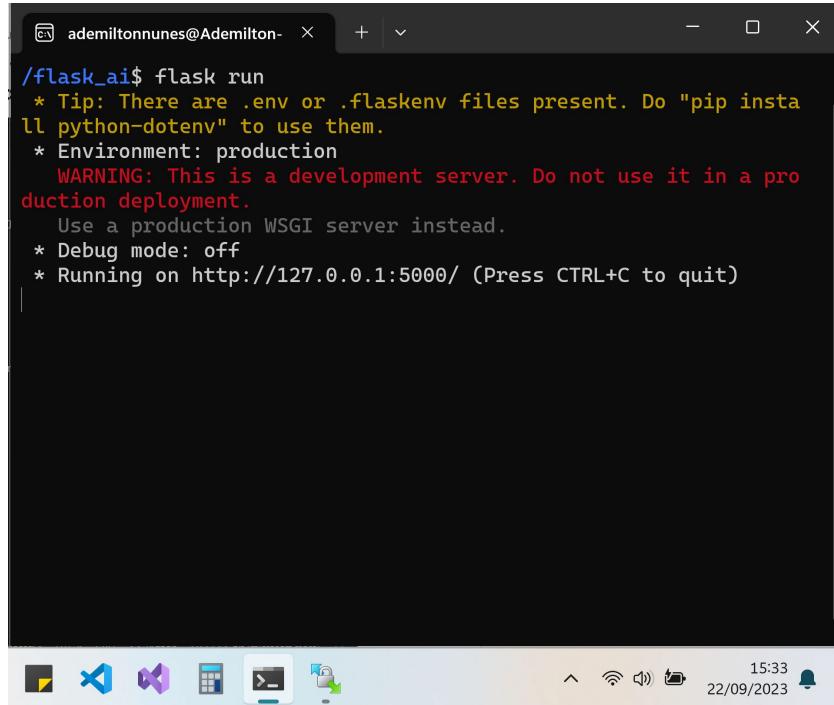


```
app.py 3, M x
app.py > ...
1  import os
2  import web_qa
3  import openai
4  from flask import Flask, redirect, render_template, request, url_for
5
6  app = Flask(__name__)
7  openai.api_key = os.getenv("OPENAI_API_KEY")
8
9  @app.route("/", methods=("GET", "POST"))
10 def index() -> Response | str:
11     if request.method == "POST":
12         question = request.form["question"]
13         answer = web_qa.Ask(str(os.getenv("OPENAI_API_KEY"))).answerQuestion(question)
14         return redirect(url_for("index", result=answer))
15
16     result = request.args.get("result")
17     return render_template("index.html", result=result)
18
```

Ln 18, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.9.13 64-bit (microsoft store) 15:32 22/09/2023

Python Flask

The Python Flask project also uses Ubuntu. Ubuntu is being used as a server.

A screenshot of a Linux desktop environment, specifically Ubuntu, showing a terminal window. The terminal window has a dark background and light-colored text. It displays the command '/flask_ai\$ flask run' followed by several configuration details and a warning message. The warning message is in red text and reads: 'WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.' The terminal window is titled 'ademiltnunes@Ademilton-' and has standard window controls (minimize, maximize, close) at the top. At the bottom of the screen, there is a dock with various icons, including a yellow square, a blue square, a purple square, a calculator, a file manager, and a terminal icon. The system tray shows the date '22/09/2023' and the time '15:33'.

```
/flask_ai$ flask run
 * Tip: There are .env or .flaskenv files present. Do "pip install python-dotenv" to use them.
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Starting on Ubuntu to be a server

Python Flask

With the Python Flask project, I submitted the same questions to compare with previous results. The same answers were gotten.

Python Flask

?

SFBU questions

What day is it?

Ask

I don't know.



1



15:52
22/09/2023

Python Flask

?

SFBU questions

What are MSCS Classes?

Ask

MSCS classes are graduate level courses in computer science that cover topics such as computer language theory, advanced software development, advanced algorithms, quantum computing, and machine learning.



15:52
22/09/2023



Python Flask

?

SFBU questions

Who is the SFBU's President?

Ask

Nicholas Ladany



15:53
22/09/2023



Python Flask

?

SFBU questions

What is SFBU?

Ask

San Francisco Bay University.



15:51
22/09/2023



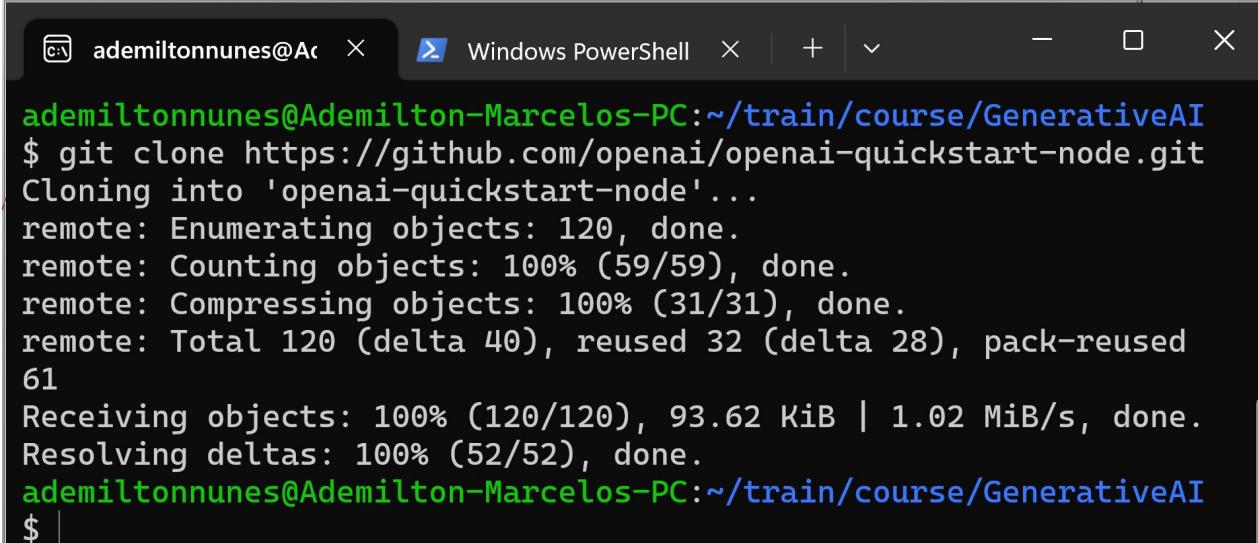
Node.js - Quick-Start

To continue transforming this project into a web-based system but on Node.js, I also acquired an [OpenAI-quickstart Node.js tutorial](#). This tutorial also served as a practical example of utilizing Node.js with ChatGPT to suggest pet names. By employing this Node.js Quick-Start guide, I gained insights into constructing a website and subsequently adapted our project into a web-based system capable of answering questions about a website using Node.

To implement the Node.js quickstart project, it is imperative to download the corresponding repository and establish a development environment.

Node.js - Quick-Start

Downloading the tutorial project



The screenshot shows a Windows PowerShell window with the title bar "ademiltnunes@Ac" and "Windows PowerShell". The command \$ git clone https://github.com/openai/openai-quickstart-node.git was run, resulting in the following output:

```
ademiltnunes@Ademilton-Marcelos-PC:~/train/course/GenerativeAI
$ git clone https://github.com/openai/openai-quickstart-node.git
Cloning into 'openai-quickstart-node'...
remote: Enumerating objects: 120, done.
remote: Counting objects: 100% (59/59), done.
remote: Compressing objects: 100% (31/31), done.
remote: Total 120 (delta 40), reused 32 (delta 28), pack-reused
61
Receiving objects: 100% (120/120), 93.62 KiB | 1.02 MiB/s, done.
Resolving deltas: 100% (52/52), done.
ademiltnunes@Ademilton-Marcelos-PC:~/train/course/GenerativeAI
$ |
```

The taskbar at the bottom of the screen includes icons for Mail, File Explorer, VS Code, Task View, and a calculator. The system tray shows the date and time as "28/09/2023 18:54".

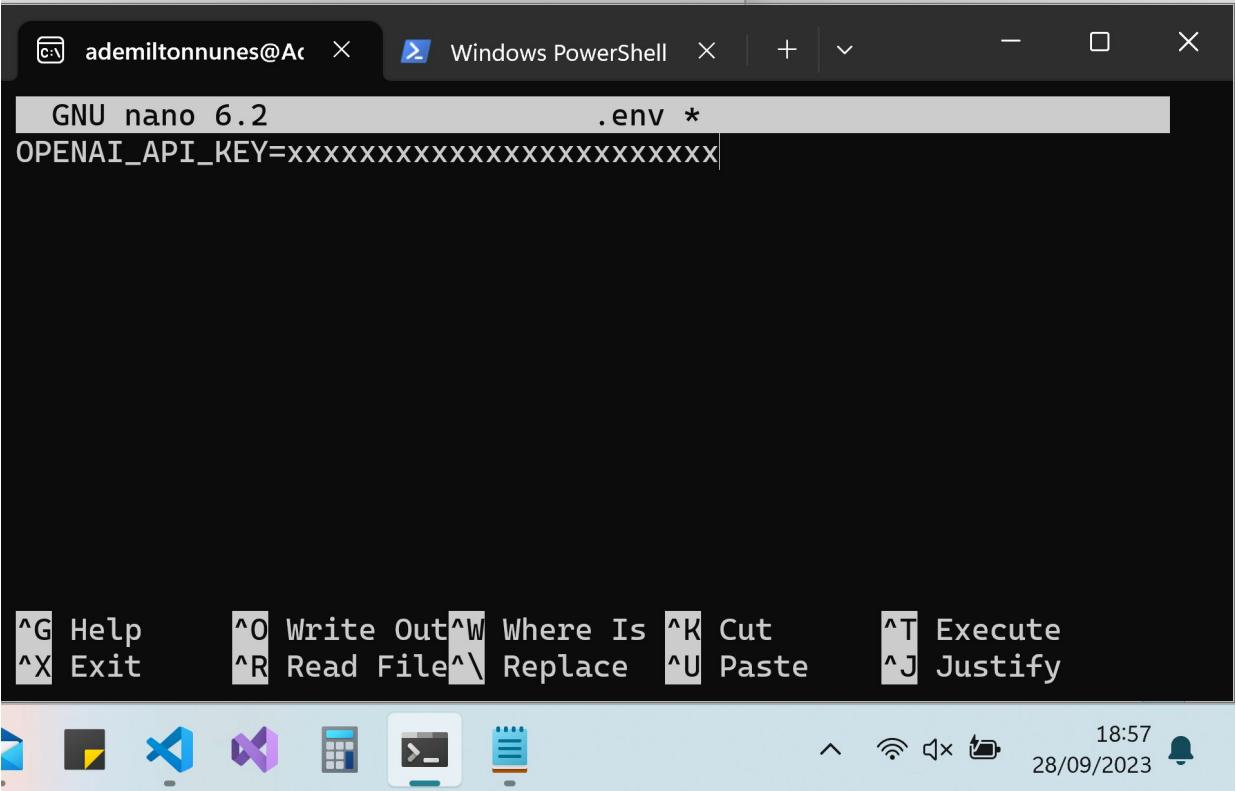
Node.js - Quick-Start

To make Node.js work it was necessary to install some modules. These modules are:

- npm install
- sudo apt install nodejs

Node.js - Quick-Start

Configuring API keys



A screenshot of a Windows PowerShell terminal window titled "Windows PowerShell". The title bar also shows the session name "ademiltonnunes@Ac" and the file name ".env". The terminal is running the command-line editor "GNU nano 6.2". The current file is ".env" and it contains the environment variable "OPENAI_API_KEY=xxxxxxxxxxxxxxxxxxxxxx". The bottom of the screen displays a series of keyboard shortcuts for the nano editor.

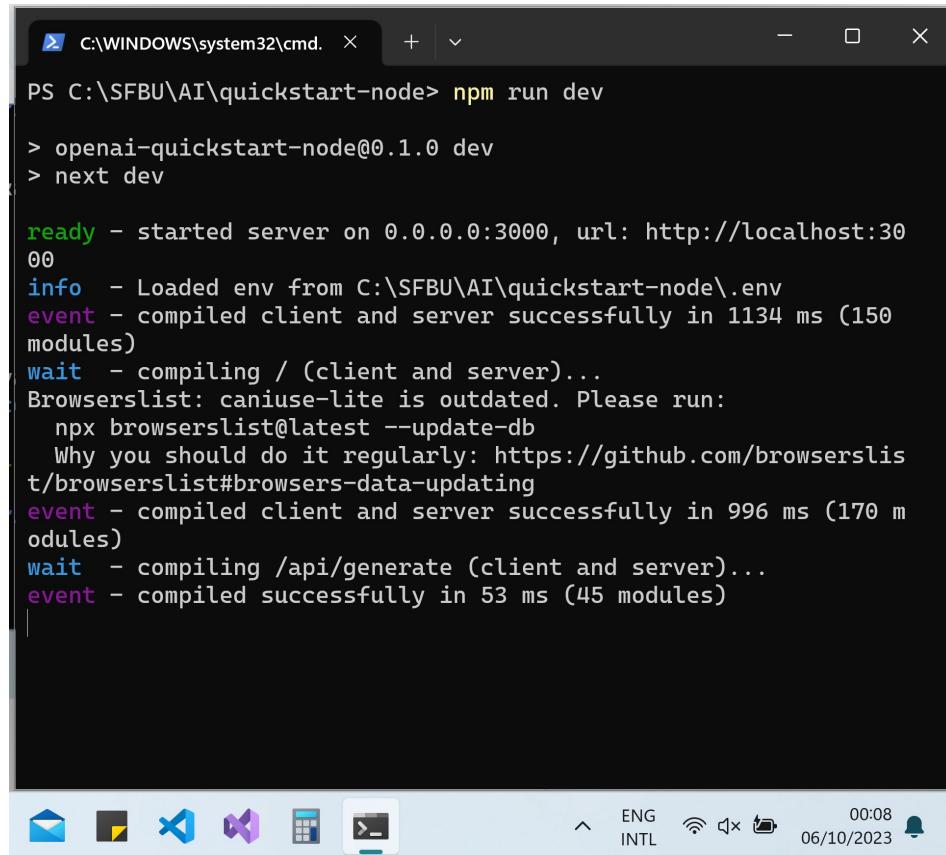
```
GNU nano 6.2 .env *
OPENAI_API_KEY=xxxxxxxxxxxxxxxxxxxxxx
```

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify

18:57 28/09/2023

Node.js - Quick-Start

Executing the project by the command: **npm run dev**



```
PS C:\SFBU\AI\quickstart-node> npm run dev

> openai-quickstart-node@0.1.0 dev
> next dev

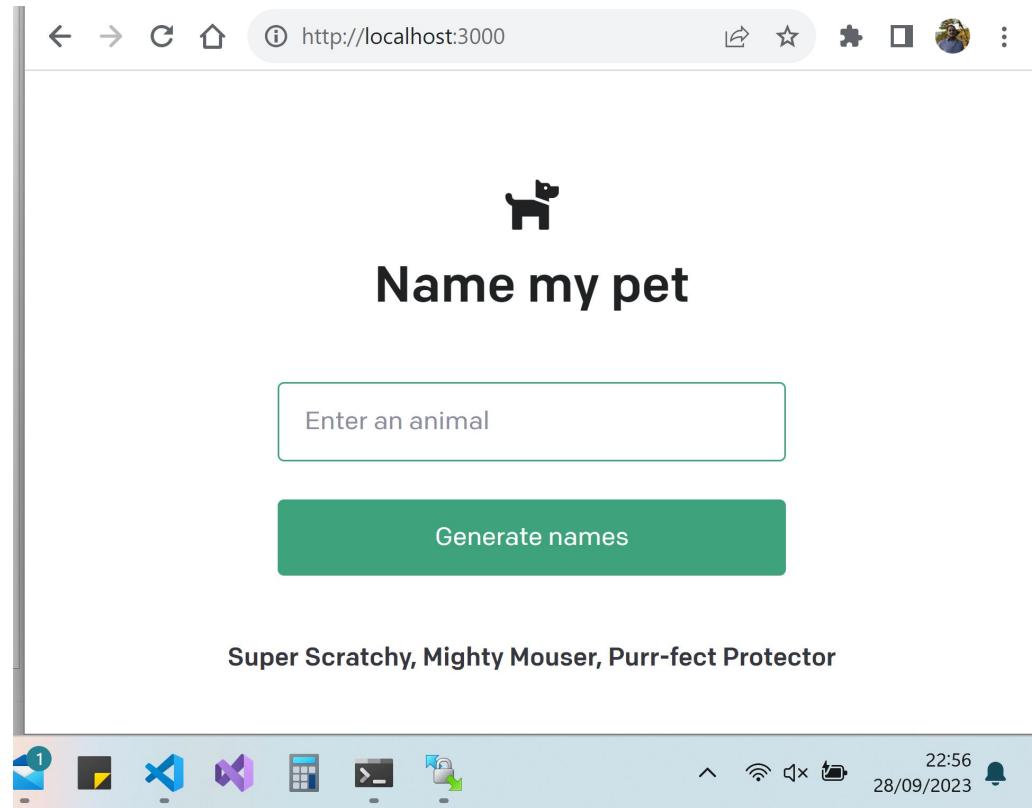
ready - started server on 0.0.0.0:3000, url: http://localhost:3000
info - Loaded env from C:\SFBU\AI\quickstart-node\.env
event - compiled client and server successfully in 1134 ms (150 modules)
wait - compiling / (client and server)...
Browserslist: caniuse-lite is outdated. Please run:
  npx browserslist@latest --update-db
  Why you should do it regularly: https://github.com/browserslist/browserslist#browsers-data-updating
event - compiled client and server successfully in 996 ms (170 modules)
wait - compiling /api/generate (client and server)...
event - compiled successfully in 53 ms (45 modules)
```

The screenshot shows a Windows Command Prompt window titled "cmd" with the path "C:\WINDOWS\system32\cmd." The window displays the output of the command "npm run dev". The output shows the execution of the "openai-quickstart-node" package version 0.1.0, specifically the "dev" script. It uses the "next" package to start a development server on port 3000. The log includes messages about loading the environment from ".env", compiling the client and server, and updating Browserslist. It also shows the compilation of an API endpoint and the successful compilation of the entire project.



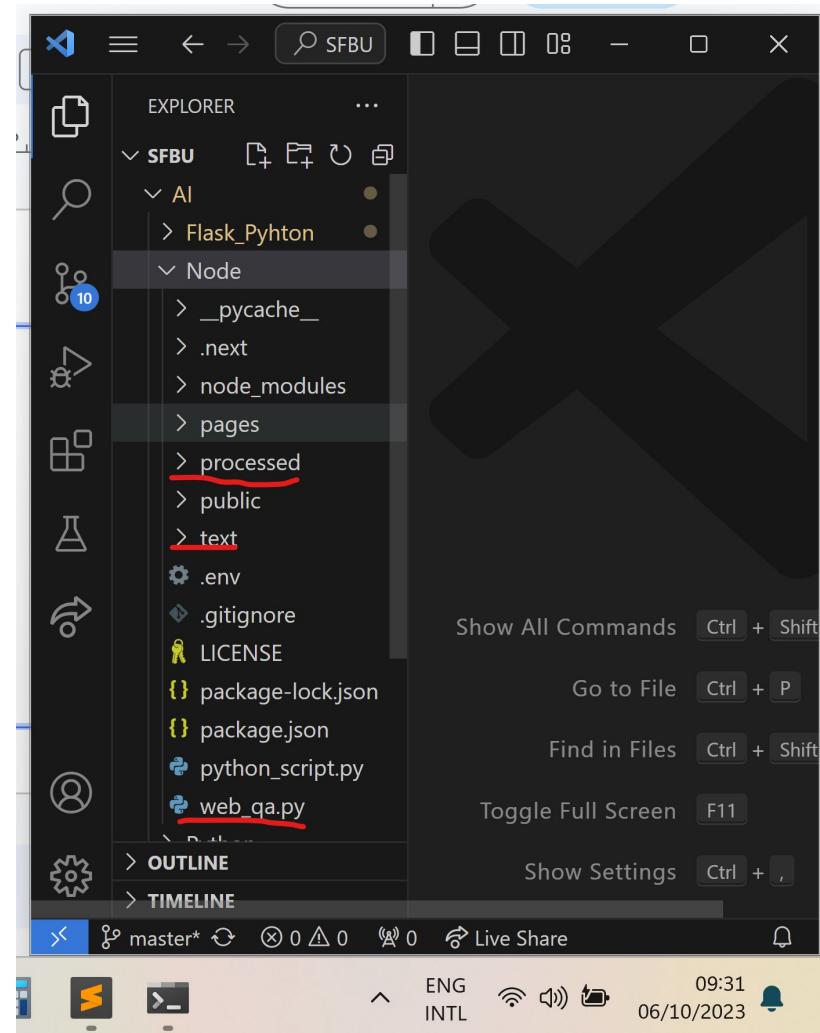
Node.js - Quick-Start

Executing the quick-start project. This project suggests pet names according to the type of pet you have, for example: cats and dogs



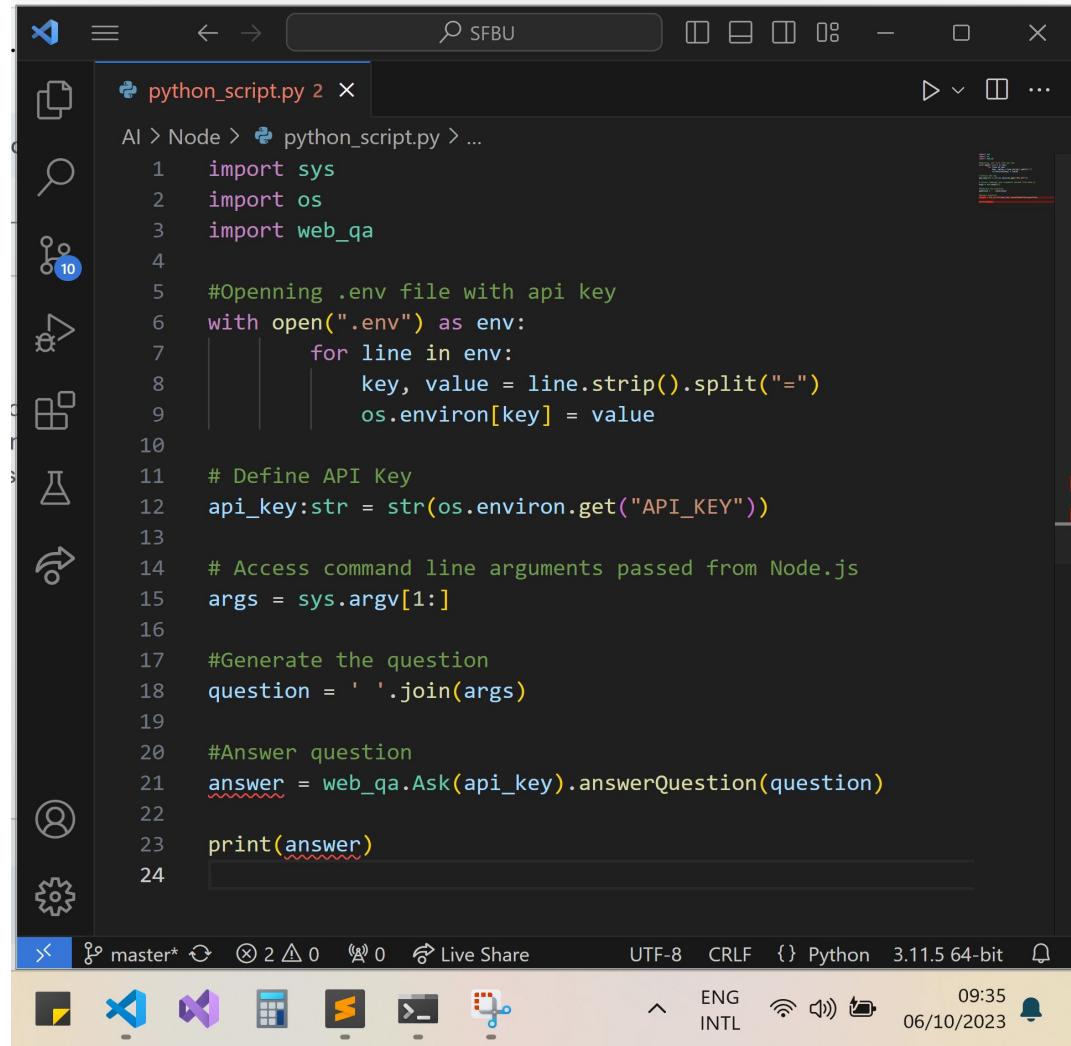
Node.js

Taking the Node.js quickstart project as a reference, I integrated it into our Python project to answer questions about a website within our project.



Node.js

Because we are using two different technologies, Python and Node, I created a intermediate Python script to ask the questions received by the Node.js client. This script is responsible for putting the responses in the format that Node.js expects.



```
python_script.py 2 x
AI > Node > python_script.py > ...
1 import sys
2 import os
3 import web_qa
4
5 #Openning .env file with api key
6 with open(".env") as env:
7     for line in env:
8         key, value = line.strip().split("=")
9         os.environ[key] = value
10
11 # Define API Key
12 api_key:str = str(os.environ.get("API_KEY"))
13
14 # Access command line arguments passed from Node.js
15 args = sys.argv[1:]
16
17 #Generate the question
18 question = ' '.join(args)
19
20 #Answer question
21 answer = web_qa.Ask(api_key).answerQuestion(question)
22
23 print(answer)
24
```

Node.js

In the quick-start Node.js project, when the client makes a request, this is received in the generate.js file, so I made changes in this file. So, upon receiving the client's request, we will call the intermediate Python file that answers the questions of a website, receiving the response, we return it to the client.

Node.js



```
index.js      generate.js
1 import { exec } from 'child_process';
2 import path from 'path';
3
4 export default async function (req, res) {
5   const question = req.body.question || '';
6   if (question.trim().length === 0) {
7     res.status(400).json({
8       error: {
9         message: "Please enter a valid question",
10      },
11    });
12    return;
13  }
14
15  try {
16    const pythonScriptPath = 'python_script.py';
17    const scriptPath = path.join(__dirname, 'python_script.py');
18    const scriptArguments = [question];
19    const command = `python ${pythonScriptPath} ${scriptArguments.join(' ')}`;
20
21    exec(command, (error, stdout, stderr) => {
22      if (error) {
23        console.error(`Error executing Python script: ${error}`);
24        return res.status(500).json({
25          error: {
26            message: 'An error occurred during script execution.',
27          },
28        });
29      }
30
31      console.log('Python script output:');
32      console.log(stdout);
33      res.status(200).json({ result: stdout });
34    });
35  } catch (error) {
36    console.error(`Error: ${error.message}`);
37    res.status(500).json({
38      error: {
39        message: 'An error occurred during your request.',
40      },
41    });
42  }
43}

```

Line 21, Column 47 Spaces: 2 JavaScript



Node.js

With the project in Node.js, I sent the same questions to compare with previous results. The same responses were received.

Node.js



?

SFBU question

What day is it?

Ask

I don't know.





Node.js

?

SFBU question

What are MSCS Classes?

Ask

MSCS classes are graduate level courses in computer science that cover topics such as computer language theory, advanced software development, advanced algorithms, quantum computing, and machine learning.





http://localhost:3000



Node.js

?

SFBU question

Who is SFBU's President?

Ask

Nicholas Ladany



ENG
INTL



09:59
06/10/2023



Node.js

?

SFBU question

What is SFBU?

Ask

San Francisco Bay University.



^

ENG
INTL



10:00
06/10/2023



Conclusion

The OpenAI tutorial, "[Website Q&A with Embeddings](#)," provided the foundation for the creation of a customer support system employing ChatGPT to construct a web-based platform capable of addressing inquiries about a website. This tutorial proved instrumental in successfully navigating every phase of the development process, offering adaptability and seamless integration.

Throughout this project, we showcased the implementation process in a Jupyter Notebook, utilizing Python on an Ubuntu platform, and further extended it to a web-based application using Python Flask and Node.js. A comprehensive overview of the entire website processing workflow can be accessed in the GitHub repository linked within this document.