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)

Table of Content

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.

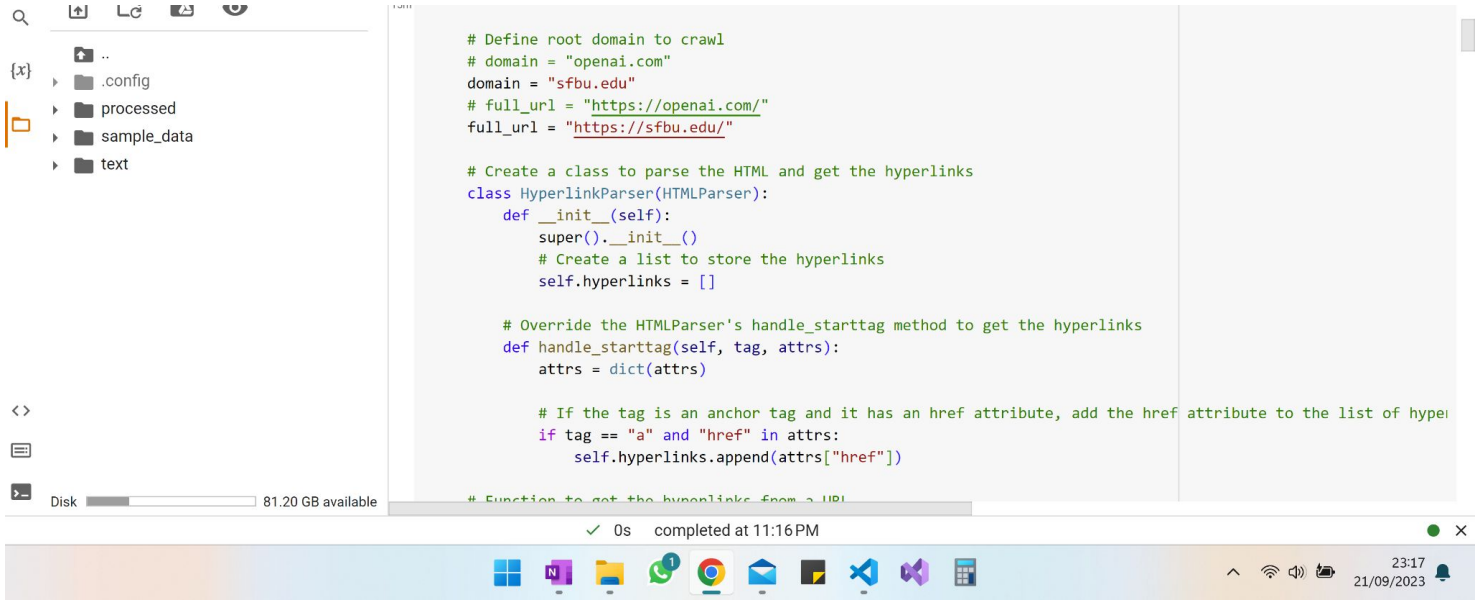
The chosen model for this project is the gpt-3.5-turbo-instruct, which generates responses in a natural and coherent manner, leveraging the information retrieved from the embeddings.

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:

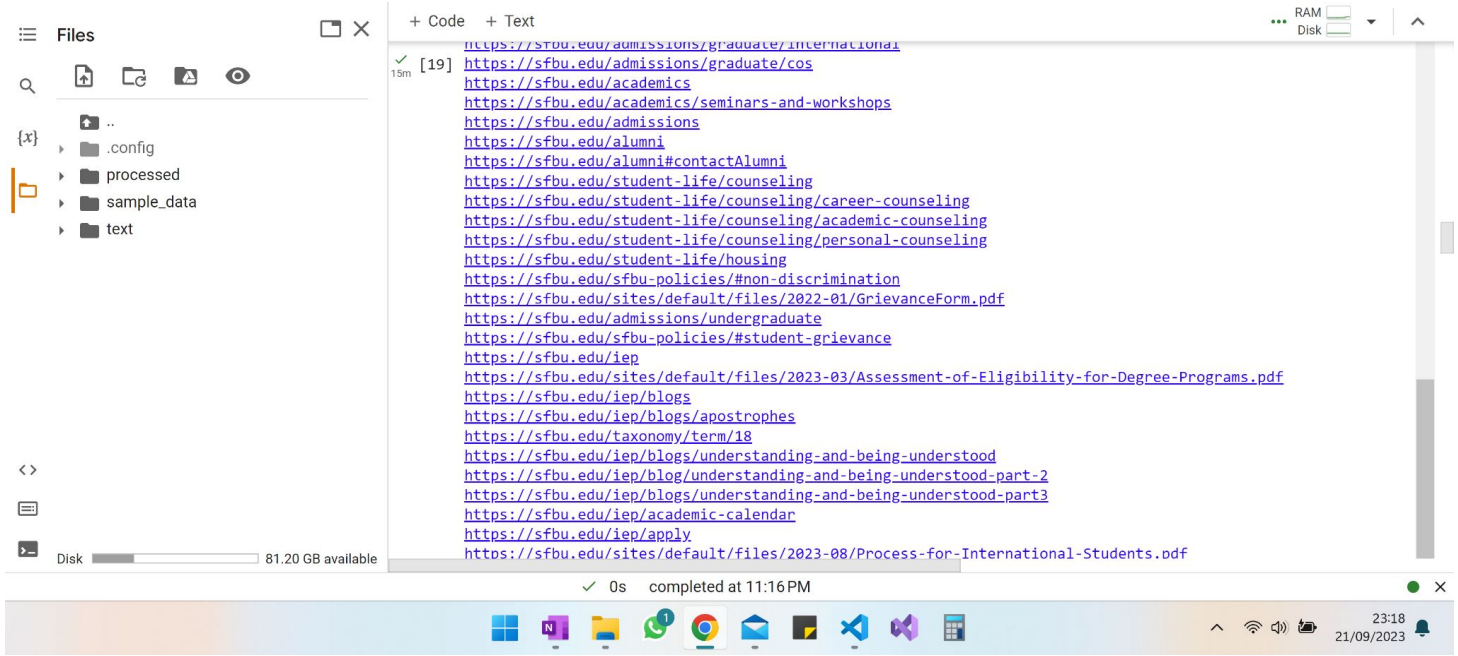
Jupyter Notebook

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



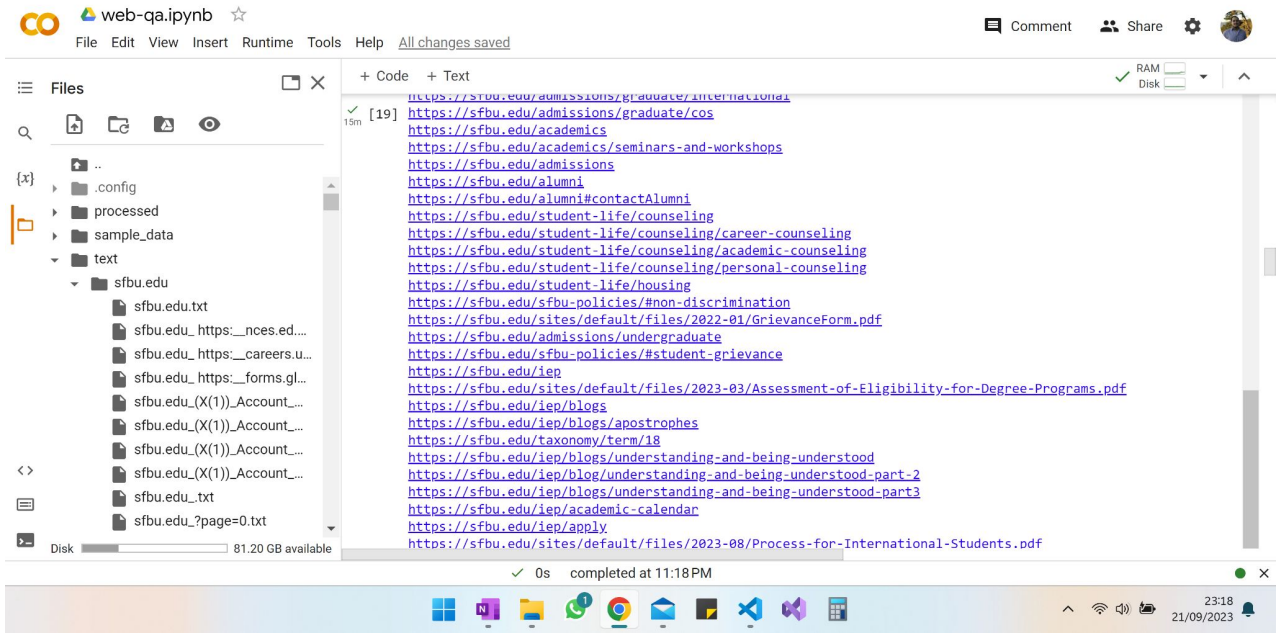
Jupyter Notebook

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



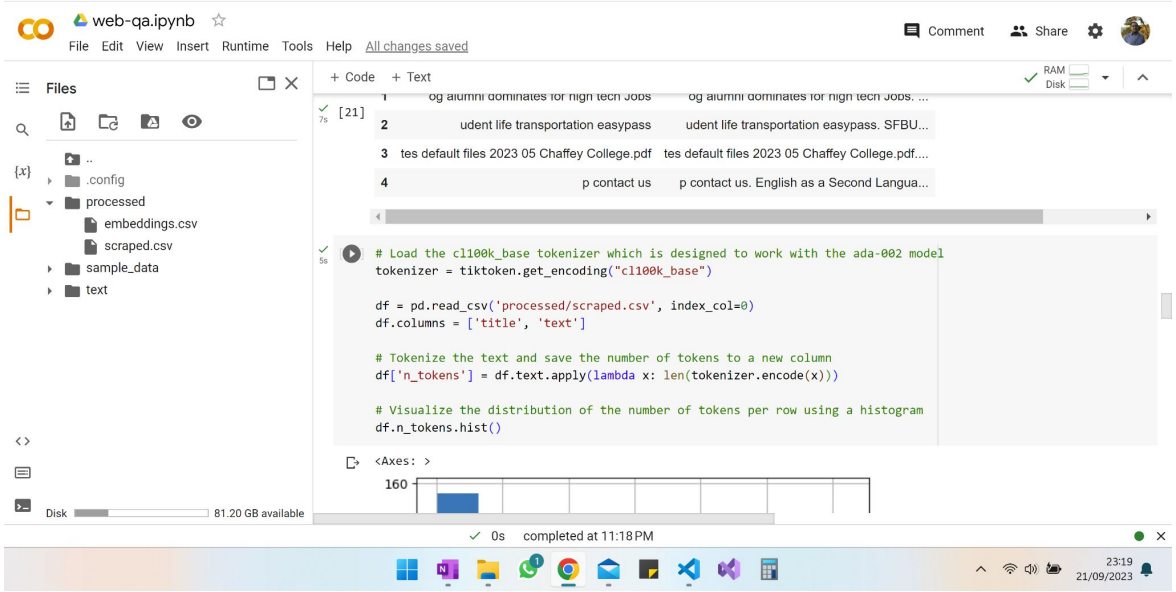
Jupyter Notebook

After visiting links, the texts are saved in txt files



Jupyter Notebook

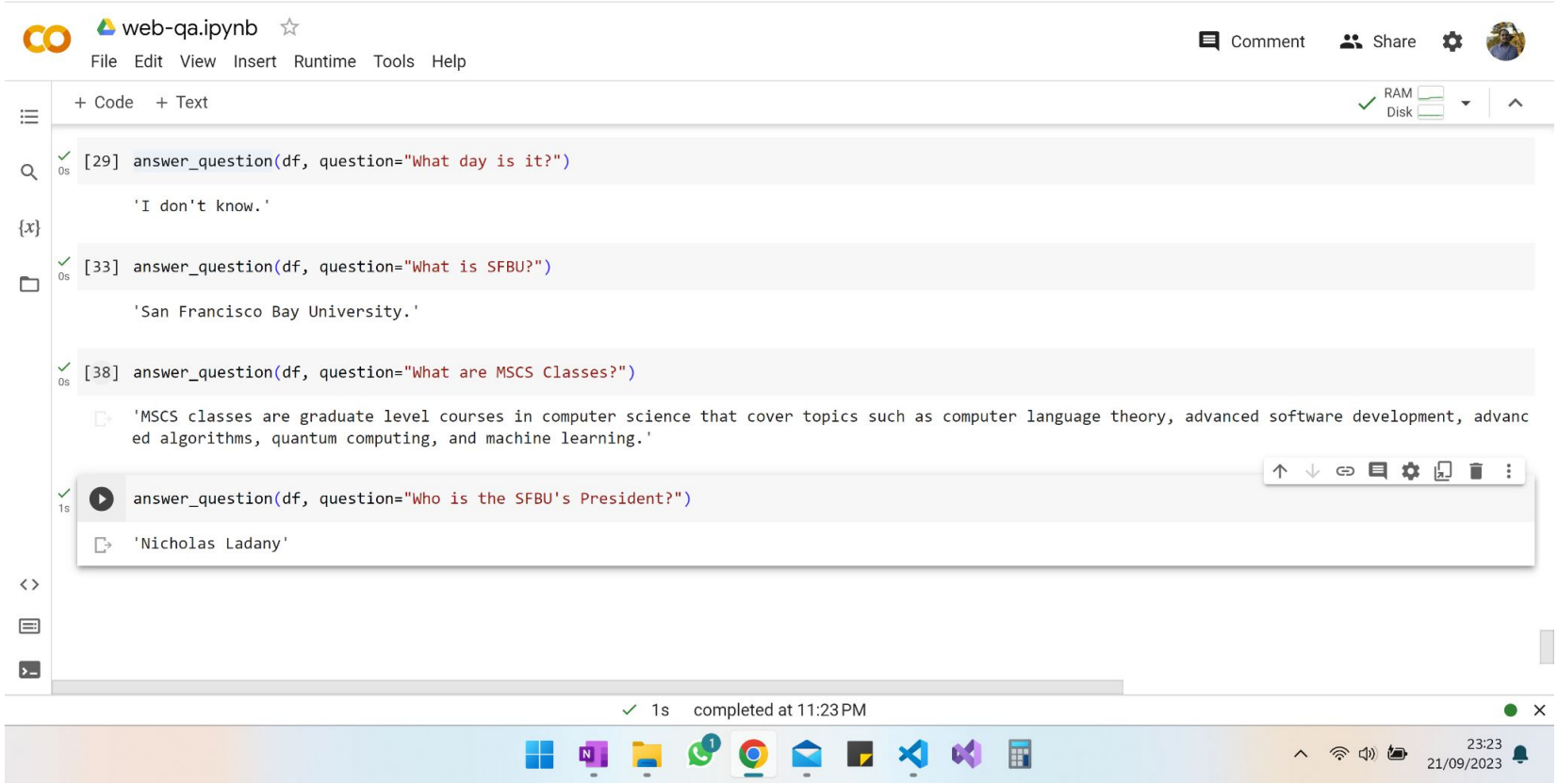
After this, the tokenization process is carried out on embedding indexes



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

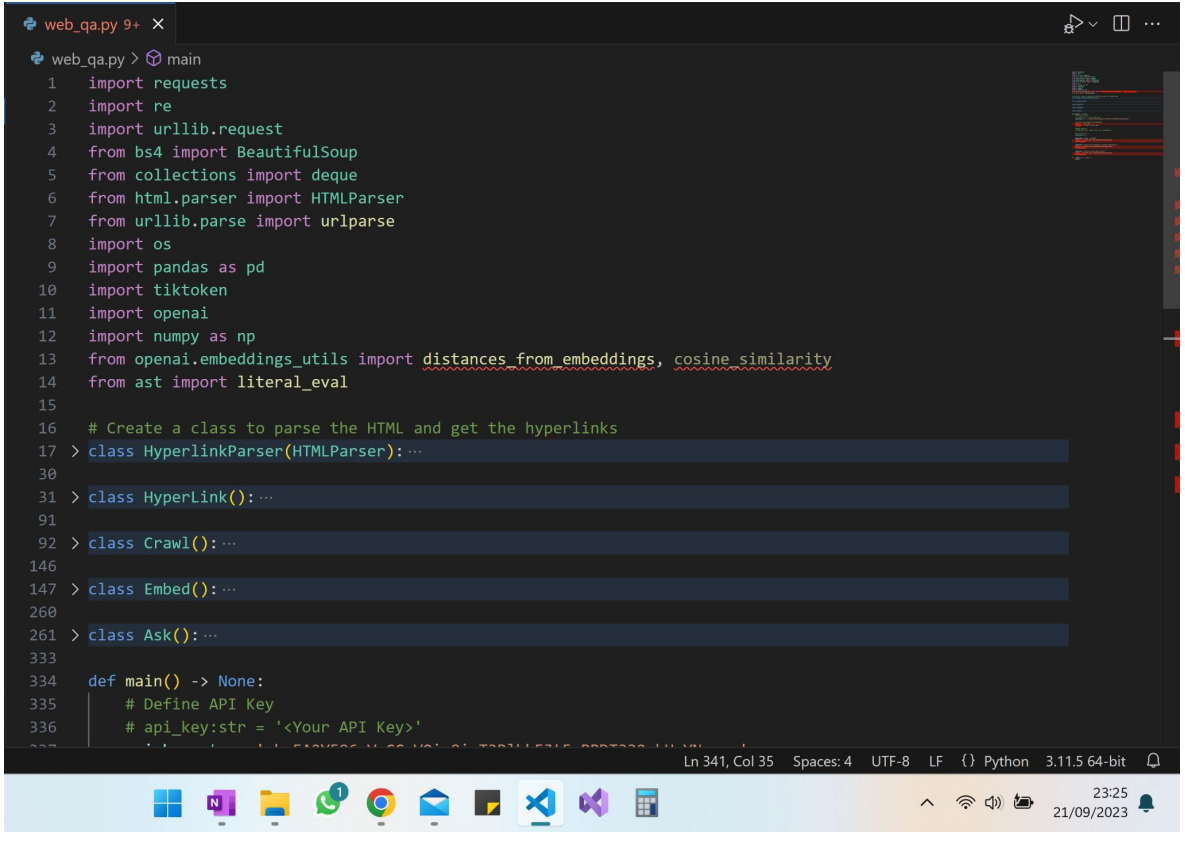
The Jupyter notebook content was exported in Python code. For this to work it was necessary to install some modules in Python. These modules are:

- 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:

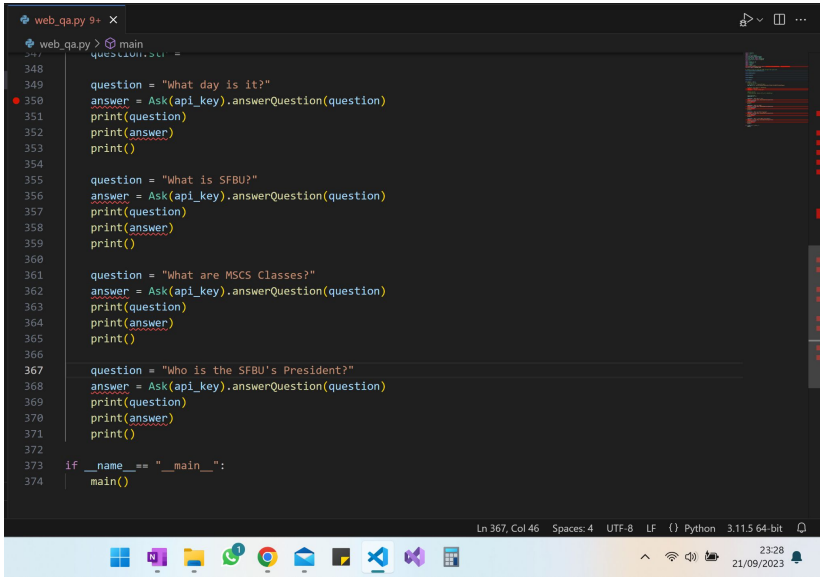
Python on Ubuntu

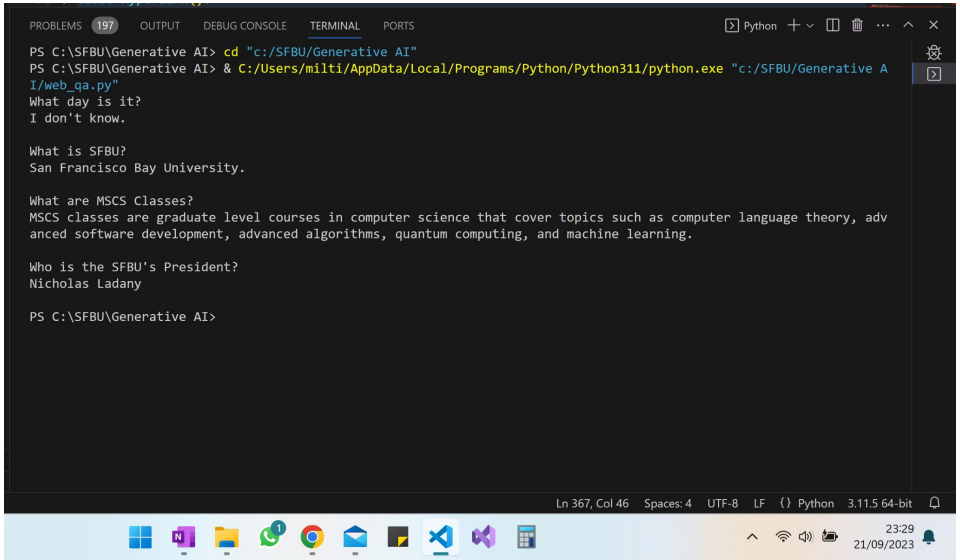


Python on Ubuntu

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

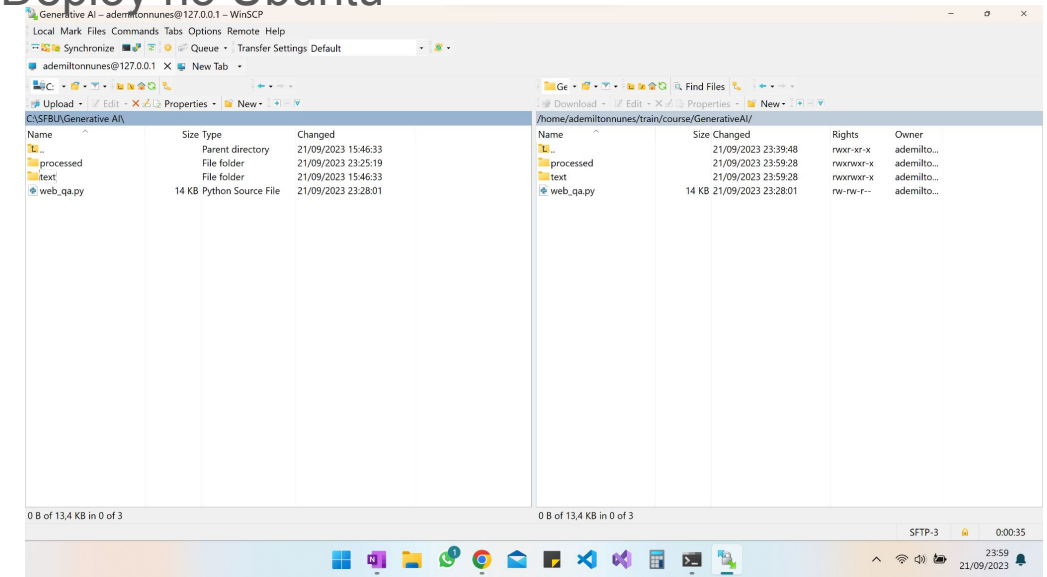




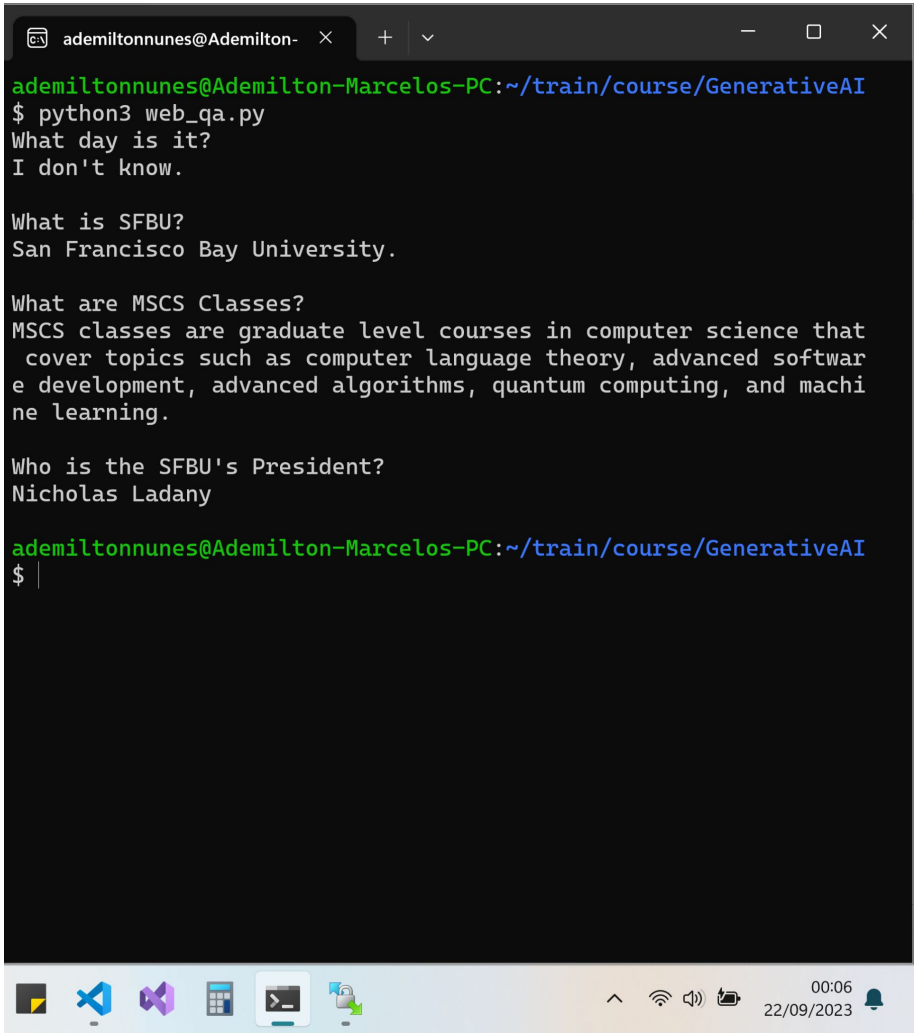
Python on Ubuntu

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

Deploy no Ubuntu



Python on Ubuntu

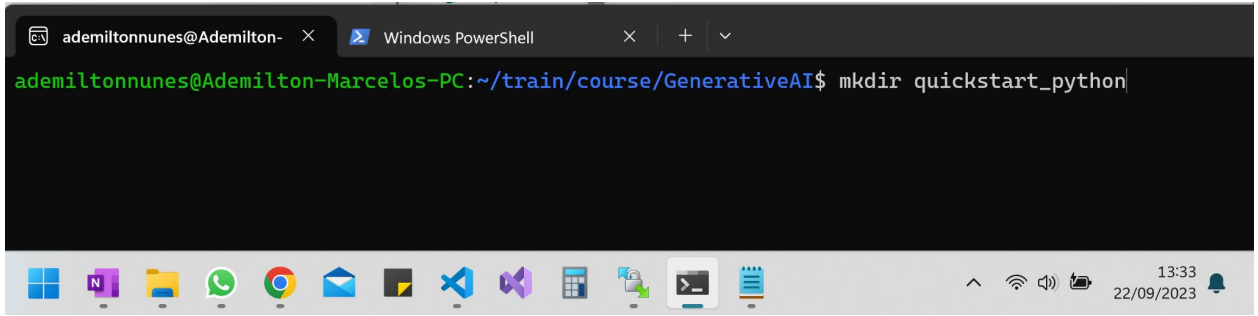


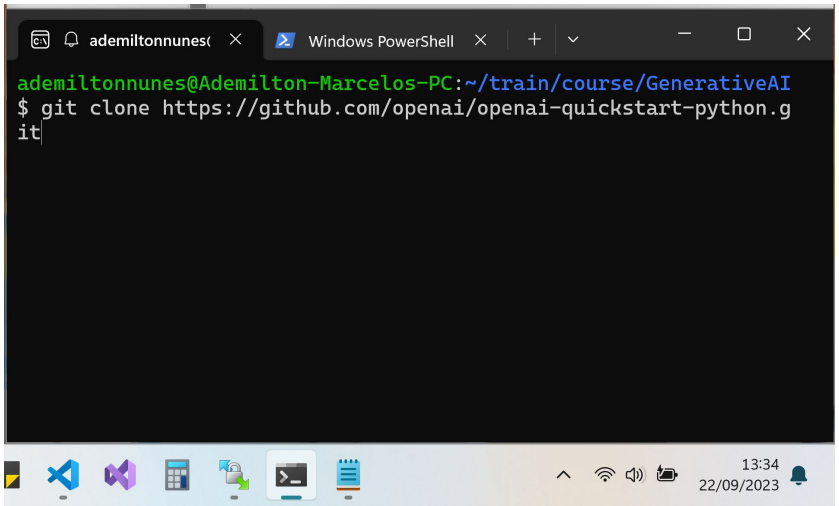
Python Flask - Quick-Start

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

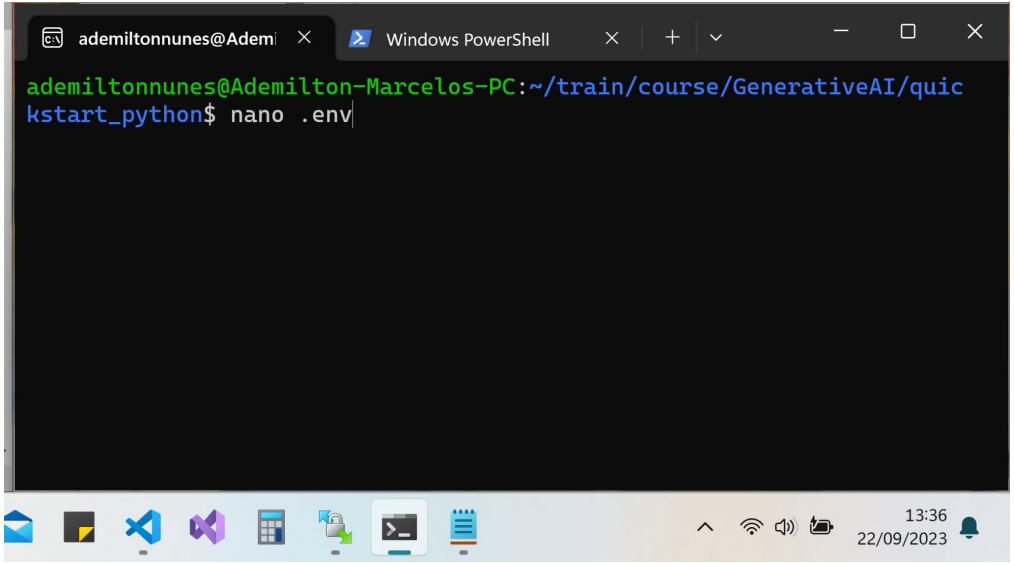


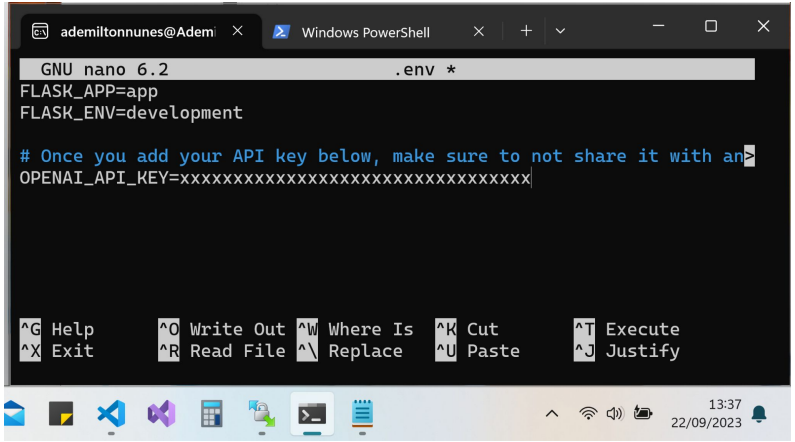


Downloading the tutorial project

Python Flask - Quick-Start

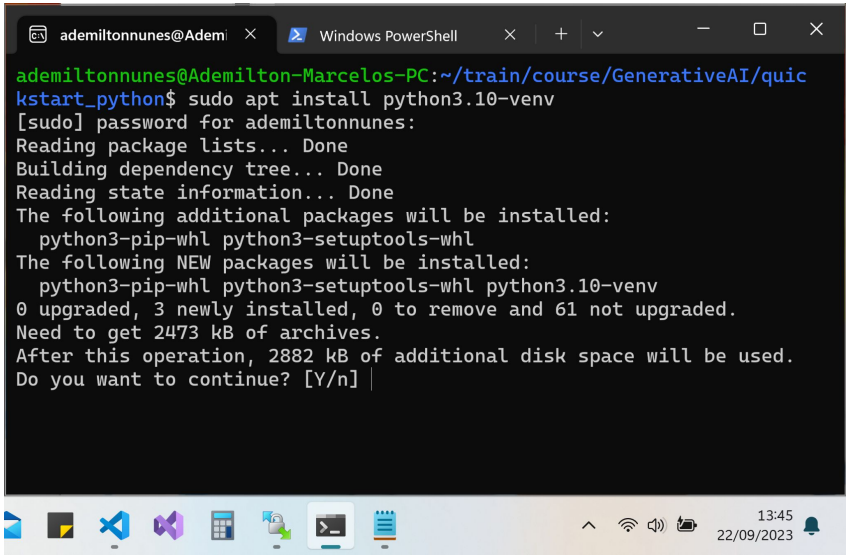
Configuring API keys

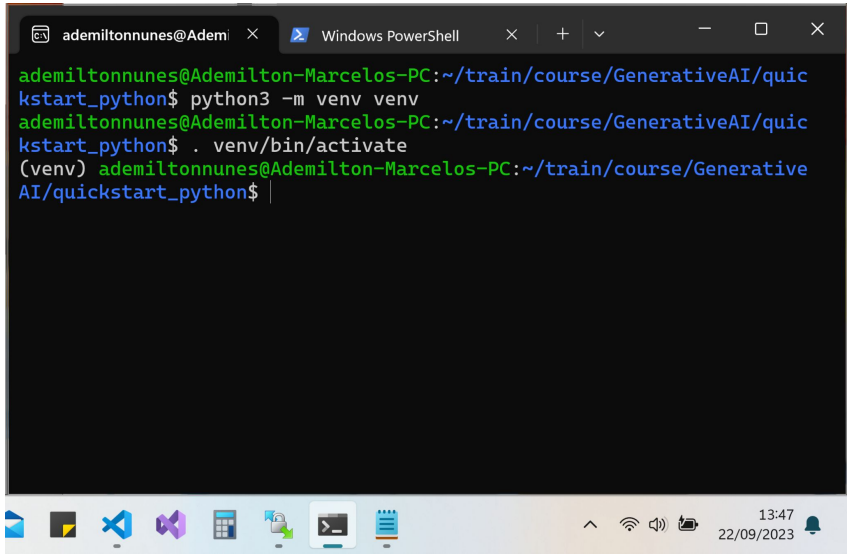




Python Flask - Quick-Start

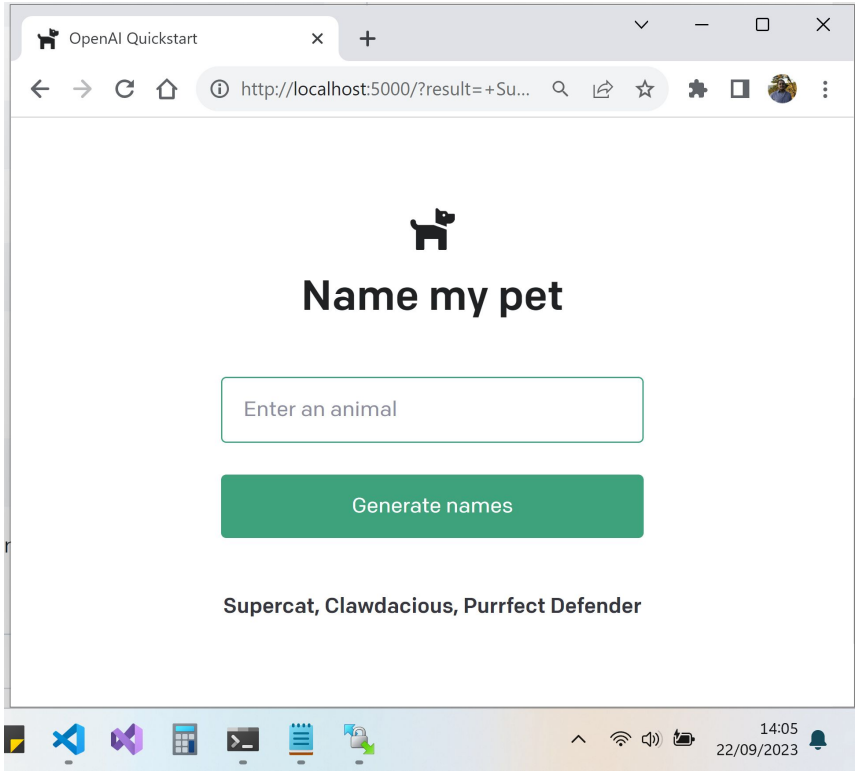
Creating and activating the development environment.





Python Flask - Quick-Start

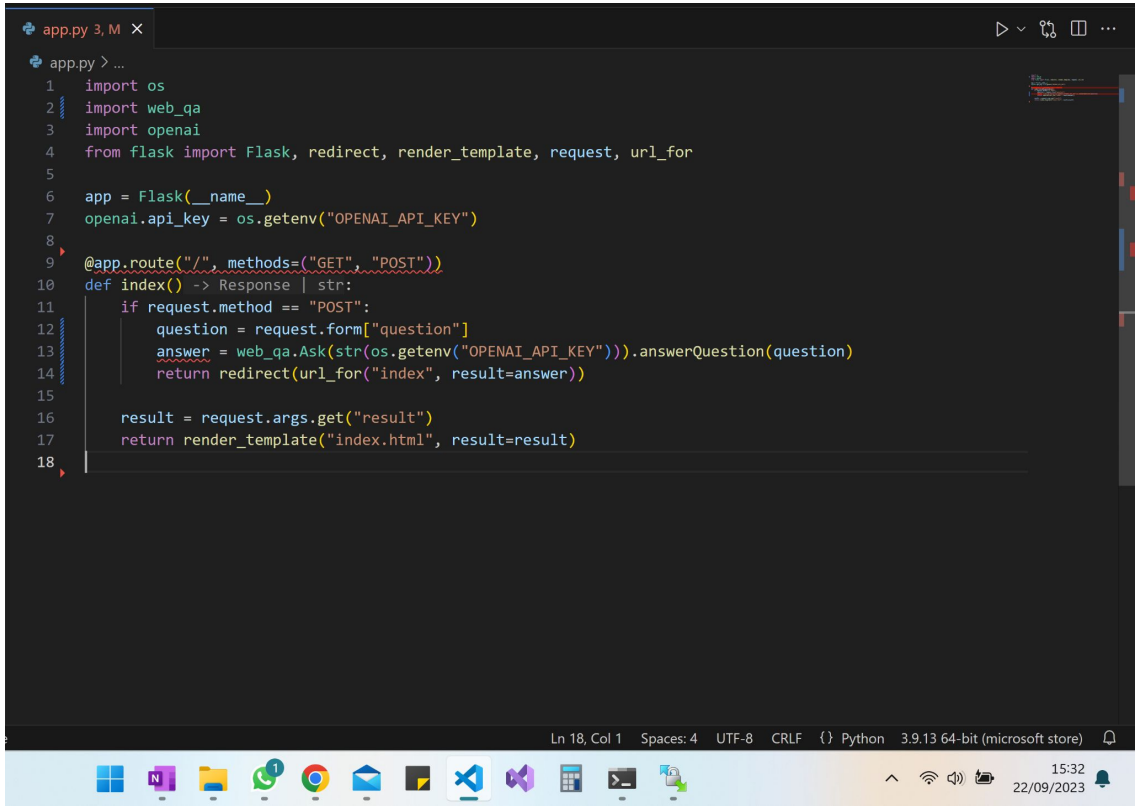
Executing the quick-start project. This project suggests pet names according to the type of pet you have, for example: cats and dogs



Python Flask

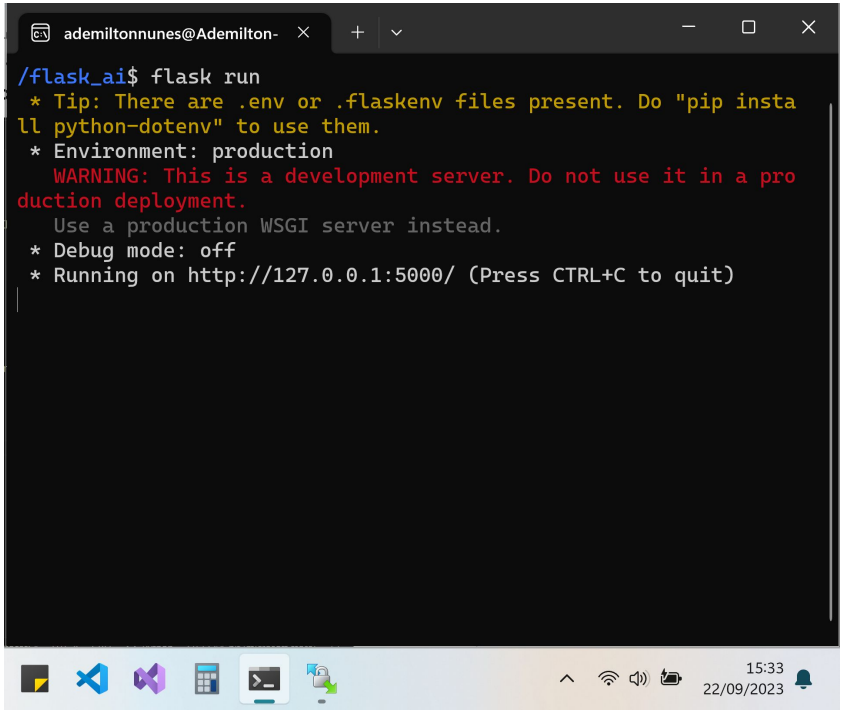
Taking the quick-start project as a reference, I integrated it with our projet of answering questions on Python.

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.



Python Flask

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

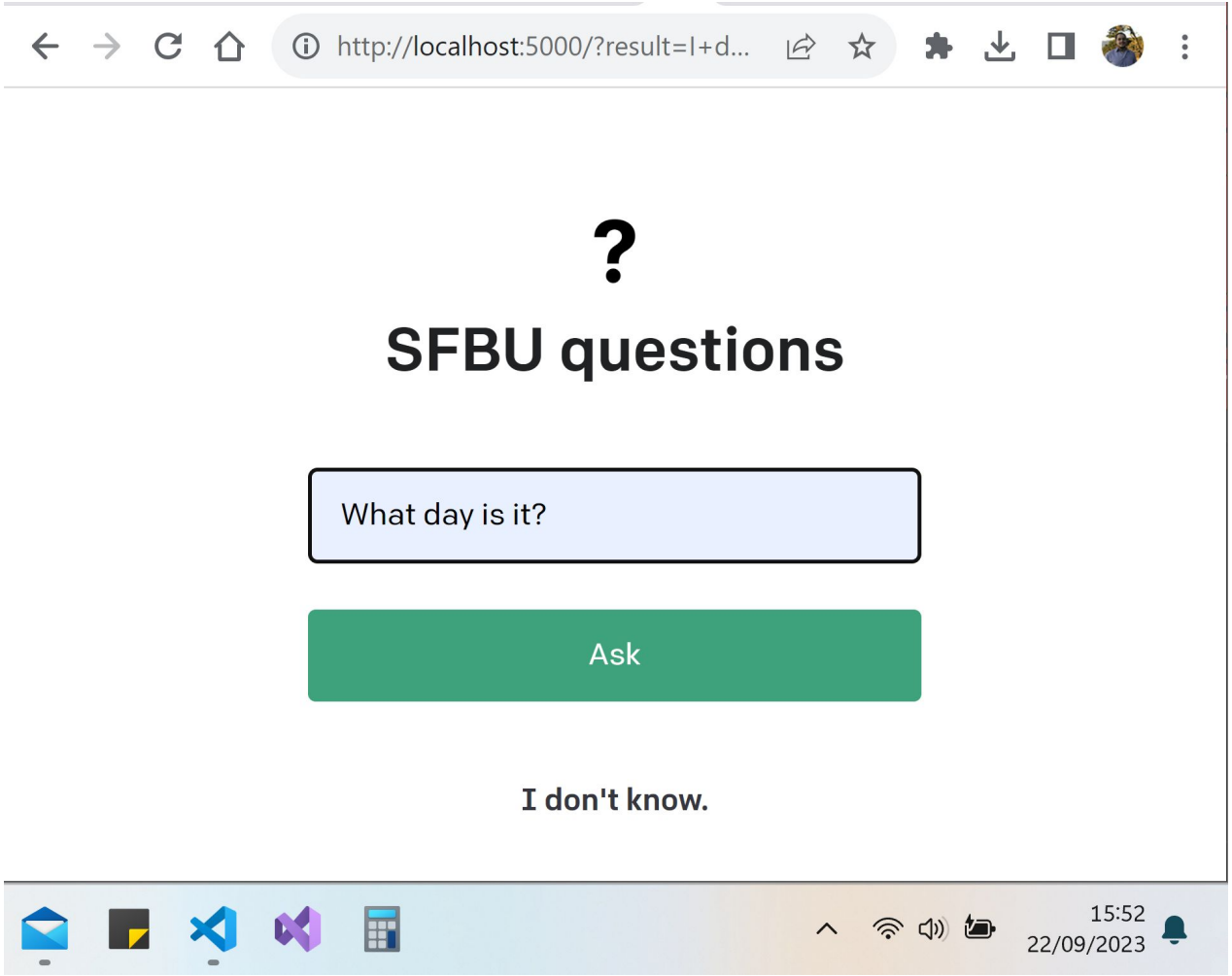


Starting on Ubuntu to be a server

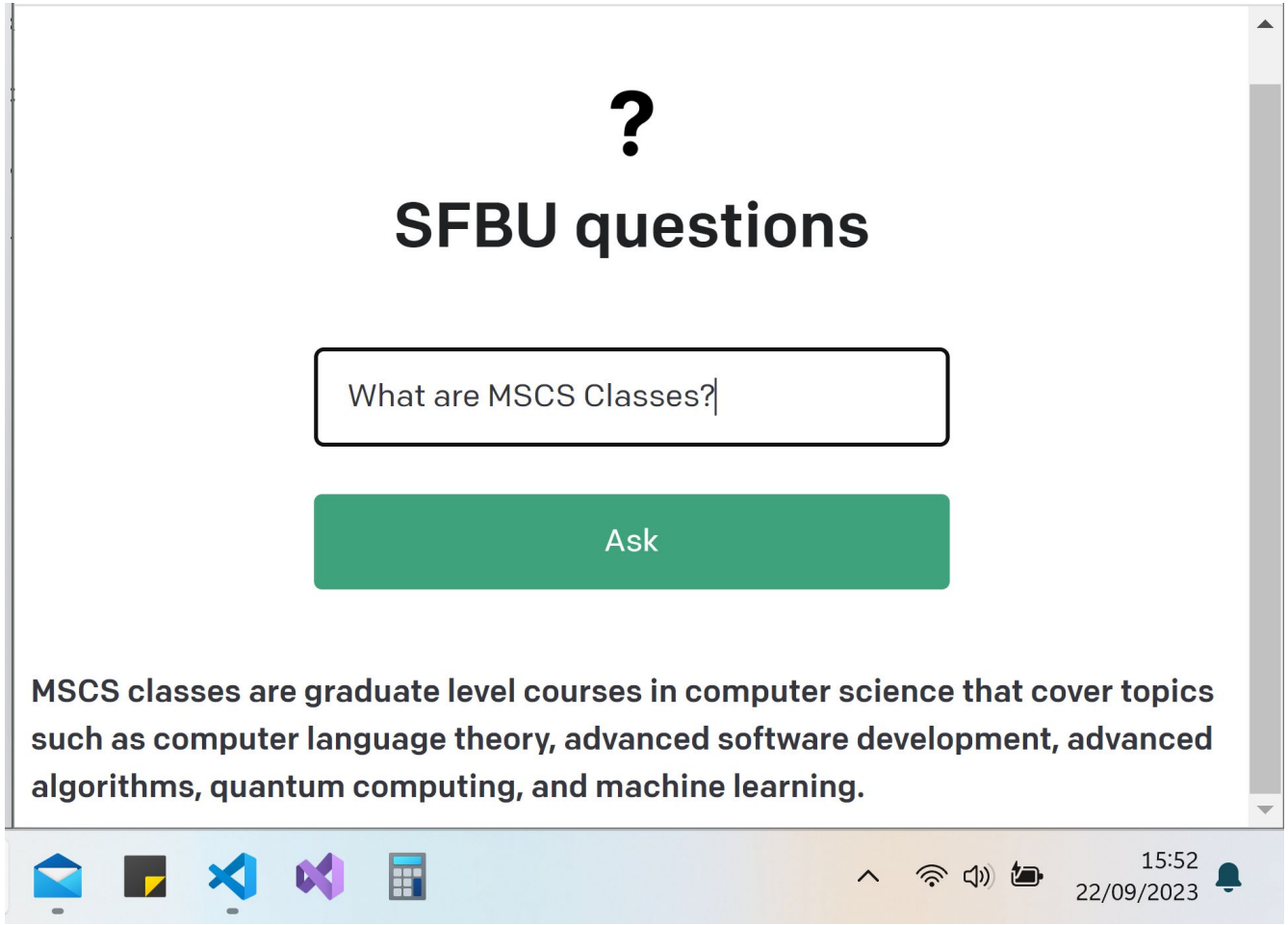
Python Flask

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

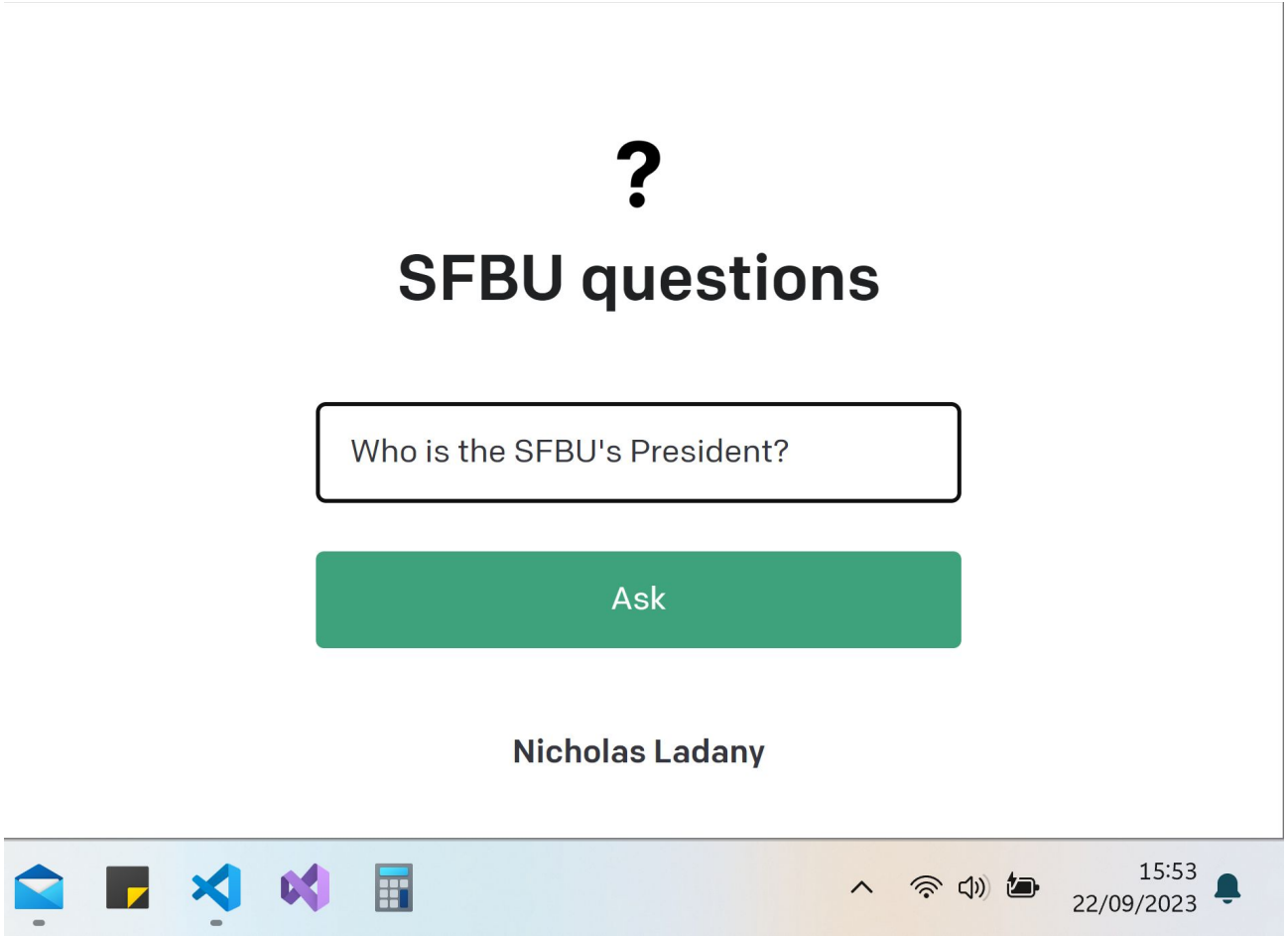
Python Flask



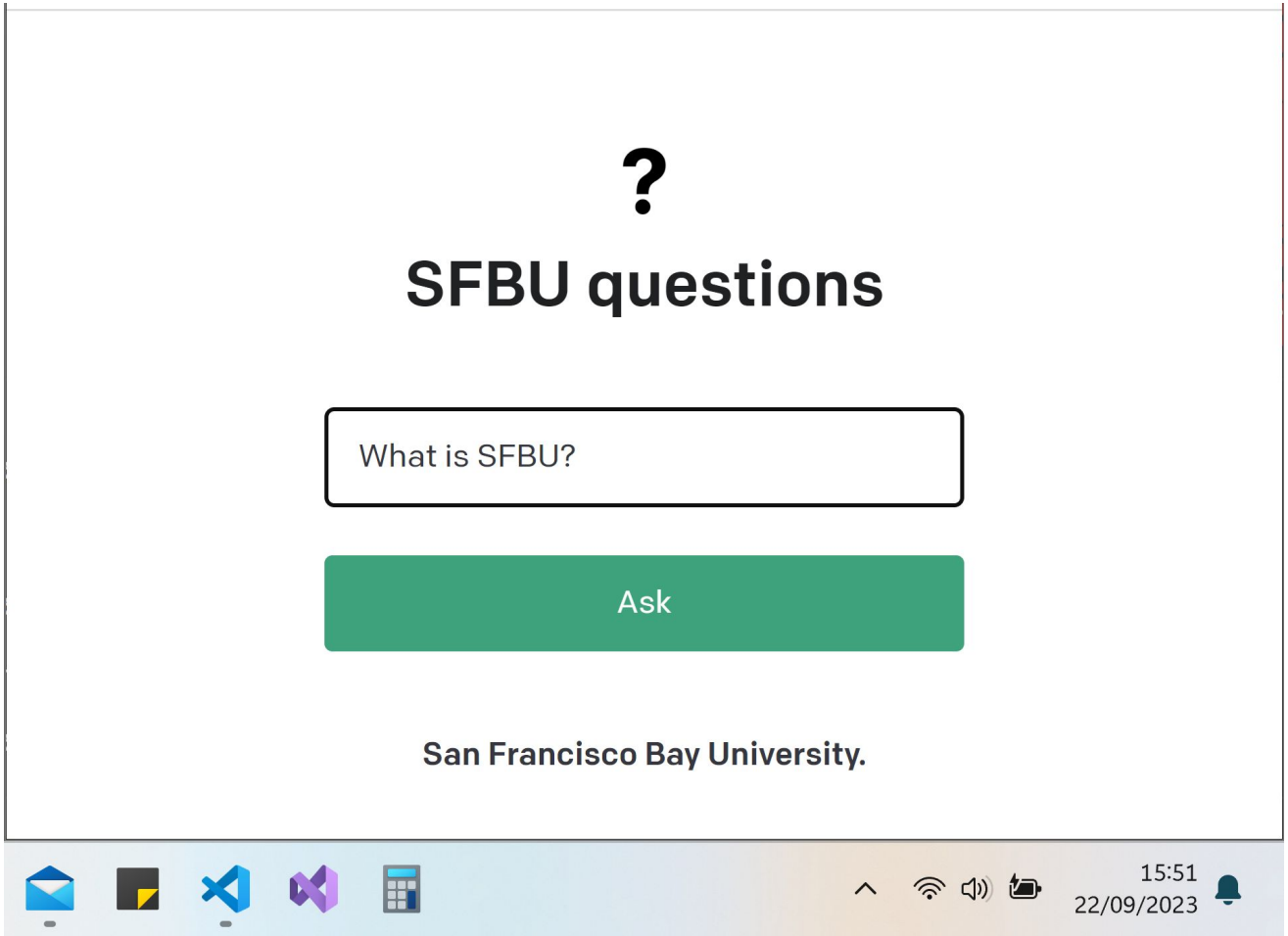
Python Flask



Python Flask



Python Flask



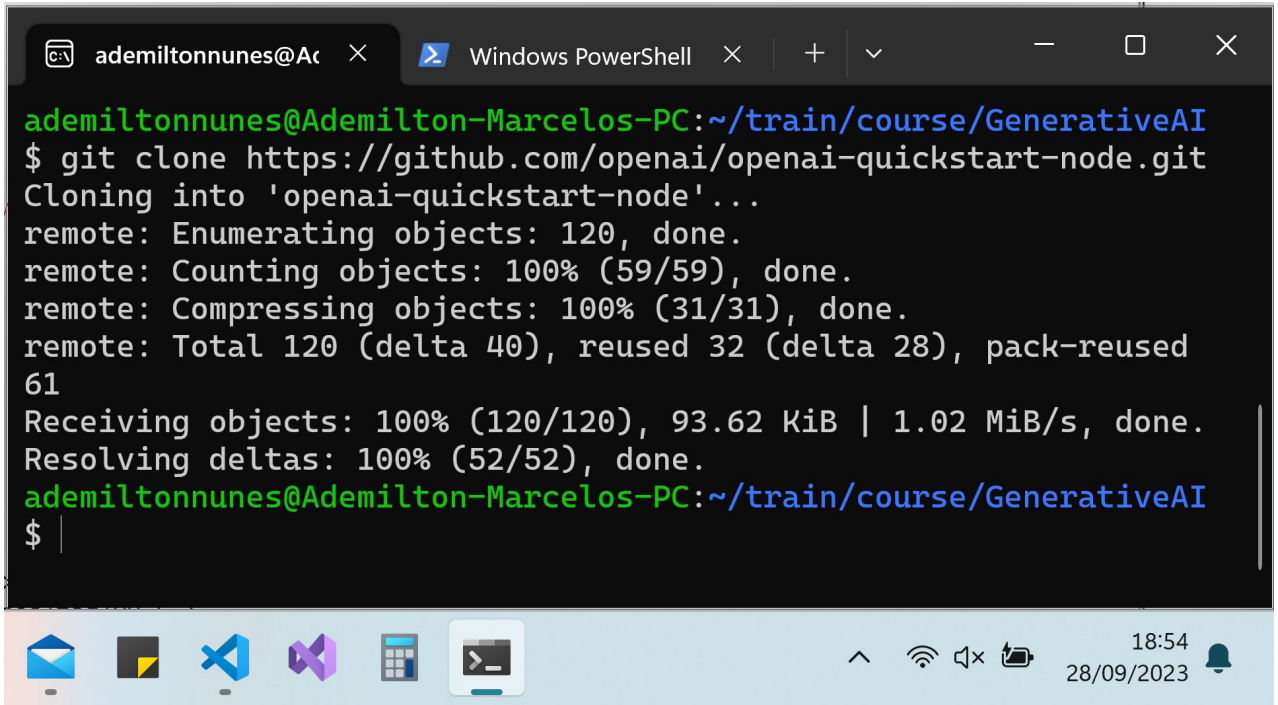
Node.js - Quick-Start

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



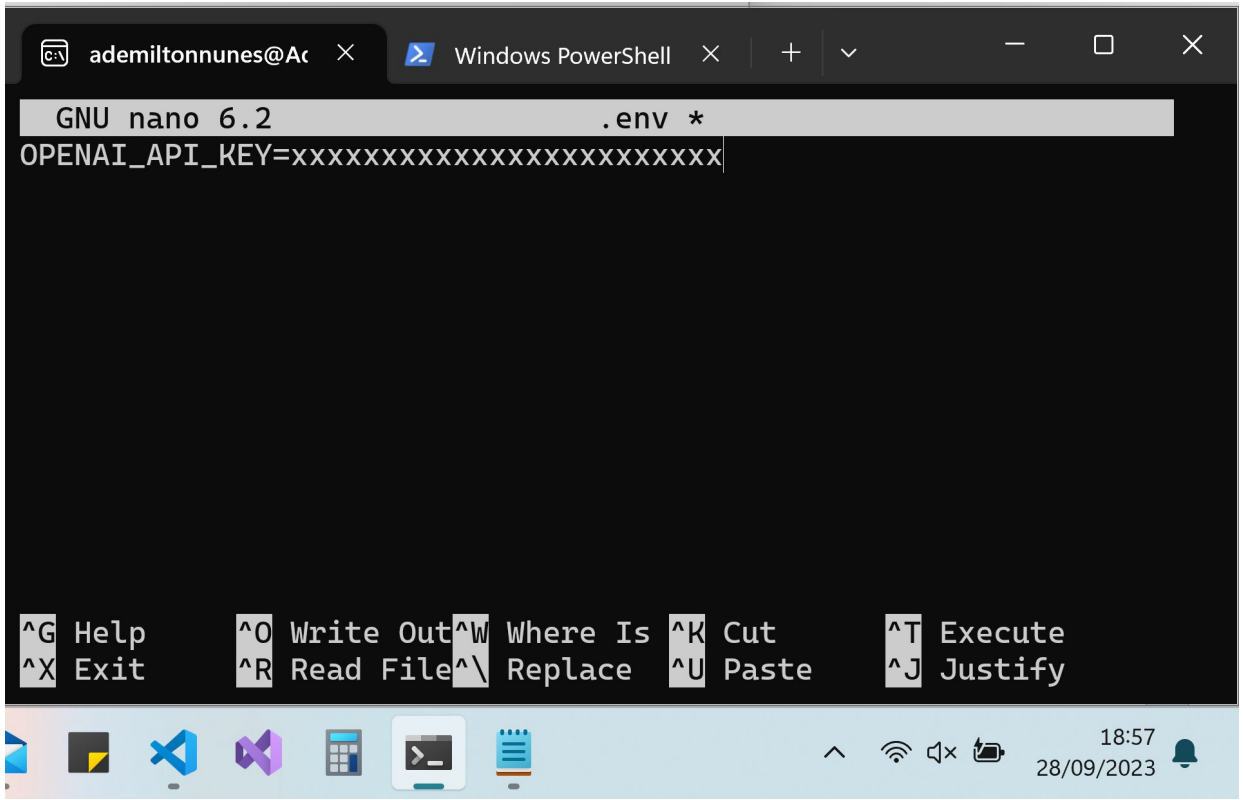
Node.js - Quick-Start

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

- npm install
- sudo apt install nodejs

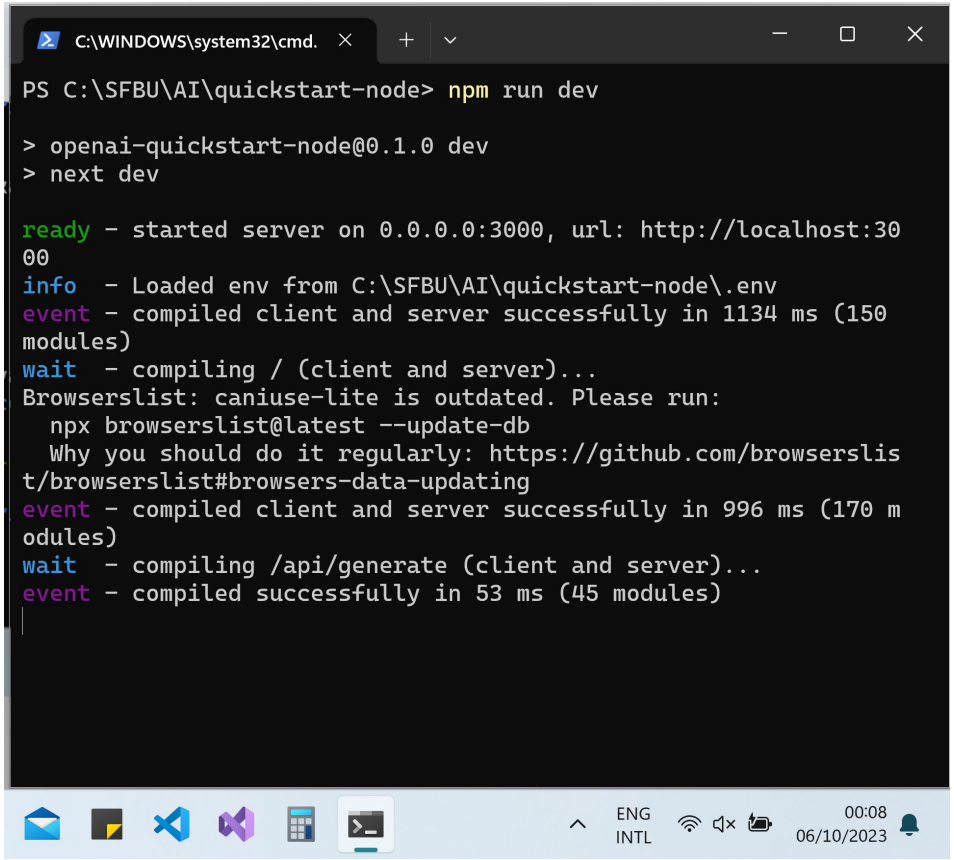
Node.js - Quick-Start

Configuring API keys



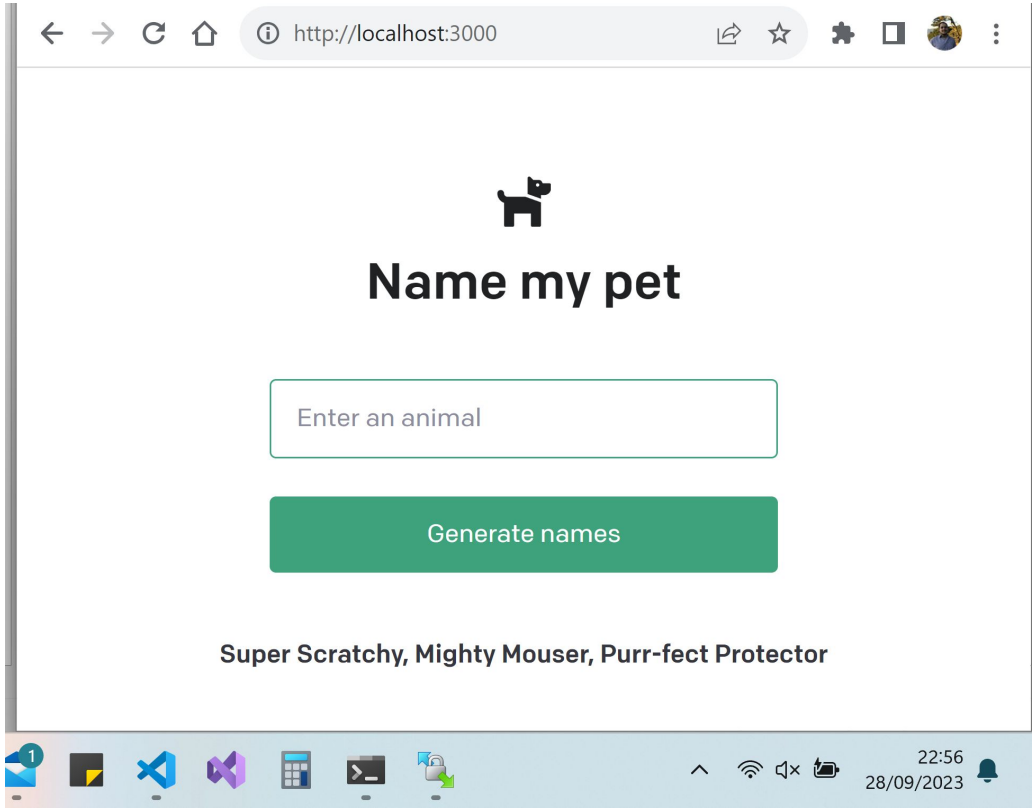
Node.js - Quick-Start

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



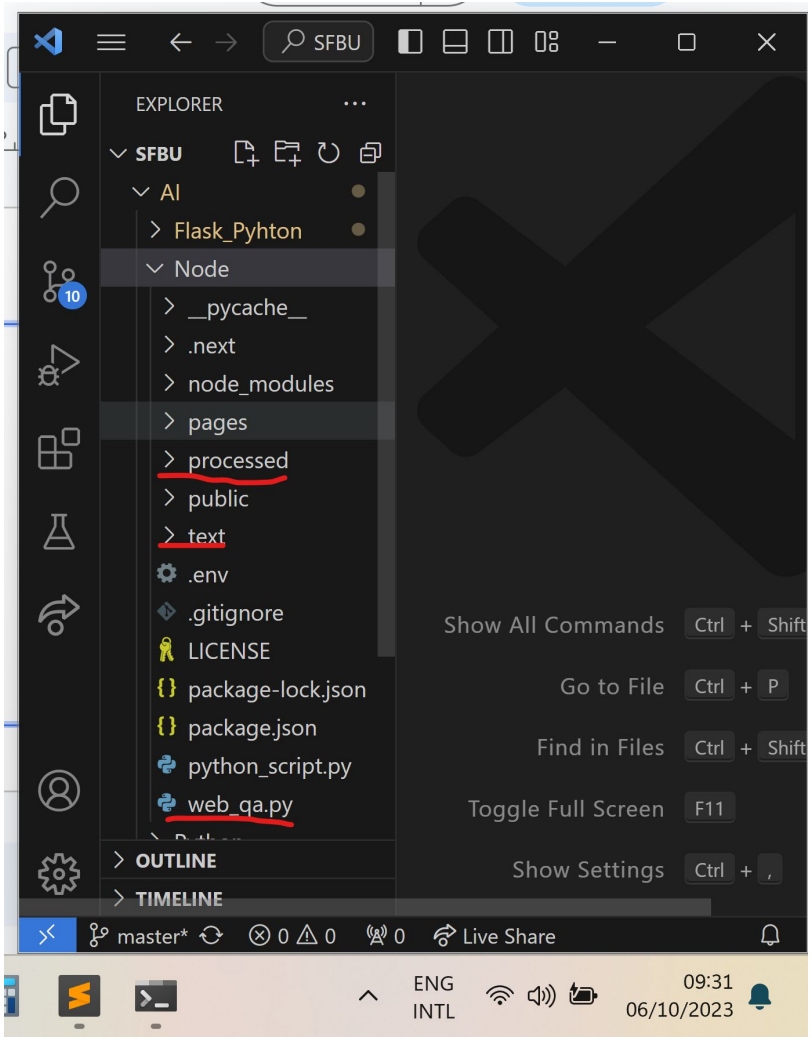
Node.js - Quick-Start

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



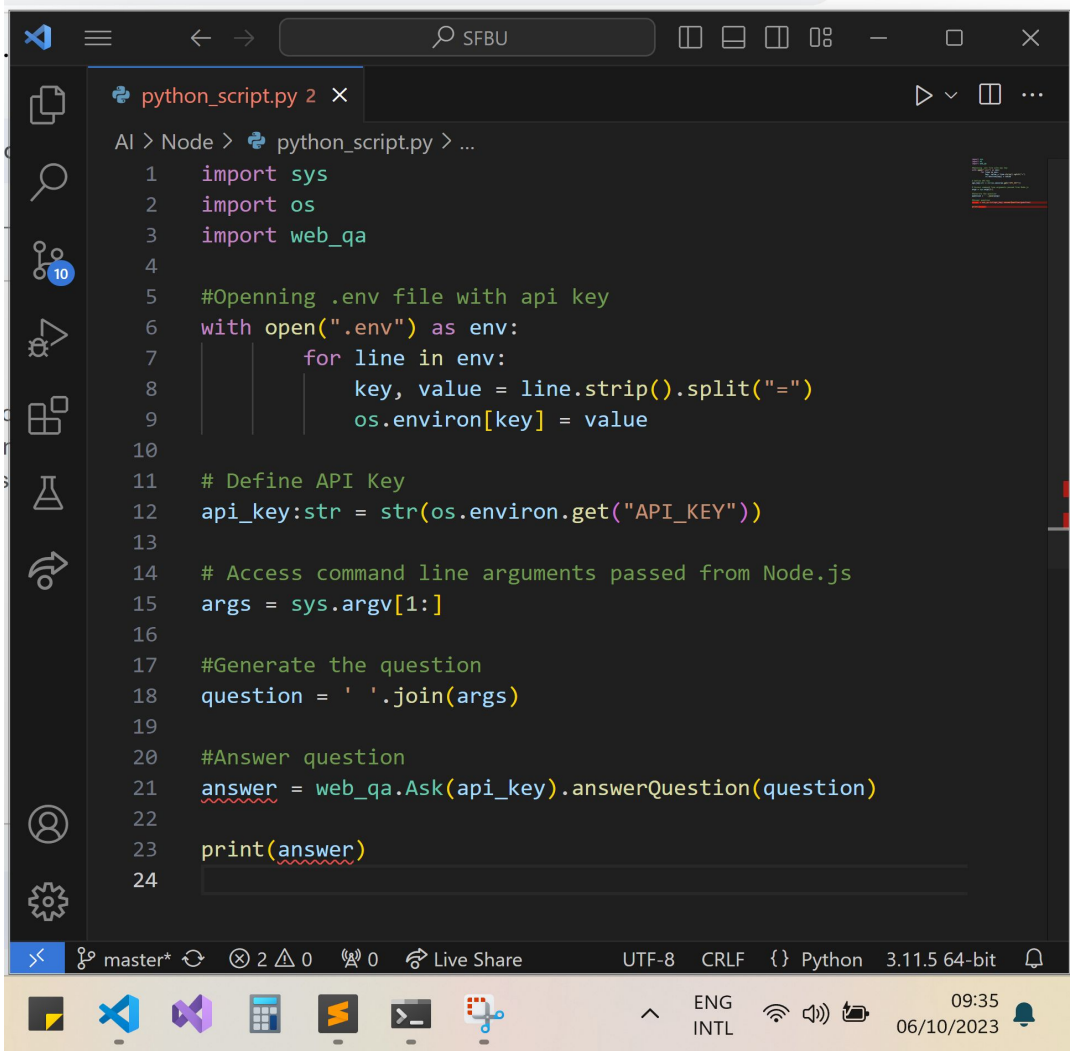
Node.js

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

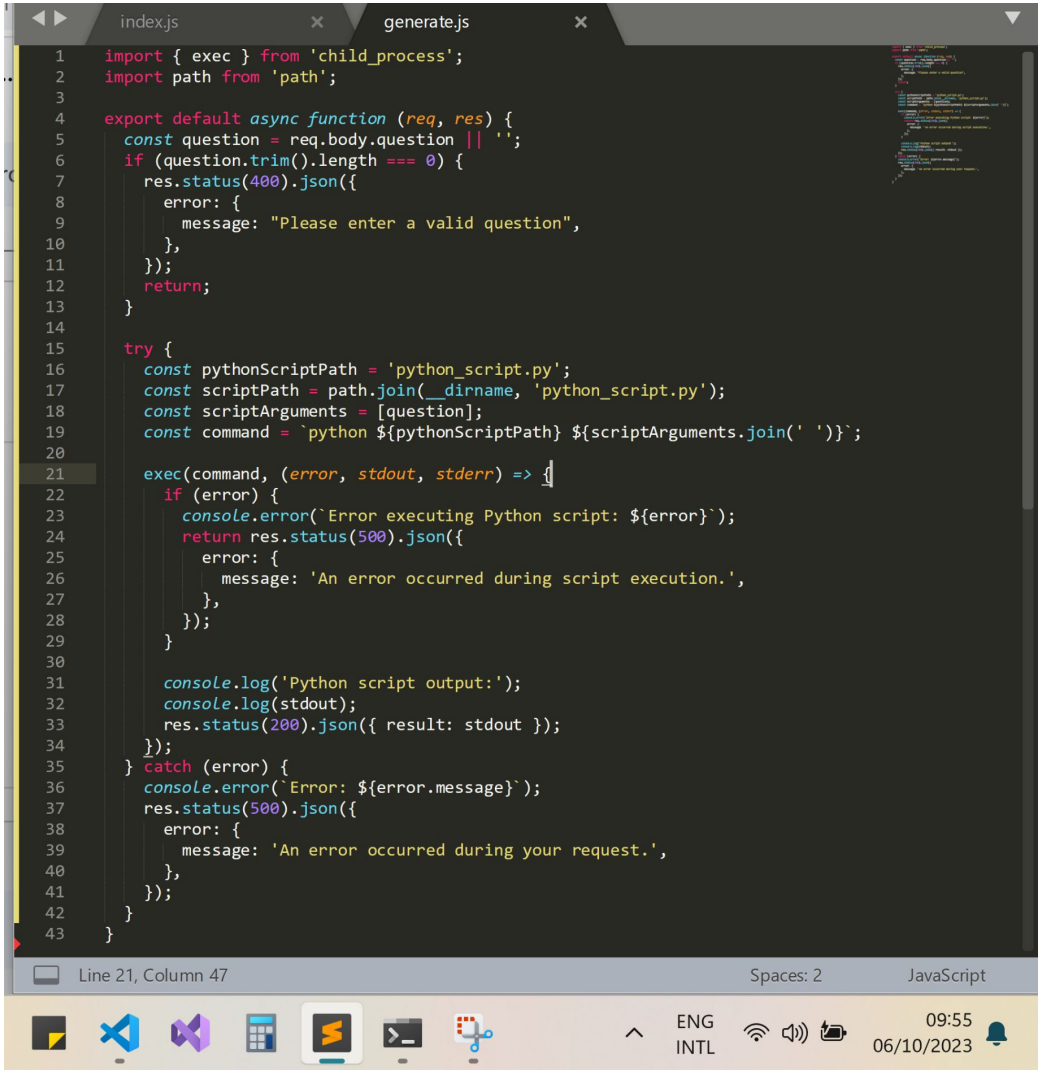
Because we are using two different technologies, Python and Node, I created a intermediate Python script to ask the questions received by the Node.js client. This script is responsible for putting the responses in the format that Node.js expects.



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.

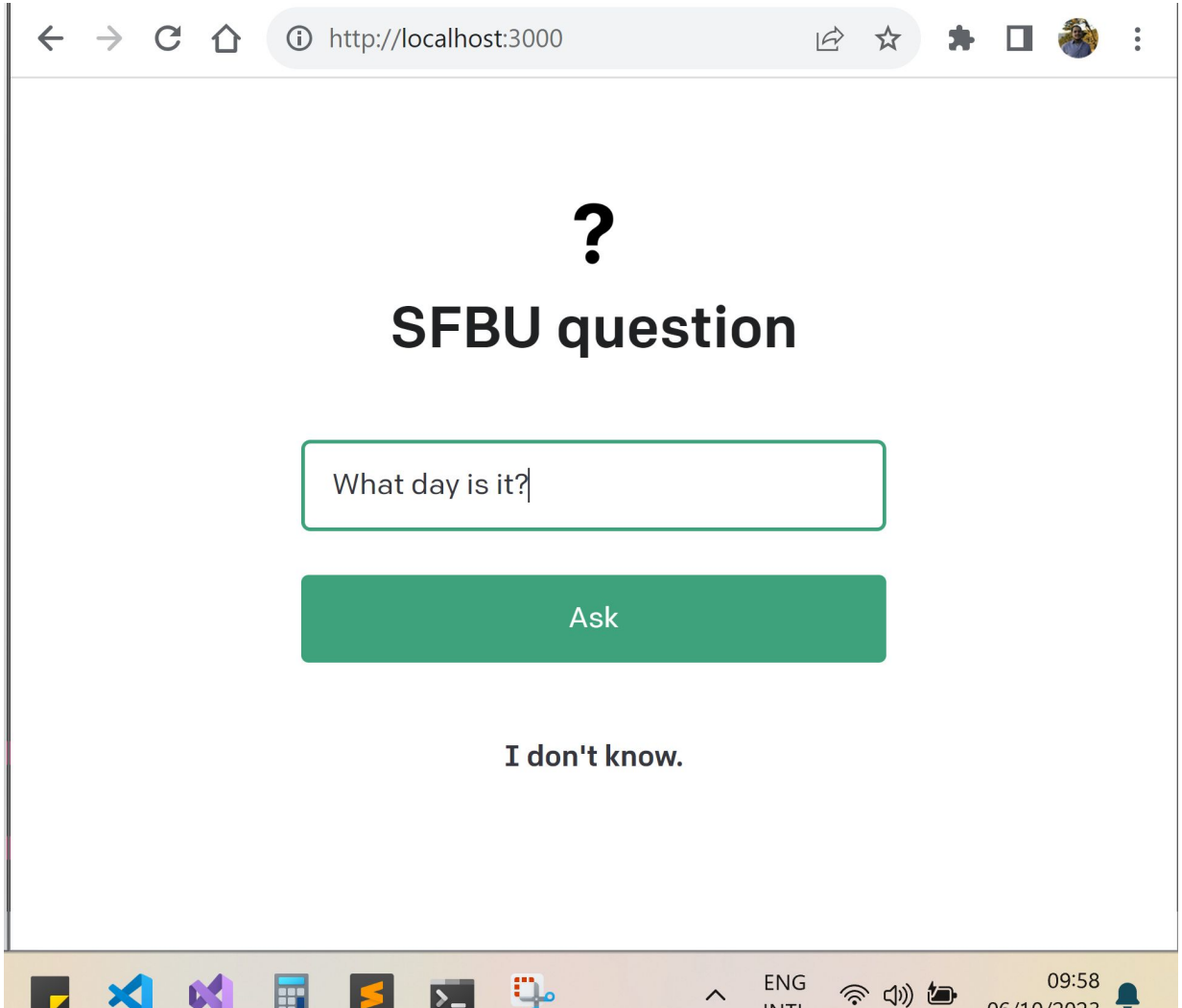
Node.js


index.js
generate.js
import { exec } from 'child_process';
import path from 'path';
export default async function (req, res) {
const question = req.body.question || '';
if (question.trim().length === 0) {
res.status(400).json({
error: {
message: "Please enter a valid question",
},
});
return;
}
try {
const pythonScriptPath = 'python_script.py';
const scriptPath = path.join(__dirname, 'python_script.py');
const scriptArguments = [question];
const command = `python ${pythonScriptPath} ${scriptArguments.join(' ')}`;
exec(command, (error, stdout, stderr) => {
if (error) {
console.error(`Error executing Python script: ${error}`);
return res.status(500).json({
error: {
message: 'An error occurred during script execution.',
},
});
}
console.log('Python script output:');
console.log(stdout);
res.status(200).json({ result: stdout });
});
} catch (error) {
console.error(`Error: ${error.message}`);
res.status(500).json({
error: {
message: 'An error occurred during your request.',
},
});
}
}
Line 21, Column 47
Spaces: 2
JavaScript
ENG
INTL
09:55
06/10/2023

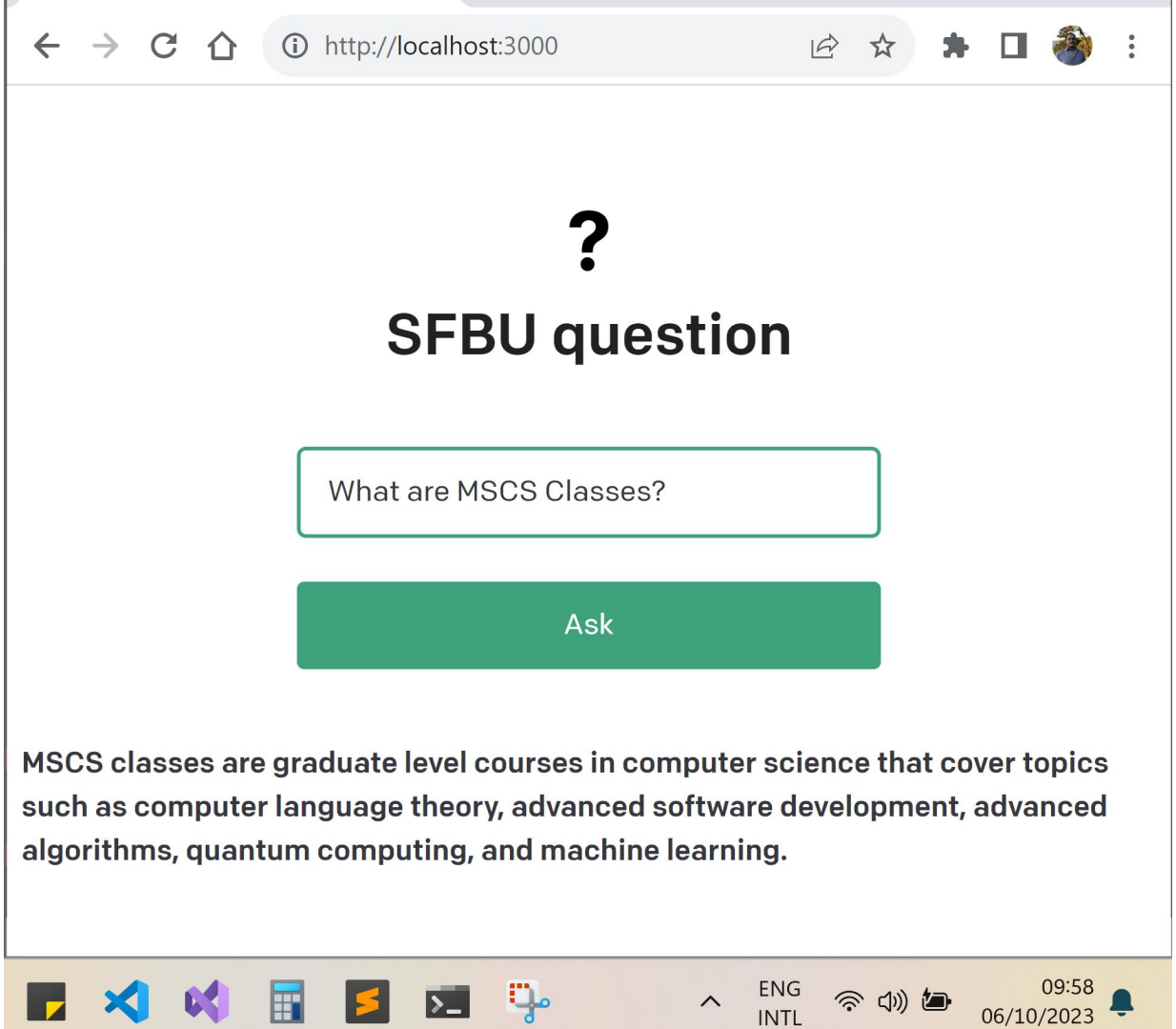
Node.js

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

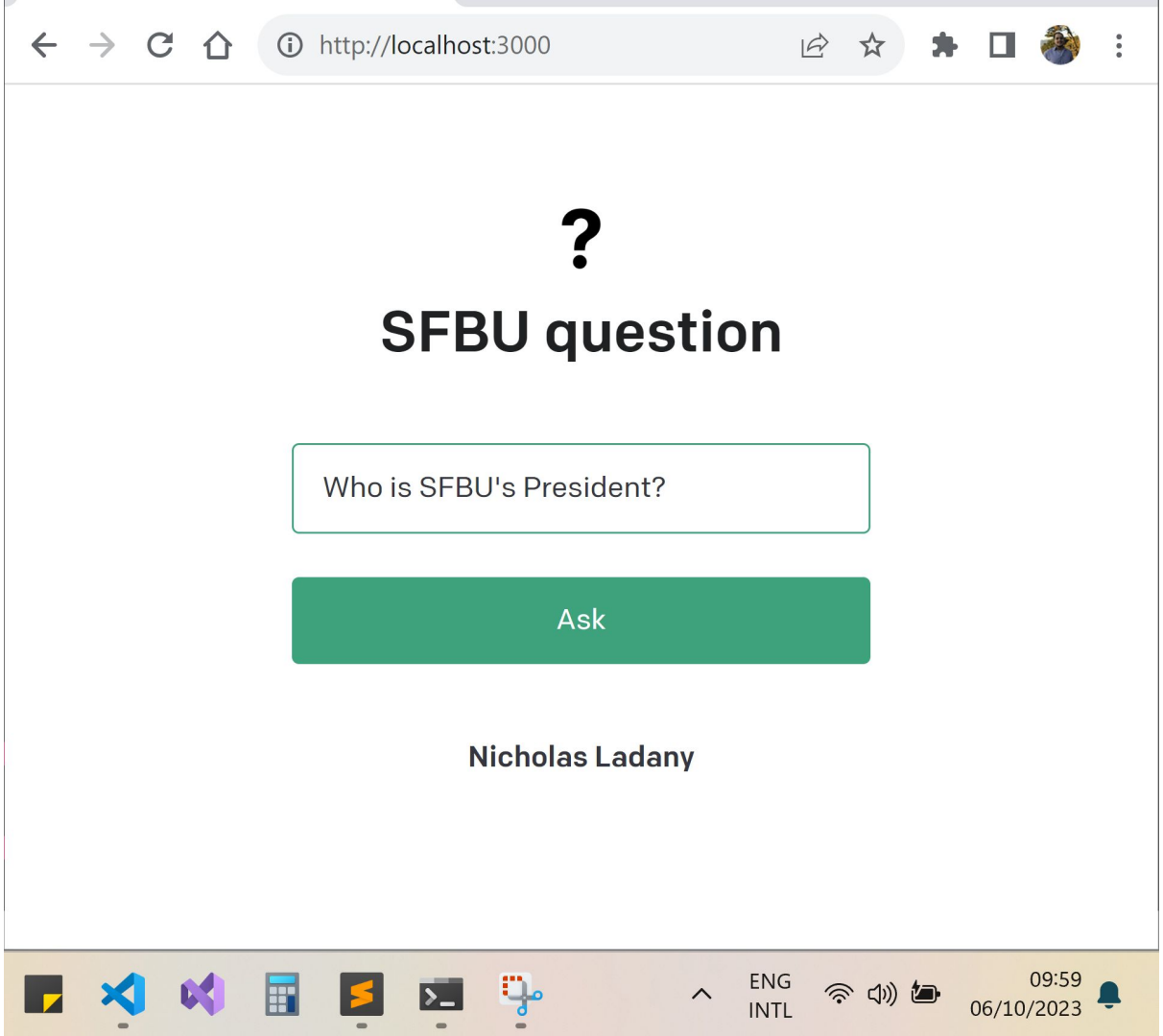
Node.js


http://localhost:3000
?
SFBU question
What day is it?
Ask
I don't know.
ENG
09:58

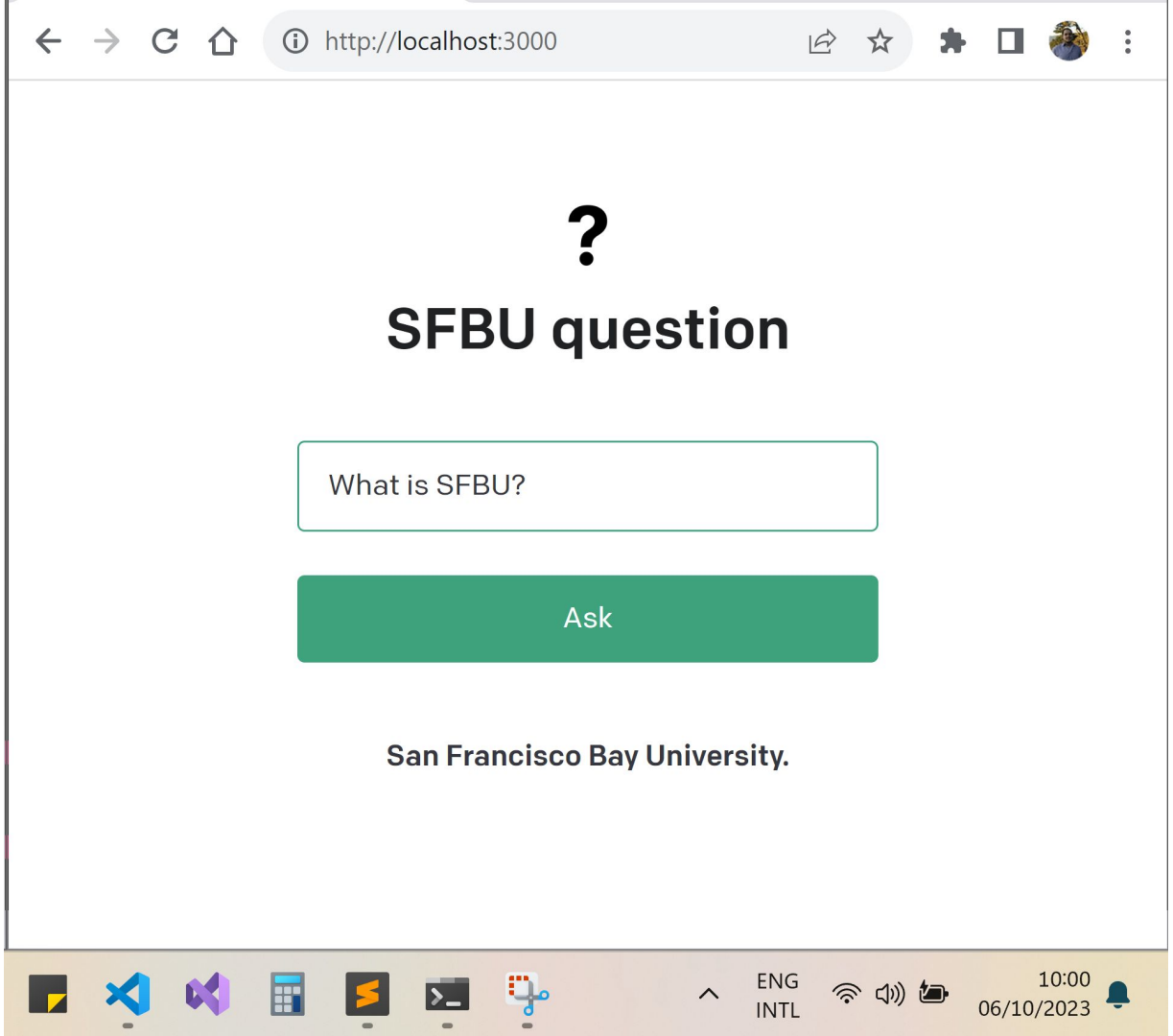
Node.js


http://localhost:3000
?
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.
ENG
INTL
09:58
06/10/2023

Node.js


http://localhost:3000
?
SFBU question
Who is SFBU's President?
Ask
Nicholas Ladany
ENG
INTL
09:59
06/10/2023

Node.js


http://localhost:3000
?
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.