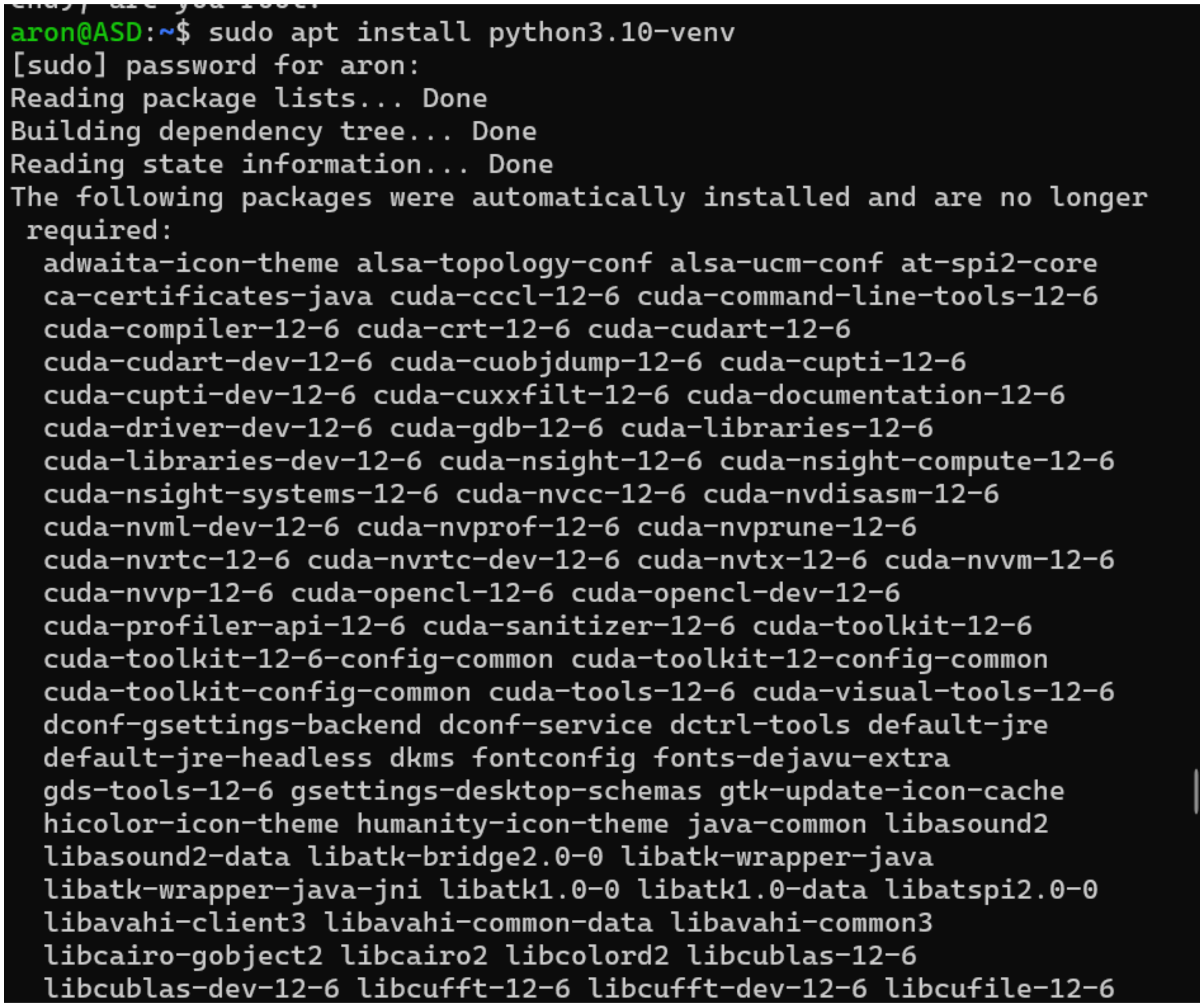
Name: Aron Sbhatu Dagniew

Id: 20073

Homework: Week 2 Homework 1 : ChatGPT project : Customer Support System: Use ChatGPT to build a web-based system that can answer questions about a website. - Step 1.2 : Web-based Solution (Python Flask webserver)

1. Setting Up the Project Environment

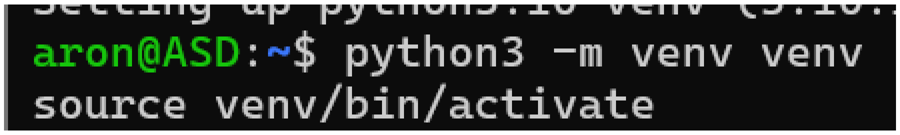
Step: Set up a virtual environment (optional but recommended)



```
aron@ASD:~$ sudo apt install python3.10-venv
[sudo] password for aron:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer
 required:
  adwaita-icon-theme alsa-topology-conf alsa-ucm-conf at-spi2-core
  ca-certificates-java cuda-cccl-12-6 cuda-command-line-tools-12-6
  cuda-compiler-12-6 cuda-crt-12-6 cuda-cudart-12-6
  cuda-cudart-dev-12-6 cuda-cuobjdump-12-6 cuda-cupti-12-6
  cuda-cupti-dev-12-6 cuda-cuxxfilt-12-6 cuda-documentation-12-6
  cuda-driver-dev-12-6 cuda-gdb-12-6 cuda-libraries-12-6
  cuda-libraries-dev-12-6 cuda-nsight-12-6 cuda-nsight-compute-12-6
  cuda-nsight-systems-12-6 cuda-nvcc-12-6 cuda-nvdisasm-12-6
  cuda-nvml-dev-12-6 cuda-nvprof-12-6 cuda-nvprune-12-6
  cuda-nvrtc-12-6 cuda-nvrtc-dev-12-6 cuda-nvtx-12-6 cuda-nvvm-12-6
  cuda-nvvp-12-6 cuda-opencl-12-6 cuda-opencl-dev-12-6
  cuda-profiler-api-12-6 cuda-sanitizer-12-6 cuda-toolkit-12-6
  cuda-toolkit-12-6-config-common cuda-toolkit-12-config-common
  cuda-toolkit-config-common cuda-tools-12-6 cuda-visual-tools-12-6
  dconf-gsettings-backend dconf-service dctrl-tools default-jre
  default-jre-headless dkms fontconfig fonts-dejavu-extra
  gds-tools-12-6 gsettings-desktop-schemas gtk-update-icon-cache
  hicolor-icon-theme humanity-icon-theme java-common libasound2
  libasound2-data libatk-bridge2.0-0 libatk-wrapper-java
  libatk-wrapper-java-jni libatk1.0-0 libatk1.0-data libatspi2.0-0
  libavahi-client3 libavahi-common-data libavahi-common3
  libcairo-gobject2 libcairo2 libcolord2 libcublas-12-6
  libcublas-dev-12-6 libcufft-12-6 libcufft-dev-12-6 libcufile-12-6
```

- Run the following commands to create and activate a virtual environment.

```
python3 -m venv venv
source venv/bin/activate
```

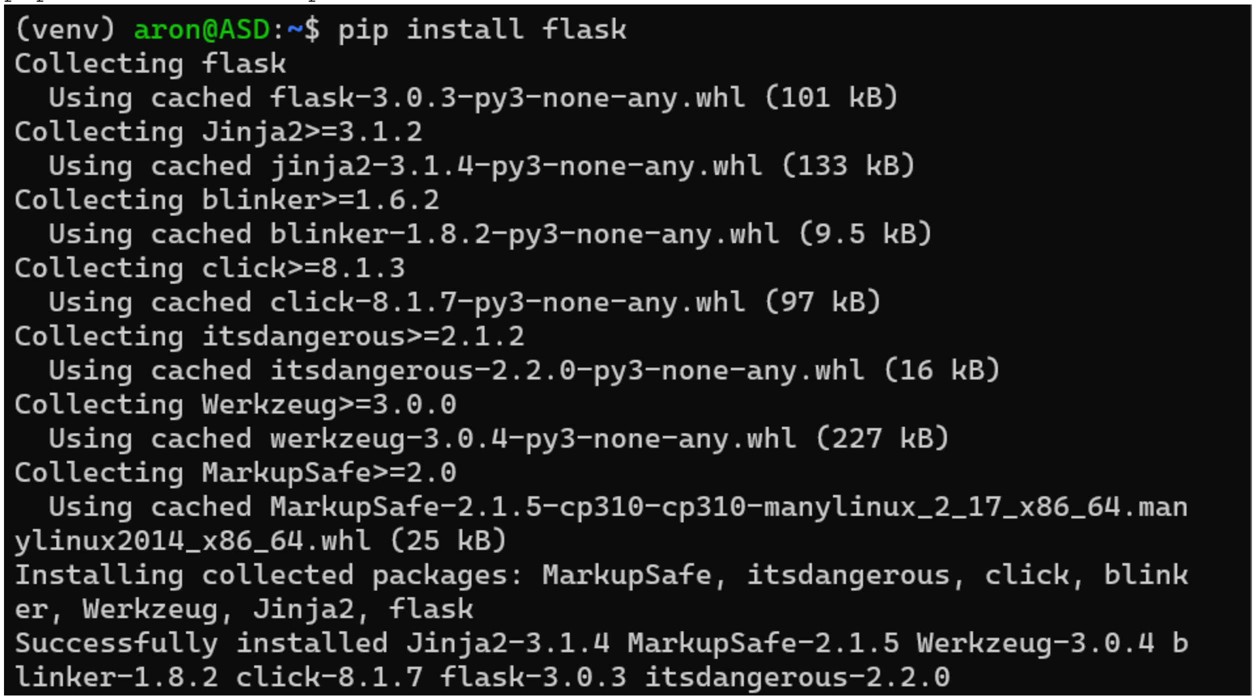


```
Setting up python3.10-venv (3.10.
aron@ASD:~$ python3 -m venv venv
source venv/bin/activate
```

Step: Install Flask and OpenAI dependencies

- Install Flask and the specific version of the OpenAI API required for your project.

```
pip install flask openai==0.28.0
```



```
(venv) aron@ASD:~$ pip install flask
Collecting flask
  Using cached flask-3.0.3-py3-none-any.whl (101 kB)
Collecting Jinja2>=3.1.2
  Using cached jinja2-3.1.4-py3-none-any.whl (133 kB)
Collecting blinker>=1.6.2
  Using cached blinker-1.8.2-py3-none-any.whl (9.5 kB)
Collecting click>=8.1.3
  Using cached click-8.1.7-py3-none-any.whl (97 kB)
Collecting itsdangerous>=2.1.2
  Using cached itsdangerous-2.2.0-py3-none-any.whl (16 kB)
Collecting Werkzeug>=3.0.0
  Using cached werkzeug-3.0.4-py3-none-any.whl (227 kB)
Collecting MarkupSafe>=2.0
  Using cached MarkupSafe-2.1.5-cp310-cp310-manylinux_2_17_x86_64.man
ylinux2014_x86_64.whl (25 kB)
Installing collected packages: MarkupSafe, itsdangerous, click, blink
er, Werkzeug, Jinja2, flask
Successfully installed Jinja2-3.1.4 MarkupSafe-2.1.5 Werkzeug-3.0.4 b
linker-1.8.2 click-8.1.7 flask-3.0.3 itsdangerous-2.2.0
```

2. Creating the Flask App

Step: Create the Flask application

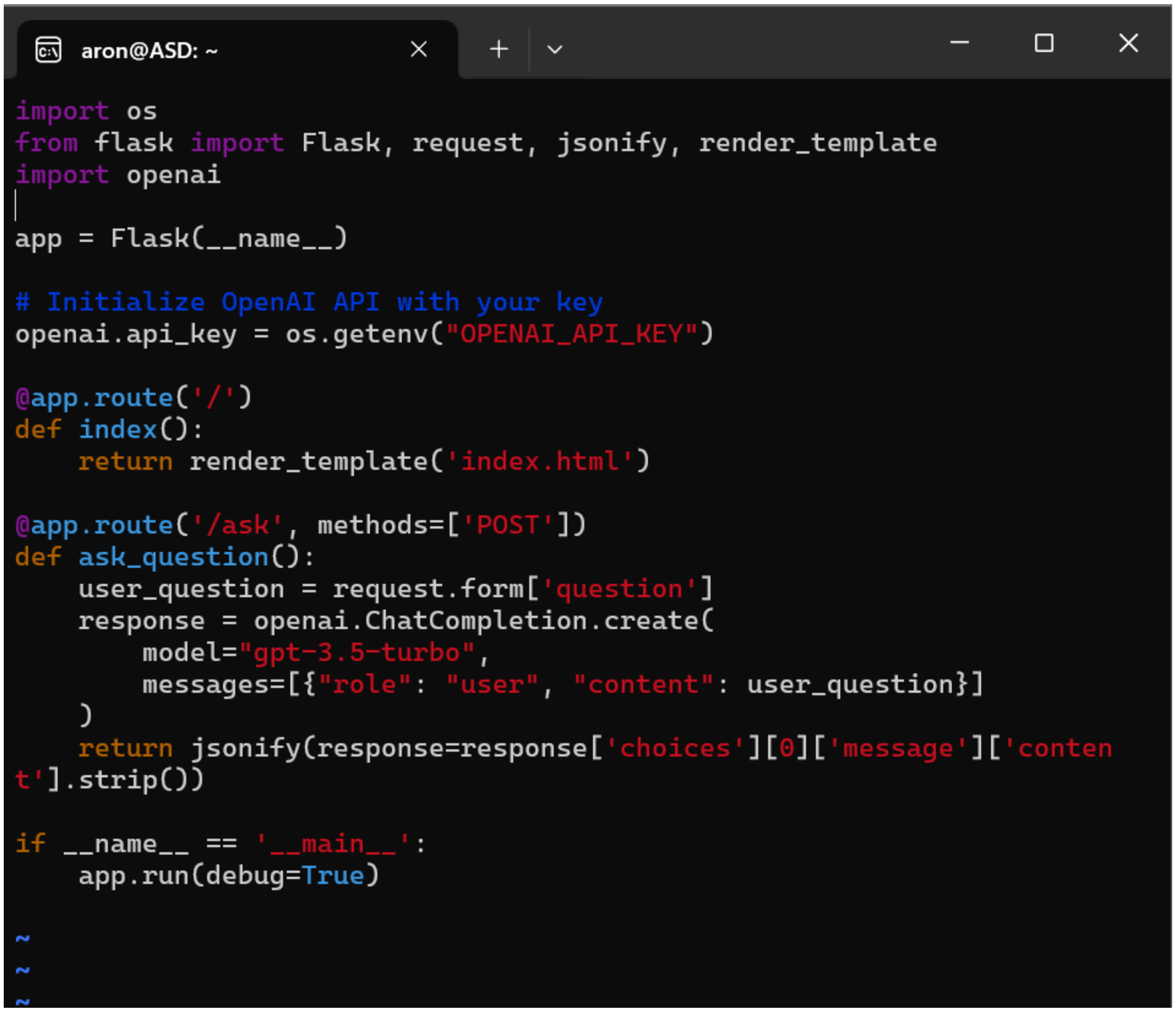
- Create a file named `app.py` in your project directory.

```
vim app.py
```

- Add the following code to `app.py`:

```
from flask import Flask, request, jsonify, render_template
import openai

app = Flask(__name__)

# Initialize OpenAI API with your key
openai.api_key = "your-api-key"
```

```
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/ask', methods=['POST'])
def ask_question():
    user_question = request.form['question']
    response = openai.Completion.create(
        engine="text-davinci-003",
        prompt=user_question,
        max_tokens=100
    )
    return jsonify(response=response.choices[0].text.strip())

if __name__ == '__main__':
    app.run(debug=True)
```



```
import os
from flask import Flask, request, jsonify, render_template
import openai

app = Flask(__name__)

# Initialize OpenAI API with your key
openai.api_key = os.getenv("OPENAI_API_KEY")

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/ask', methods=['POST'])
def ask_question():
    user_question = request.form['question']
    response = openai.ChatCompletion.create(
        model="gpt-3.5-turbo",
        messages=[{"role": "user", "content": user_question}]
    )
    return jsonify(response=response['choices'][0]['message']['conten
t'].strip())

if __name__ == '__main__':
    app.run(debug=True)

~
~
~
```

3. Building the HTML Interface

Step: Create the HTML template

- Create a folder named `templates` in your project directory:

```
mkdir templates
```

```
aron@ASD:~$ mkdir templates
```

- Create an `index.html` file inside the `templates` folder and add the provided ChatGPT-like interface code.

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Customer Support - Ask ChatGPT</title>

    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.0.0-beta3/css/all.min.css">

    <style>

        body {

            margin: 0;

            padding: 0;

            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

            background-color: #f5f5f5;

            display: flex;

            justify-content: center;

            align-items: center;

            height: 100vh;

        }
```

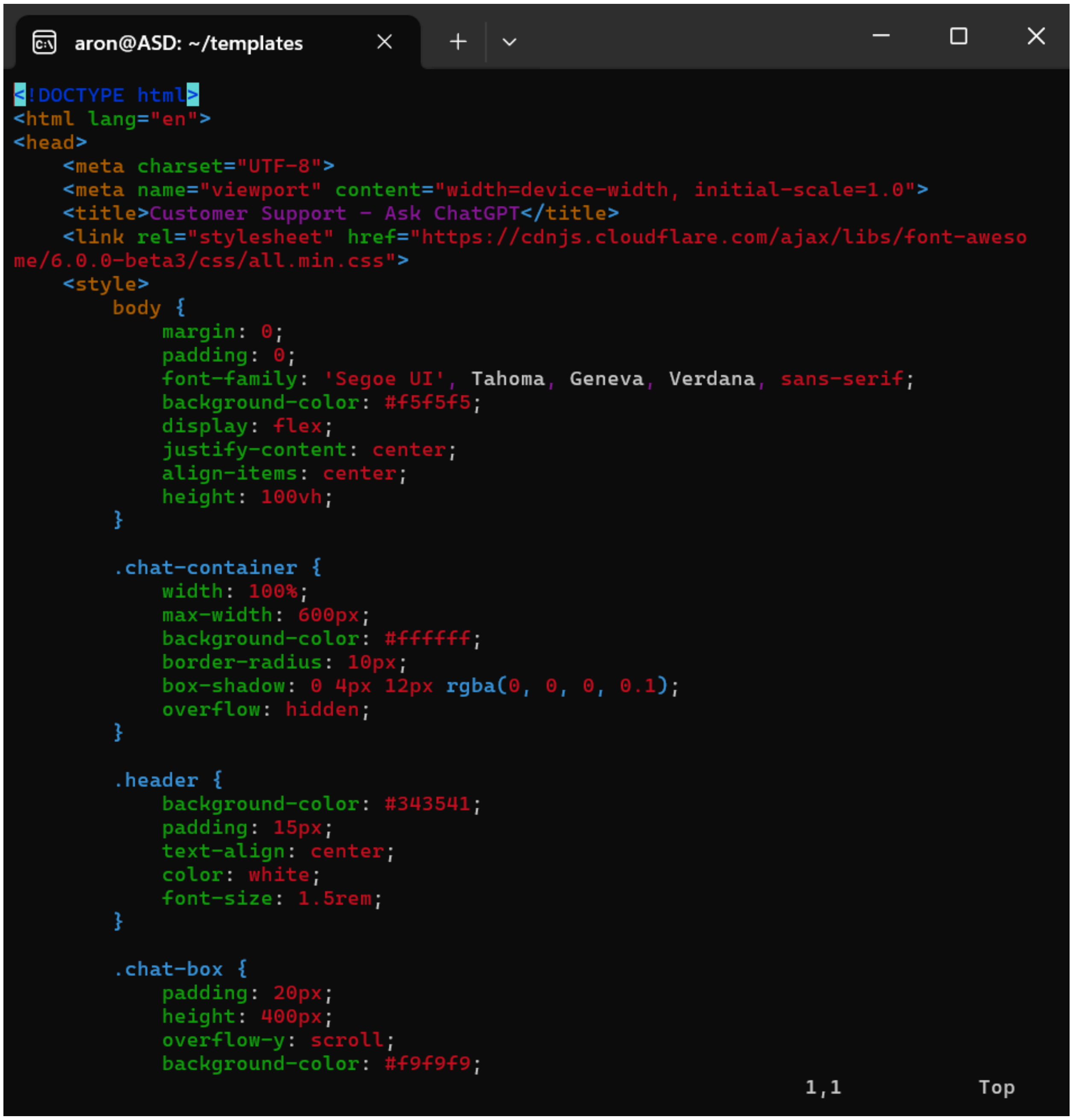
```
.chat-container {

    width: 100%;

    max-width: 600px;

    background-color: #ffffff;

    border-radius: 10px;

    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);

    overflow: hidden;

}


.header {

    background-color: #343541;

    padding: 15px;

    text-align: center;

    color: white;

    font-size: 1.5rem;

}


.chat-box {

    padding: 20px;

    height: 400px;

    overflow-y: scroll;

    background-color: #f9f9f9;

}


.chat-box .message {

    margin-bottom: 20px;
```

```
}


.chat-box .message p {

    margin: 0;

    padding: 10px;

    border-radius: 5px;

    background-color: #e0e0e0;

}


.chat-box .user-message p {

    background-color: #d1e7ff;

    text-align: right;

}


.chat-box .gpt-message p {

    background-color: #f1f1f1;

}


.input-area {

    display: flex;

    border-top: 1px solid #ddd;

    background-color: #fff;

}


.input-area input[type="text"] {

    width: 100%;
```

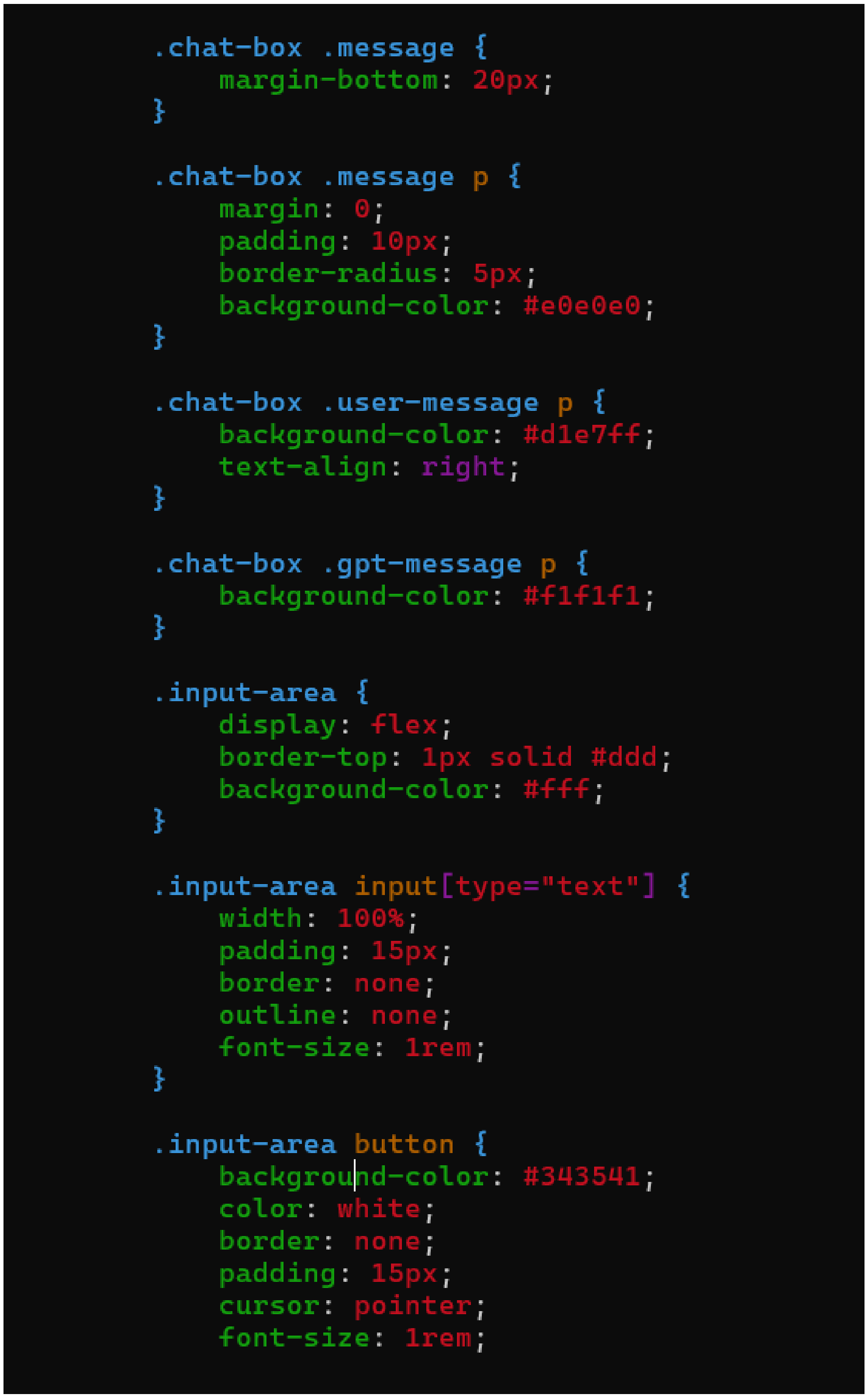
```
            padding: 15px;

            border: none;

            outline: none;

            font-size: 1rem;

        }


        .input-area button {

            background-color: #343541;

            color: white;

            border: none;

            padding: 15px;

            cursor: pointer;

            font-size: 1rem;

        }


        .input-area button:hover {

            background-color: #46485f;

        }


        .input-area button i {

            margin-right: 5px;

        }

    </style>

</head>

<body>

    <div class="chat-container">
```

```
        <div class="header">

            <i class="fas fa-robot"></i> Customer Support - Ask ChatGPT

        </div>

        <div class="chat-box" id="chat-box">

            <!-- Chat history goes here -->

        </div>

        <div class="input-area">

            <input type="text" id="question" name="question"
placeholder="Type your question..." required>

            <button id="sendButton"><i class="fas fa-paper-plane"></i>
Send</button>

        </div>

    </div>



    <script>

        const form = document.getElementById('sendButton');

        const chatBox = document.getElementById('chat-box');



        form.addEventListener('click', async function (event) {

            event.preventDefault();

            const question = document.getElementById('question').value;

            if (question.trim() === "") return;



            // Display user message

            chatBox.innerHTML += `<div class="message user-
message"><p>${question}</p></div>`;

            document.getElementById('question').value = "";  // Clear the
input
```

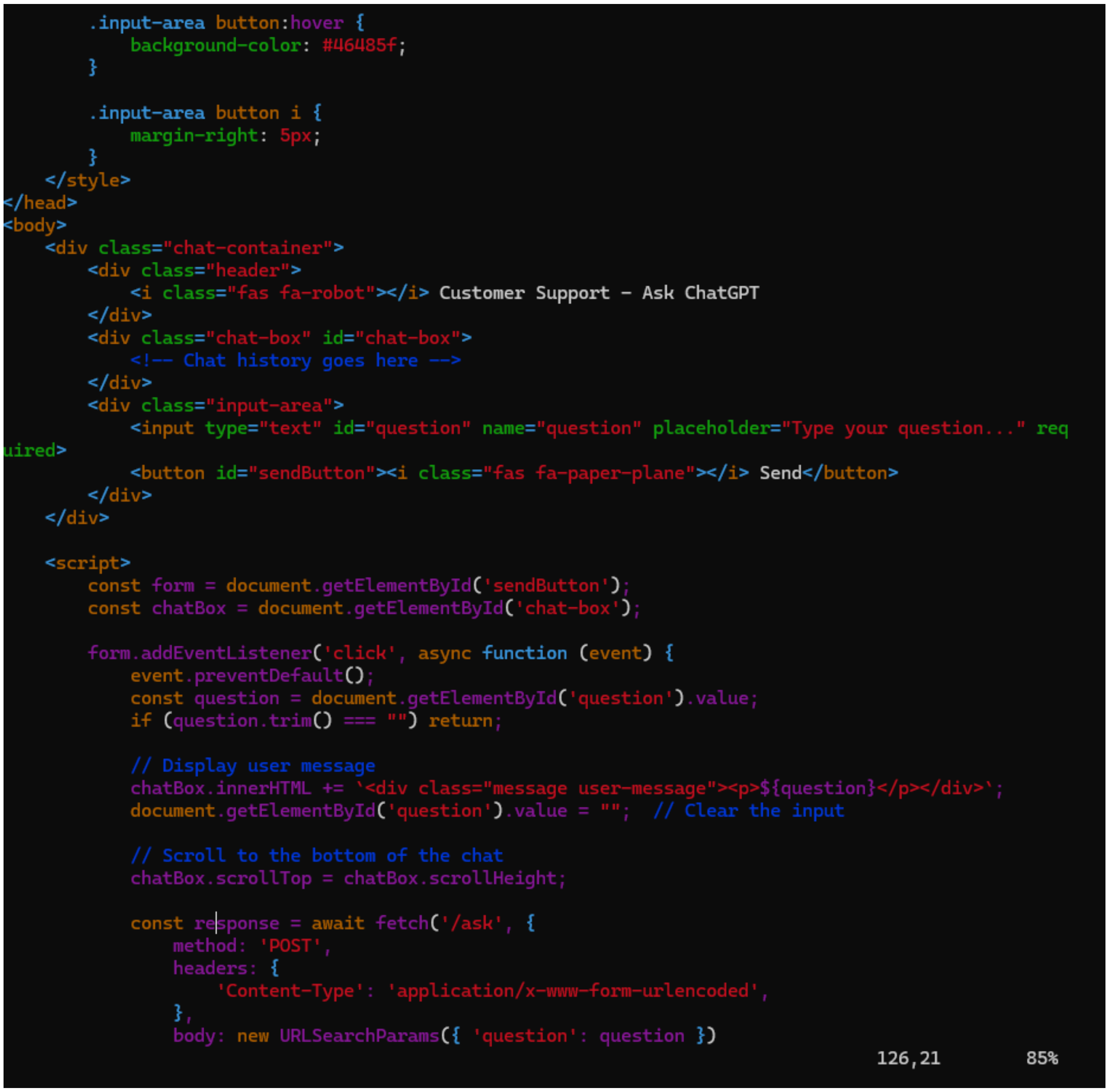
```
            // Scroll to the bottom of the chat
            chatBox.scrollTop = chatBox.scrollHeight;

            const response = await fetch('/ask', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/x-www-form-urlencoded',
                },
                body: new URLSearchParams({ 'question': question })
            });

            const data = await response.json();

            // Display ChatGPT's response
            chatBox.innerHTML += `<div class="message gpt-
message"><p>${data.response}</p></div>`;

            // Scroll to the bottom of the chat
            chatBox.scrollTop = chatBox.scrollHeight;
        });
    </script>
</body>
</html>
```



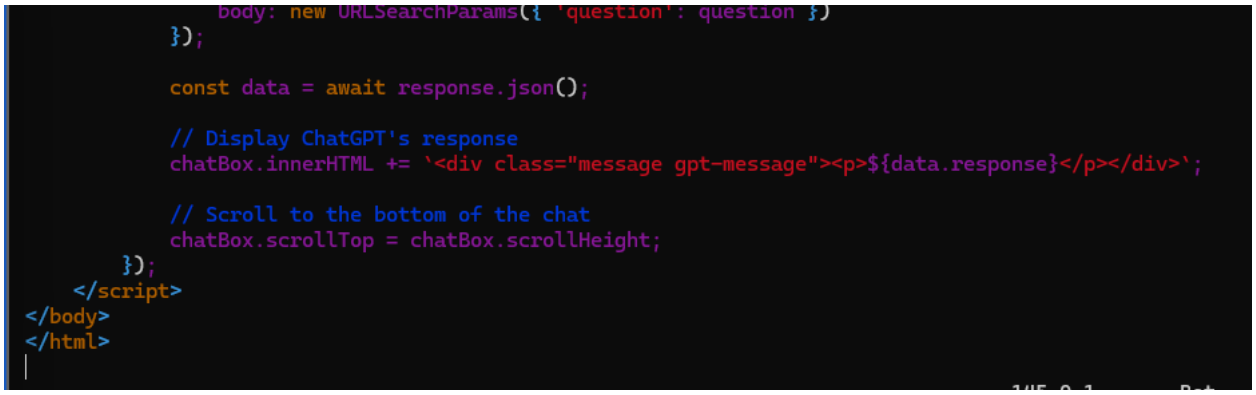
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Customer Support - Ask ChatGPT</title>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-aweso
me/6.0.0-beta3/css/all.min.css">
    <style>
        body {
            margin: 0;
            padding: 0;
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            background-color: #f5f5f5;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
        }

        .chat-container {
            width: 100%;
            max-width: 600px;
            background-color: #ffffff;
            border-radius: 10px;
            box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);
            overflow: hidden;
        }

        .header {
            background-color: #343541;
            padding: 15px;
            text-align: center;
            color: white;
            font-size: 1.5rem;
        }

        .chat-box {
            padding: 20px;
            height: 400px;
            overflow-y: scroll;
            background-color: #f9f9f9;
                                                              1,1           Top
```



```css
.chat-box .message {
    margin-bottom: 20px;
}

.chat-box .message p {
    margin: 0;
    padding: 10px;
    border-radius: 5px;
    background-color: #e0e0e0;
}

.chat-box .user-message p {
    background-color: #d1e7ff;
    text-align: right;
}

.chat-box .gpt-message p {
    background-color: #f1f1f1;
}

.input-area {
    display: flex;
    border-top: 1px solid #ddd;
    background-color: #fff;
}

.input-area input[type="text"] {
    width: 100%;
    padding: 15px;
    border: none;
    outline: none;
    font-size: 1rem;
}

.input-area button {
    background-color: #343541;
    color: white;
    border: none;
    padding: 15px;
    cursor: pointer;
    font-size: 1rem;
```



```
        .input-area button:hover {
            background-color: #46485f;
        }

        .input-area button i {
            margin-right: 5px;
        }
    </style>
</head>
<body>
    <div class="chat-container">
        <div class="header">
            <i class="fas fa-robot"></i> Customer Support - Ask ChatGPT
        </div>
        <div class="chat-box" id="chat-box">
            <!-- Chat history goes here -->
        </div>
        <div class="input-area">
            <input type="text" id="question" name="question" placeholder="Type your question..." req
uired>
            <button id="sendButton"><i class="fas fa-paper-plane"></i> Send</button>
        </div>
    </div>

    <script>
        const form = document.getElementById('sendButton');
        const chatBox = document.getElementById('chat-box');

        form.addEventListener('click', async function (event) {
            event.preventDefault();
            const question = document.getElementById('question').value;
            if (question.trim() === "") return;

            // Display user message
            chatBox.innerHTML += `<div class="message user-message"><p>${question}</p></div>`;
            document.getElementById('question').value = "";  // Clear the input

            // Scroll to the bottom of the chat
            chatBox.scrollTop = chatBox.scrollHeight;

            const response = await fetch('/ask', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/x-www-form-urlencoded',
                },
                body: new URLSearchParams({ 'question': question })
                                                                                  126,21          85%
```



```
                body: new URLSearchParams({ 'question': question })
            });

            const data = await response.json();

            // Display ChatGPT's response
            chatBox.innerHTML += `<div class="message gpt-message"><p>${data.response}</p></div>`;

            // Scroll to the bottom of the chat
            chatBox.scrollTop = chatBox.scrollHeight;
        });
    </script>
</body>
</html>
```

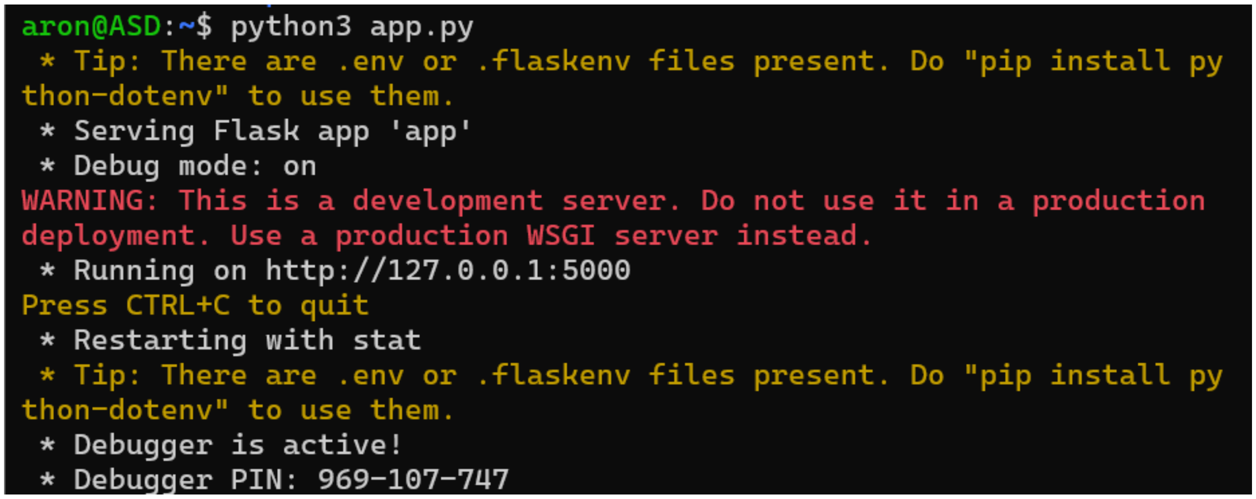
4. Running the Flask Application

Step: Running the app

- In your terminal, navigate to the project directory and run the Flask app.

```
python app.py
```

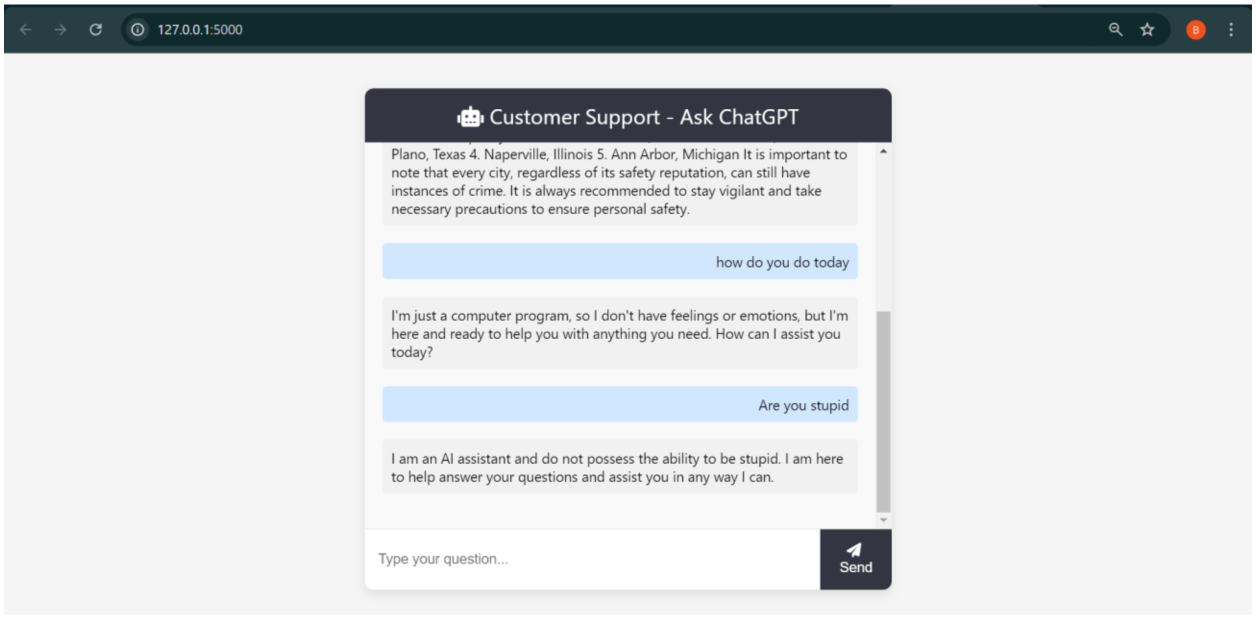
- You should see output indicating the app is running on http://127.0.0.1:5000/.



```
aron@ASD:~$ python3 app.py
 * Tip: There are .env or .flaskenv files present. Do "pip install py
thon-dotenv" to use them.
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production
deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Tip: There are .env or .flaskenv files present. Do "pip install py
thon-dotenv" to use them.
 * Debugger is active!
 * Debugger PIN: 969-107-747
```

Step: Open the app in a browser

- Open a browser and navigate to `http://127.0.0.1:5000/` to see the web interface.



Customer Support - Ask ChatGPT

Plano, Texas 4. Naperville, Illinois 5. Ann Arbor, Michigan It is important to note that every city, regardless of its safety reputation, can still have instances of crime. It is always recommended to stay vigilant and take necessary precautions to ensure personal safety.

how do you do today

I'm just a computer program, so I don't have feelings or emotions, but I'm here and ready to help you with anything you need. How can I assist you today?

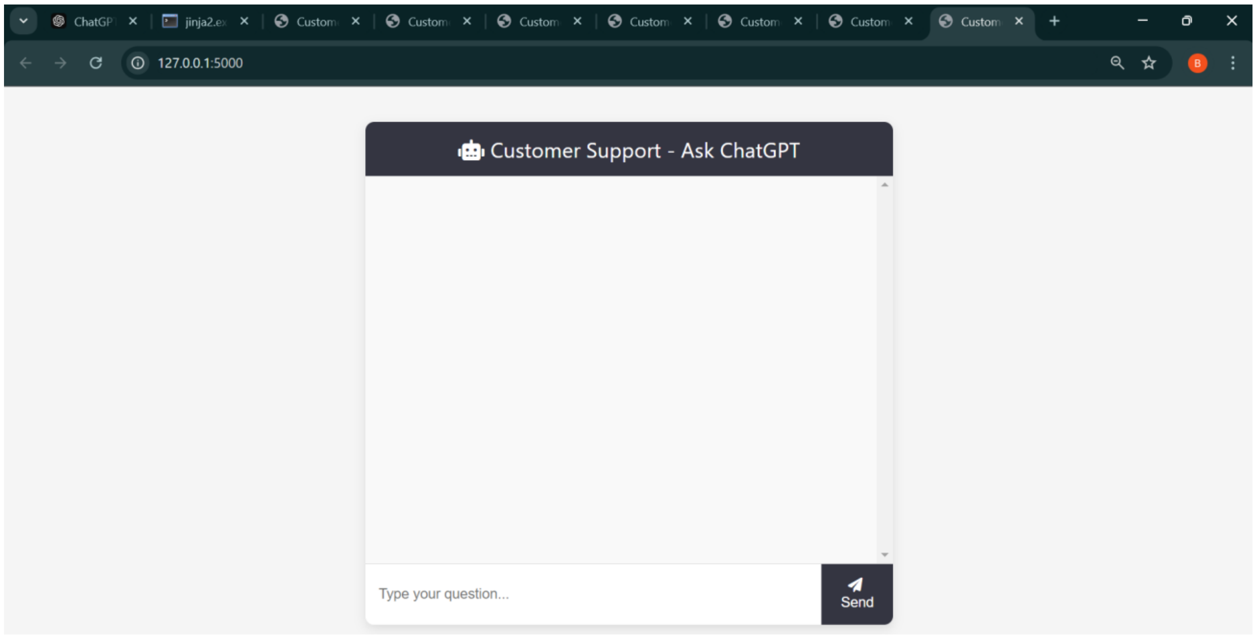
Are you stupid

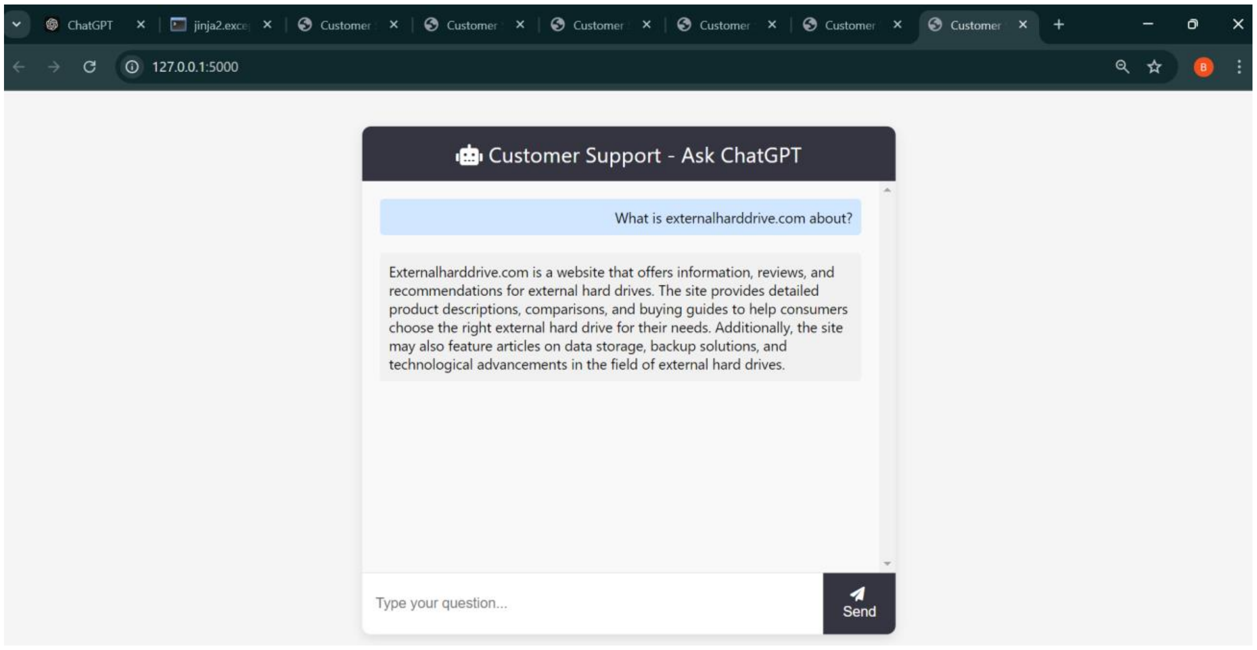
I am an AI assistant and do not possess the ability to be stupid. I am here to help answer your questions and assist you in any way I can.

5. Testing the ChatGPT Integration

Step: Ask a question

- Type a question into the input box (e.g., "What is externalharddrive.com about?") and click the "Send" button.





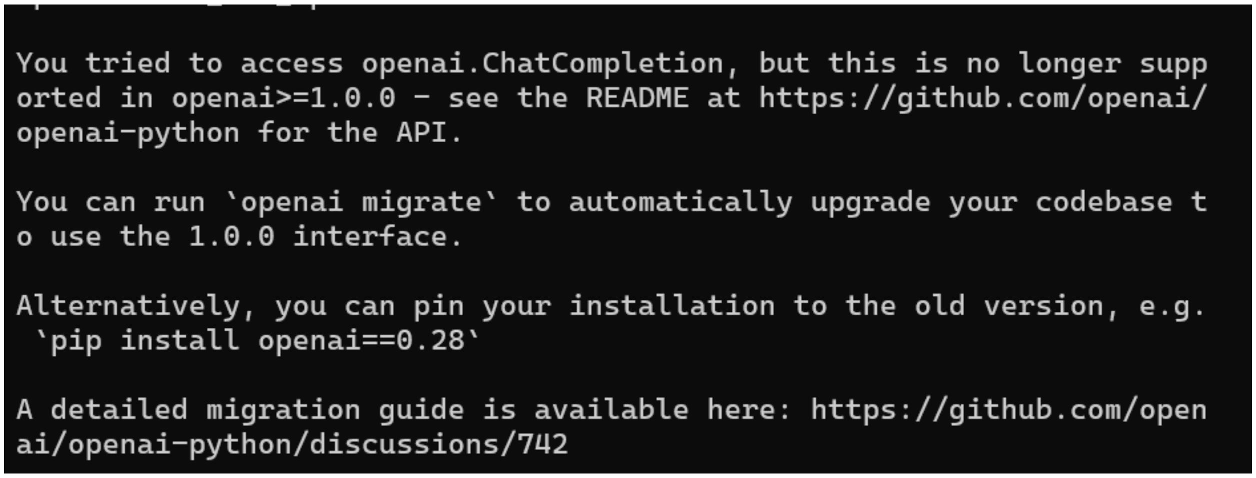
What is externalharddrive.com about?

Externalharddrive.com is a website that offers information, reviews, and recommendations for external hard drives. The site provides detailed product descriptions, comparisons, and buying guides to help consumers choose the right external hard drive for their needs. Additionally, the site may also feature articles on data storage, backup solutions, and technological advancements in the field of external hard drives.

6. Documenting Errors and Solutions

Step: Capture common issues and fixes

- If you encountered any errors during setup (e.g., `TemplateNotFound` or `OpenAI API` errors), document how you solved them.



```
You tried to access openai.ChatCompletion, but this is no longer supp
orted in openai>=1.0.0 - see the README at https://github.com/openai/
openai-python for the API.

You can run `openai migrate` to automatically upgrade your codebase t
o use the 1.0.0 interface.

Alternatively, you can pin your installation to the old version, e.g.
 `pip install openai==0.28`

A detailed migration guide is available here: https://github.com/open
ai/openai-python/discussions/742
```

1. Error: `TemplateNotFound`

Error Message:

```
jinja2.exceptions.TemplateNotFound: index.html
```

Cause: This error occurred because Flask was unable to locate the `index.html` file. Flask looks for templates in a folder named `templates` by default.

Solution: To resolve this issue, I ensured that:

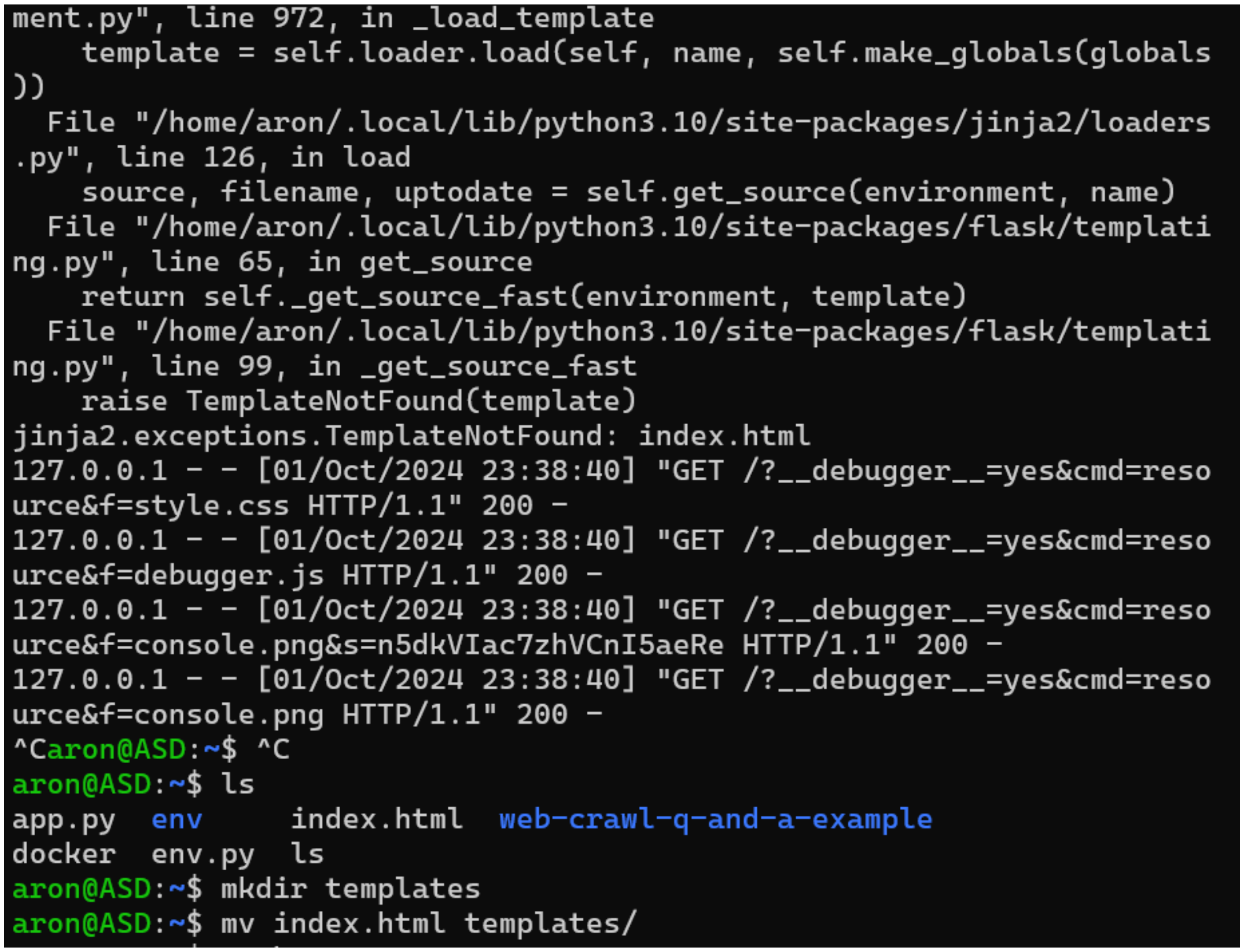
- The `index.html` file was placed in a folder named `templates` inside the project directory.
- The file was named correctly (`index.html` with no typos).

Steps Taken:

1. I created the `templates` folder using:

   ```
   mkdir templates
   ```

2. I placed the `index.html` file inside the `templates` folder.



```
ment.py", line 972, in _load_template
    template = self.loader.load(self, name, self.make_globals(globals
))
  File "/home/aron/.local/lib/python3.10/site-packages/jinja2/loaders
.py", line 126, in load
    source, filename, uptodate = self.get_source(environment, name)
  File "/home/aron/.local/lib/python3.10/site-packages/flask/templati
ng.py", line 65, in get_source
    return self._get_source_fast(environment, template)
  File "/home/aron/.local/lib/python3.10/site-packages/flask/templati
ng.py", line 99, in _get_source_fast
    raise TemplateNotFound(template)
jinja2.exceptions.TemplateNotFound: index.html
127.0.0.1 - - [01/Oct/2024 23:38:40] "GET /?__debugger__=yes&cmd=reso
urce&f=style.css HTTP/1.1" 200 -
127.0.0.1 - - [01/Oct/2024 23:38:40] "GET /?__debugger__=yes&cmd=reso
urce&f=debugger.js HTTP/1.1" 200 -
127.0.0.1 - - [01/Oct/2024 23:38:40] "GET /?__debugger__=yes&cmd=reso
urce&f=console.png&s=n5dkVIac7zhVCnI5aeRe HTTP/1.1" 200 -
127.0.0.1 - - [01/Oct/2024 23:38:40] "GET /?__debugger__=yes&cmd=reso
urce&f=console.png HTTP/1.1" 200 -
^Caron@ASD:~$ ^C
aron@ASD:~$ ls
app.py  env     index.html  web-crawl-q-and-a-example
docker  env.py  ls
aron@ASD:~$ mkdir templates
aron@ASD:~$ mv index.html templates/
```

2. Error: OpenAI API Deprecation (`APIRemovedInV1`)

Error Message:

```
openai.lib._old_api.APIRemovedInV1:
You tried to access openai.Completion, but this is no longer supported in
openai>=1.0.0.
```

Cause: This error occurred because the method `openai.Completion.create` is deprecated in OpenAI versions 1.0.0 and later. I was using a newer version of the OpenAI API that no longer supported this method.

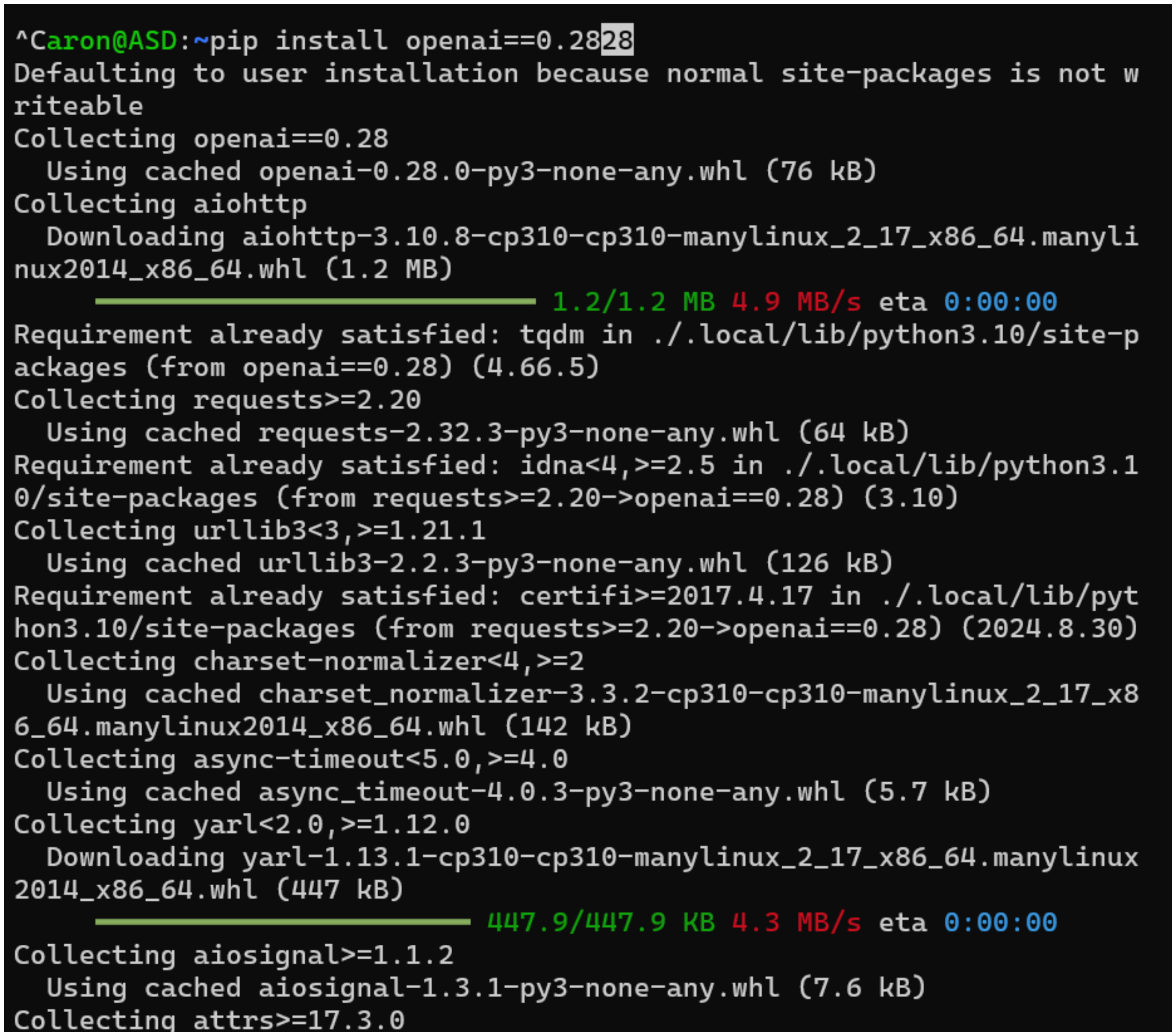
Solution 1: To fix the issue, I downgraded the OpenAI library to version `0.28.0`, which still supports the old API.

Command:

```
pip install openai==0.28.0
```

Steps Taken:

1. I uninstalled the latest OpenAI version and installed the older version.
2. After installing the correct version, I verified that the function worked correctly with the existing code.



```
^Caron@ASD:~pip install openai==0.2828
Defaulting to user installation because normal site-packages is not w
riteable
Collecting openai==0.28
  Using cached openai-0.28.0-py3-none-any.whl (76 kB)
Collecting aiohttp
  Downloading aiohttp-3.10.8-cp310-cp310-manylinux_2_17_x86_64.manyli
nux2014_x86_64.whl (1.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.2/1.2 MB 4.9 MB/s eta 0:00:00
Requirement already satisfied: tqdm in ./.local/lib/python3.10/site-p
ackages (from openai==0.28) (4.66.5)
Collecting requests>=2.20
  Using cached requests-2.32.3-py3-none-any.whl (64 kB)
Requirement already satisfied: idna<4,>=2.5 in ./.local/lib/python3.1
0/site-packages (from requests>=2.20->openai==0.28) (3.10)
Collecting urllib3<3,>=1.21.1
  Using cached urllib3-2.2.3-py3-none-any.whl (126 kB)
Requirement already satisfied: certifi>=2017.4.17 in ./.local/lib/pyt
hon3.10/site-packages (from requests>=2.20->openai==0.28) (2024.8.30)
Collecting charset-normalizer<4,>=2
  Using cached charset_normalizer-3.3.2-cp310-cp310-manylinux_2_17_x8
6_64.manylinux2014_x86_64.whl (142 kB)
Collecting async-timeout<5.0,>=4.0
  Using cached async_timeout-4.0.3-py3-none-any.whl (5.7 kB)
Collecting yarl<2.0,>=1.12.0
  Downloading yarl-1.13.1-cp310-cp310-manylinux_2_17_x86_64.manylinux
2014_x86_64.whl (447 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━ 447.9/447.9 KB 4.3 MB/s eta 0:00:00
Collecting aiosignal>=1.1.2
  Using cached aiosignal-1.3.1-py3-none-any.whl (7.6 kB)
Collecting attrs>=17.3.0
```

Solution 2 (Alternative): Alternatively, I could have updated the code to use the new `chat.Completion.create` method introduced in OpenAI version 1.0.0 or later.

Updated Code Example:

```
response = openai.ChatCompletion.create(
    model="gpt-3.5-turbo",
    messages=[{"role": "user", "content": user_question}]
)
```

---

3. Error: Incorrect API Key

Error Message:

```
openai.error.AuthenticationError: Incorrect API key provided
```

Cause: This error occurred because I either missed setting the OpenAI API key or used an incorrect one.

Solution: To fix this, I double-checked my API key from the OpenAI dashboard and updated the `openai.api_key` in my `app.py` file.

Steps Taken:

1. I logged in to my OpenAI account and retrieved the correct API key.
2. I updated the key in the Python code:

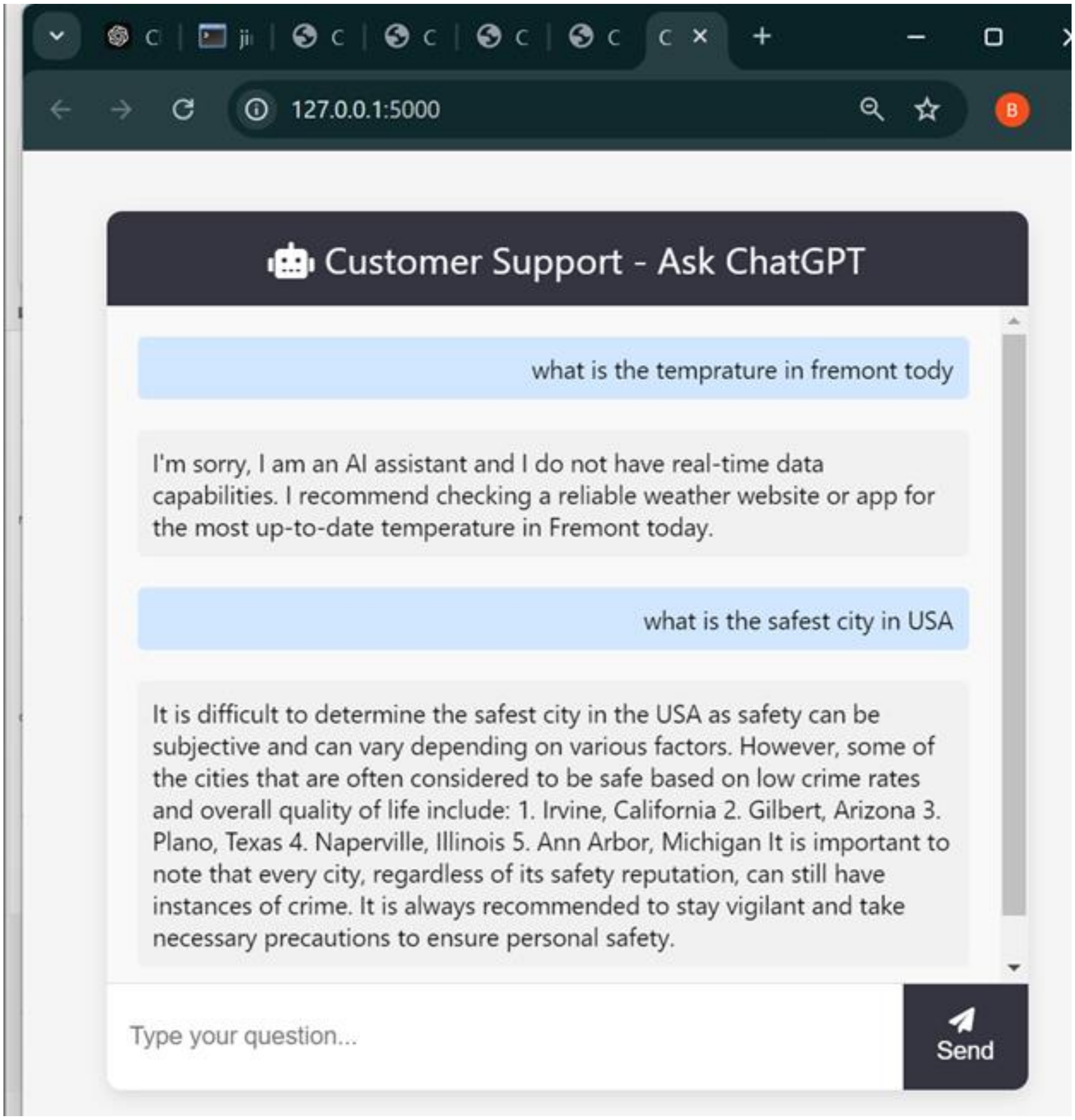
   ```
   openai.api_key = "your-correct-api-key"
   ```

---

7. Final Application Functionality

Step: Final demonstration

- Show the application running smoothly, with multiple questions being asked and responses generated in the chat interface.



Customer Support - Ask ChatGPT

what is the temprature in fremont tody

I'm sorry, I am an AI assistant and I do not have real-time data capabilities. I recommend checking a reliable weather website or app for the most up-to-date temperature in Fremont today.

what is the safest city in USA

It is difficult to determine the safest city in the USA as safety can be subjective and can vary depending on various factors. However, some of the cities that are often considered to be safe based on low crime rates and overall quality of life include: 1. Irvine, California 2. Gilbert, Arizona 3. Plano, Texas 4. Naperville, Illinois 5. Ann Arbor, Michigan It is important to note that every city, regardless of its safety reputation, can still have instances of crime. It is always recommended to stay vigilant and take necessary precautions to ensure personal safety.