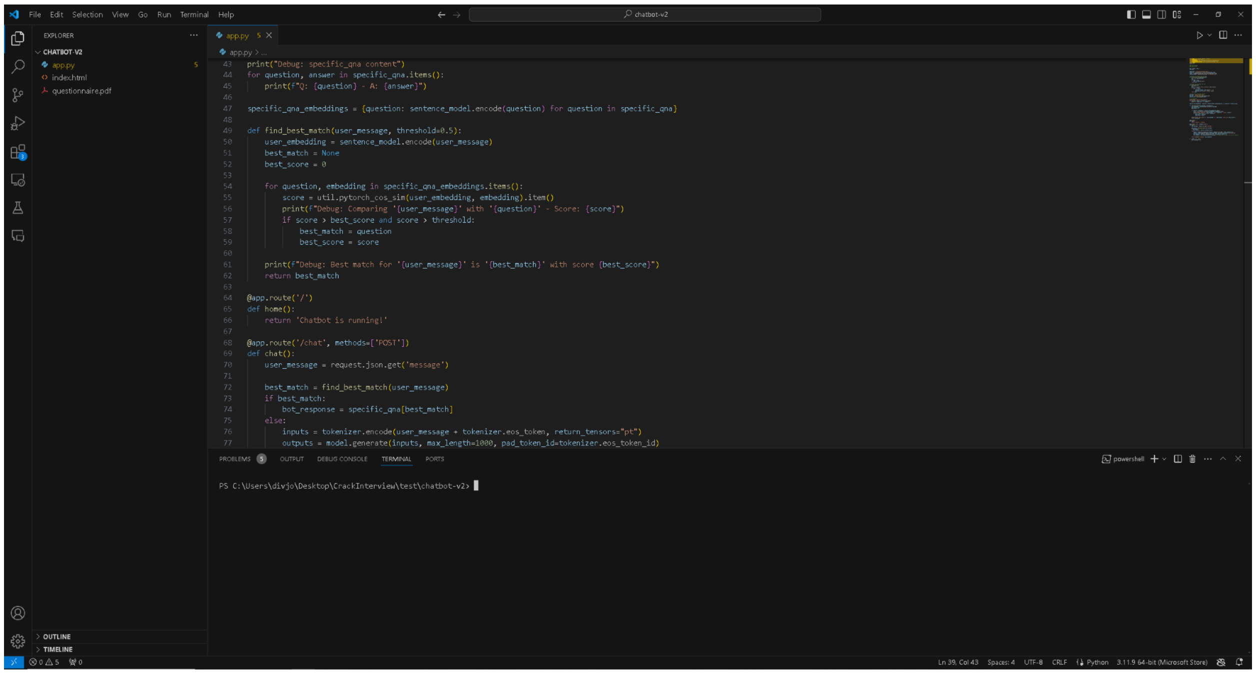
Chatbot Documentation

Initialization:

Step 1:

- Open code in Visual Studio Code



```
print("Debug: specific_qna content")
for question, answer in specific_qna.items():
    print(f"Q: {question} - A: {answer}")

specific_qna_embeddings = {question: sentence_model.encode(question) for question in specific_qna}

def find_best_match(user_message, threshold=0.5):
    user_embedding = sentence_model.encode(user_message)
    best_match = None
    best_score = 0

    for question, embedding in specific_qna_embeddings.items():
        score = util.pytorch_cos_sim(user_embedding, embedding).item()
        print(f"Debug: Comparing '{user_message}' with '{question}' - Score: {score}")
        if score > best_score and score > threshold:
            best_match = question
            best_score = score

    print(f"Debug: Best match for '{user_message}' is '{best_match}' with score {best_score}")
    return best_match

@app.route('/')
def home():
    return 'Chatbot is running!'

@app.route('/chat', methods=['POST'])
def chat():
    user_message = request.json.get('message')

    best_match = find_best_match(user_message)
    if best_match:
        bot_response = specific_qna[best_match]
    else:
        inputs = tokenizer.encode(user_message + tokenizer.eos_token, return_tensors="pt")
        outputs = model.generate(inputs, max_length=1000, pad_token_id=tokenizer.eos_token_id)
```

```
PS C:\Users\divjo\Desktop\CrackInterview\test\chatbot-v2>
```

-

Step 2:

- Open terminal in visual studio code from the top header. Terminal > New Terminal
- Ensure you have Flask and the necessary libraries installed by entering this command:

```
\CrackInterview\test\chatbot-v2> pip install Flask Flask-CORS transformers sentence-transformers pymupdf
```

pip install Flask Flask-CORS transformers sentence-transformers pymupdf tf-keras torch flask_socketio

Running:

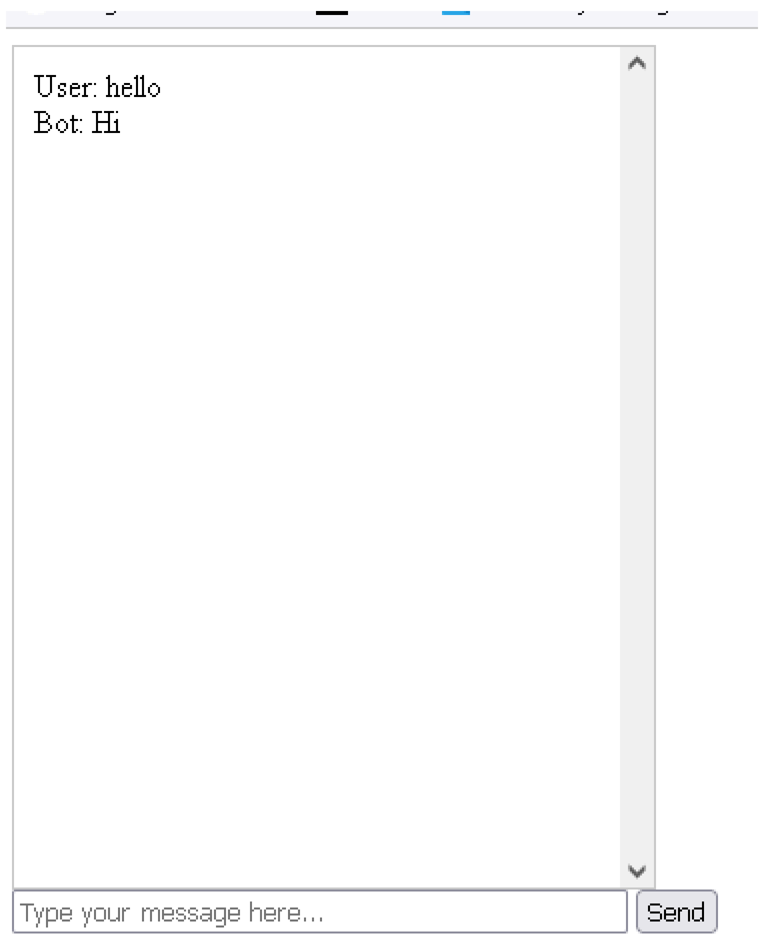
Step 1:

- Run the command:

```
\Desktop\CrackInterview\test\chatbot-v2> python app.py
```

Step 2:

- Open Index.html and enter any commands once the server finishes running on your local address (Have to click send button to enter message)

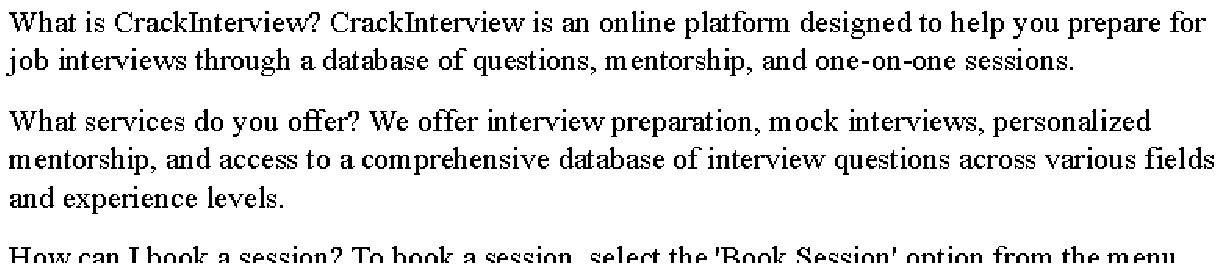

User: hello
Bot: Hi

How it works:

Questionnaire File:

The format of the file must be in:

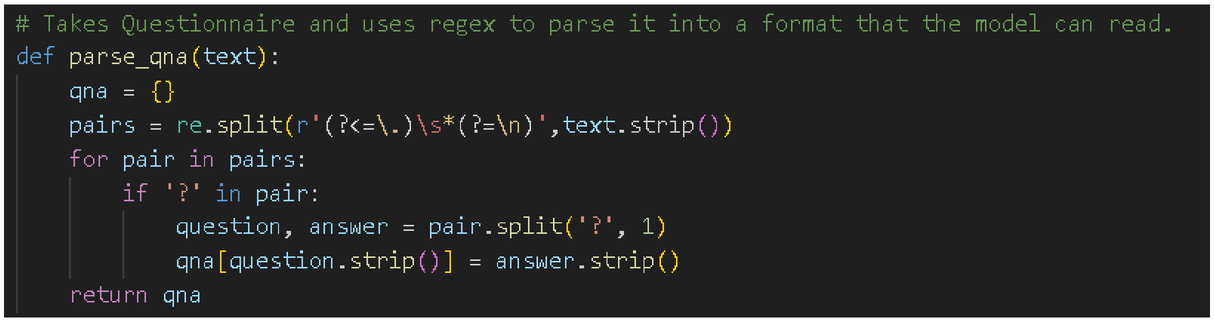
<questions?><space(optional)><answer><newline>
<...>


What is CrackInterview? CrackInterview is an online platform designed to help you prepare for job interviews through a database of questions, mentorship, and one-on-one sessions.

What services do you offer? We offer interview preparation, mock interviews, personalized mentorship, and access to a comprehensive database of interview questions across various fields and experience levels.

How can I book a session? To book a session, select the 'Book Session' option from the menu

This is so the model can easily read the file and train itself for our liking. To change the reading of the formatting, this can be edited using Regex at the function parse_qna:

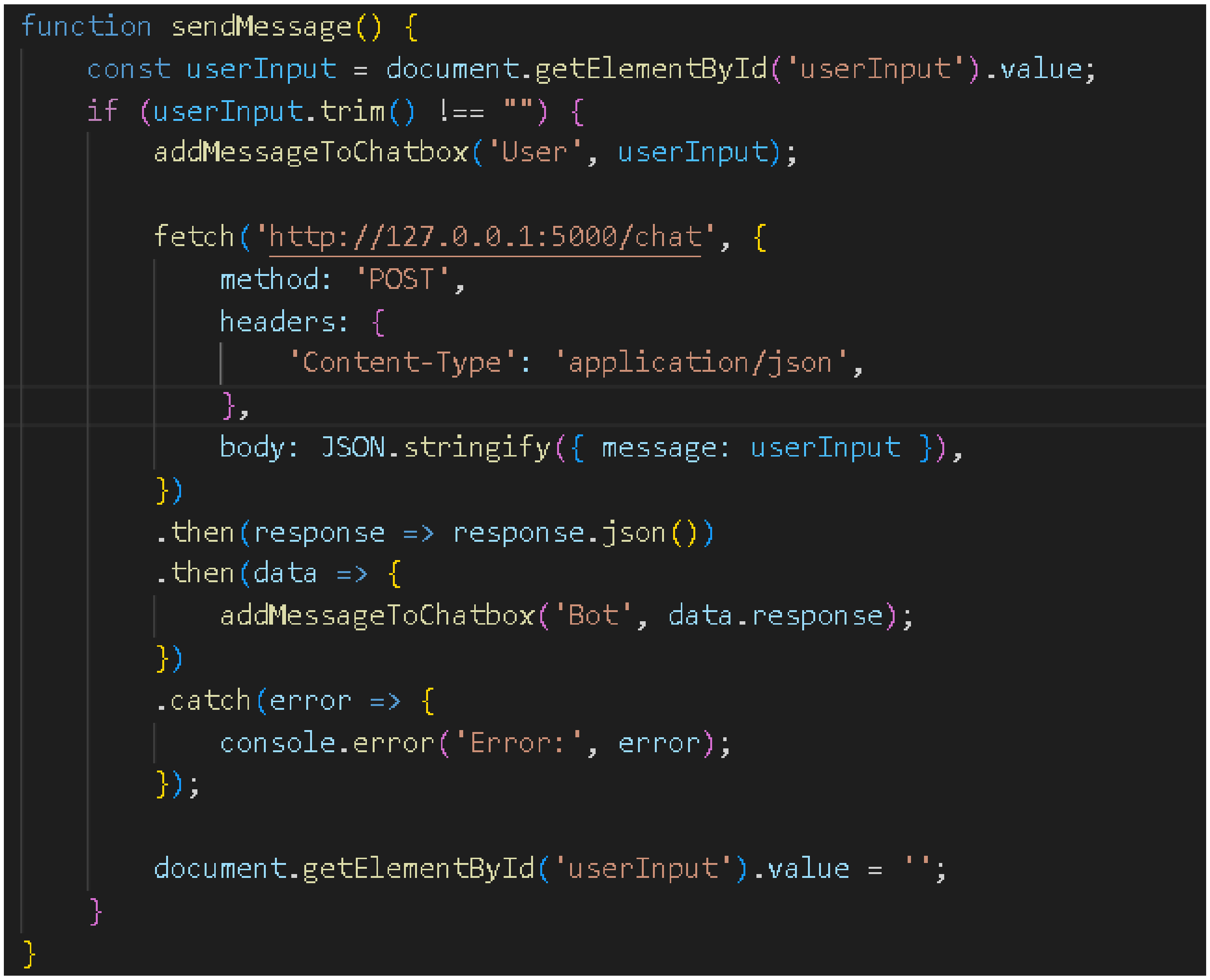


```python
# Takes Questionnaire and uses regex to parse it into a format that the model can read.
def parse_qna(text):
    qna = {}
    pairs = re.split(r'(?<=\.)\s*(?=\n)',text.strip())
    for pair in pairs:
        if '?' in pair:
            question, answer = pair.split('?', 1)
            qna[question.strip()] = answer.strip()
    return qna
```

<u>API Call:</u>

/chat

API calls from the front end can be made using a POST request.



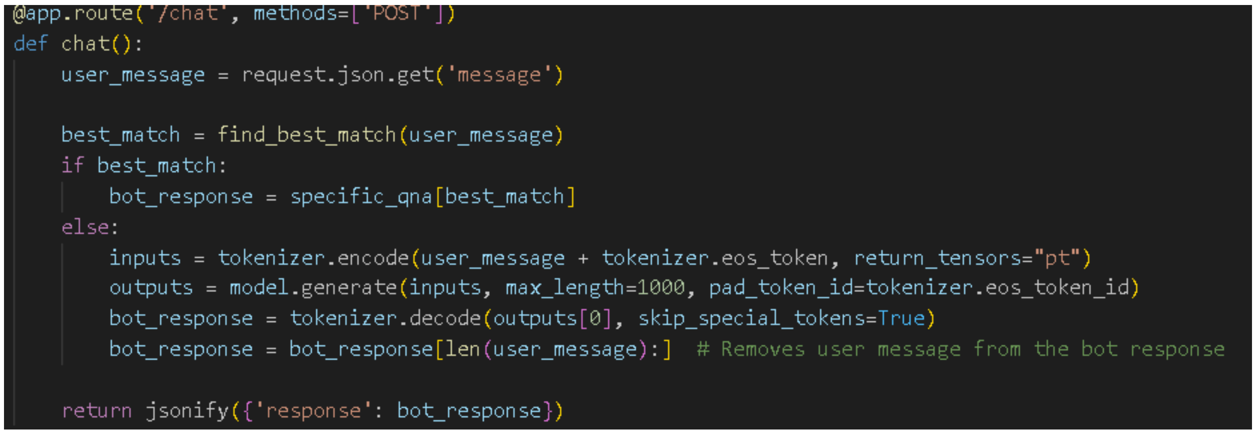
```javascript
function sendMessage() {
    const userInput = document.getElementById('userInput').value;
    if (userInput.trim() !== "") {
        addMessageToChatbox('User', userInput);

        fetch('http://127.0.0.1:5000/chat', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify({ message: userInput }),
        })
        .then(response => response.json())
        .then(data => {
            addMessageToChatbox('Bot', data.response);
        })
        .catch(error => {
            console.error('Error:', error);
        });

        document.getElementById('userInput').value = '';
    }
}
```

Here I just asked ChatGPT to generate a general JS frontend. Since we're going to be using React, this will look different. Since we do not have a defined URL, you will need to change the IP Address into yours. The post request can be accessed by <server url>/chat and sending a JSON file.

The request is handled in the backend here:



```python
@app.route('/chat', methods=['POST'])
def chat():
    user_message = request.json.get('message')

    best_match = find_best_match(user_message)
    if best_match:
        bot_response = specific_qna[best_match]
    else:
        inputs = tokenizer.encode(user_message + tokenizer.eos_token, return_tensors="pt")
        outputs = model.generate(inputs, max_length=1000, pad_token_id=tokenizer.eos_token_id)
        bot_response = tokenizer.decode(outputs[0], skip_special_tokens=True)
        bot_response = bot_response[len(user_message):]  # Removes user message from the bot response

    return jsonify({'response': bot_response})
```