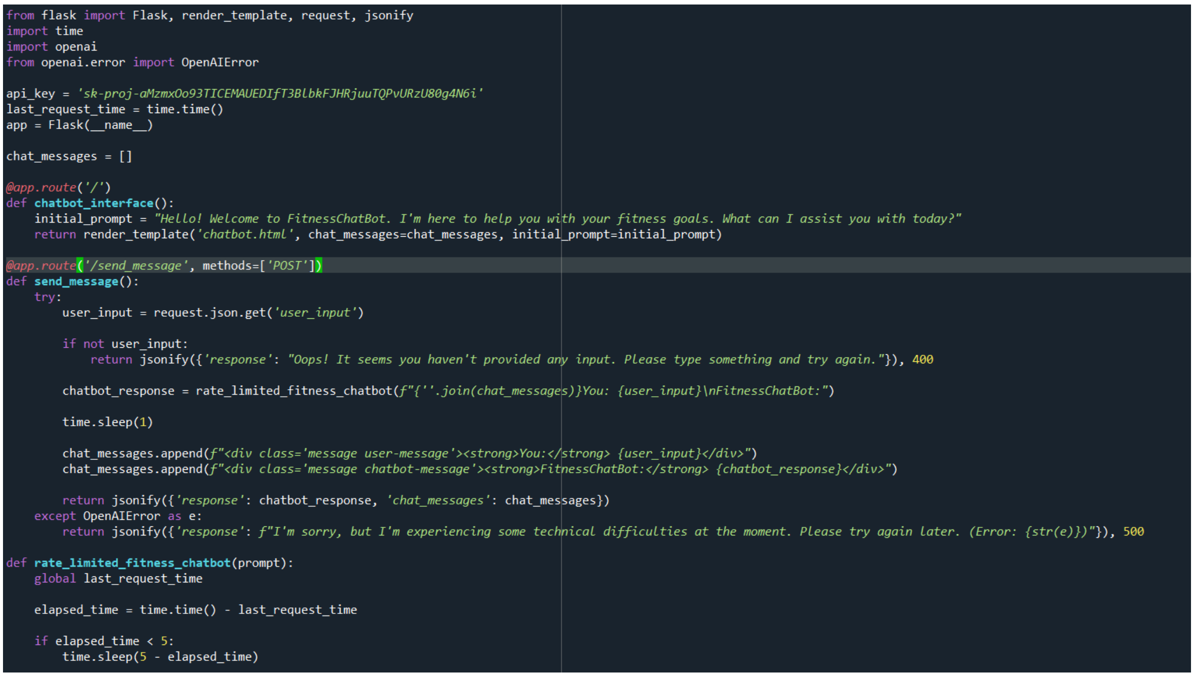
1.0 Purpose and Domain

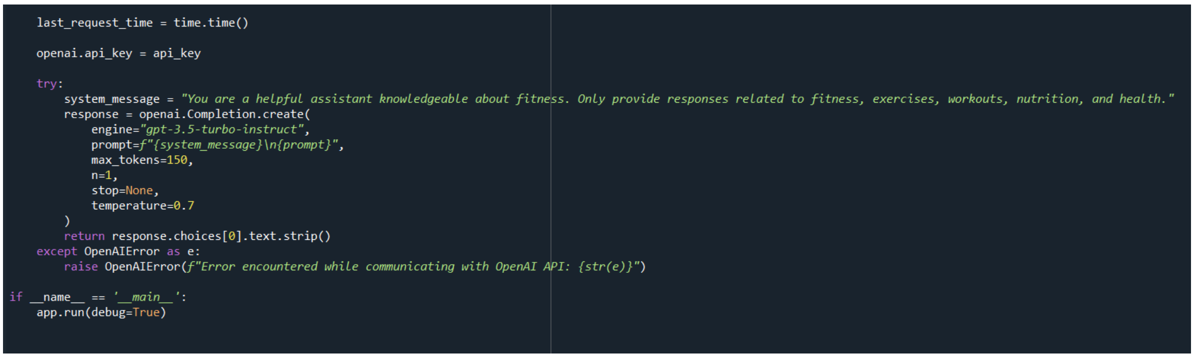
FitnessChatBot offers individualized exercise plans, dietary guidance, and health advice to help people reach their fitness objectives. FitnessChatBot's target market consists of gym-goers, fitness lovers, those trying to get healthier, and everybody who wants to lead an active lifestyle.

2.0 Technical Aspects

- Programming Languages: Python for backend development, HTML/CSS for frontend interface.
- Libraries: Flask for the web application framework, OpenAI for natural language processing.
- Unique Features: Speech recognition input, personalized fitness recommendations based on user input, rate limiting for API requests to prevent abuse.



```python
from flask import Flask, render_template, request, jsonify
import time
import openai
from openai.error import OpenAIError

api_key = 'sk-proj-aMzmxOo93TICEMAUEDIfT3BlbkFJHRjuuTQPvURzU80g4N6i'
last_request_time = time.time()
app = Flask(__name__)

chat_messages = []

@app.route('/')
def chatbot_interface():
    initial_prompt = "Hello! Welcome to FitnessChatBot. I'm here to help you with your fitness goals. What can I assist you with today?"
    return render_template('chatbot.html', chat_messages=chat_messages, initial_prompt=initial_prompt)

@app.route('/send_message', methods=['POST'])
def send_message():
    try:
        user_input = request.json.get('user_input')

        if not user_input:
            return jsonify({'response': "Oops! It seems you haven't provided any input. Please type something and try again."}), 400

        chatbot_response = rate_limited_fitness_chatbot(f"{''.join(chat_messages)}You: {user_input}\nFitnessChatBot:")

        time.sleep(1)

        chat_messages.append(f"<div class='message user-message'><strong>You:</strong> {user_input}</div>")
        chat_messages.append(f"<div class='message chatbot-message'><strong>FitnessChatBot:</strong> {chatbot_response}</div>")

        return jsonify({'response': chatbot_response, 'chat_messages': chat_messages})
    except OpenAIError as e:
        return jsonify({'response': f"I'm sorry, but I'm experiencing some technical difficulties at the moment. Please try again later. (Error: {str(e)})"}), 500

def rate_limited_fitness_chatbot(prompt):
    global last_request_time

    elapsed_time = time.time() - last_request_time

    if elapsed_time < 5:
        time.sleep(5 - elapsed_time)
```

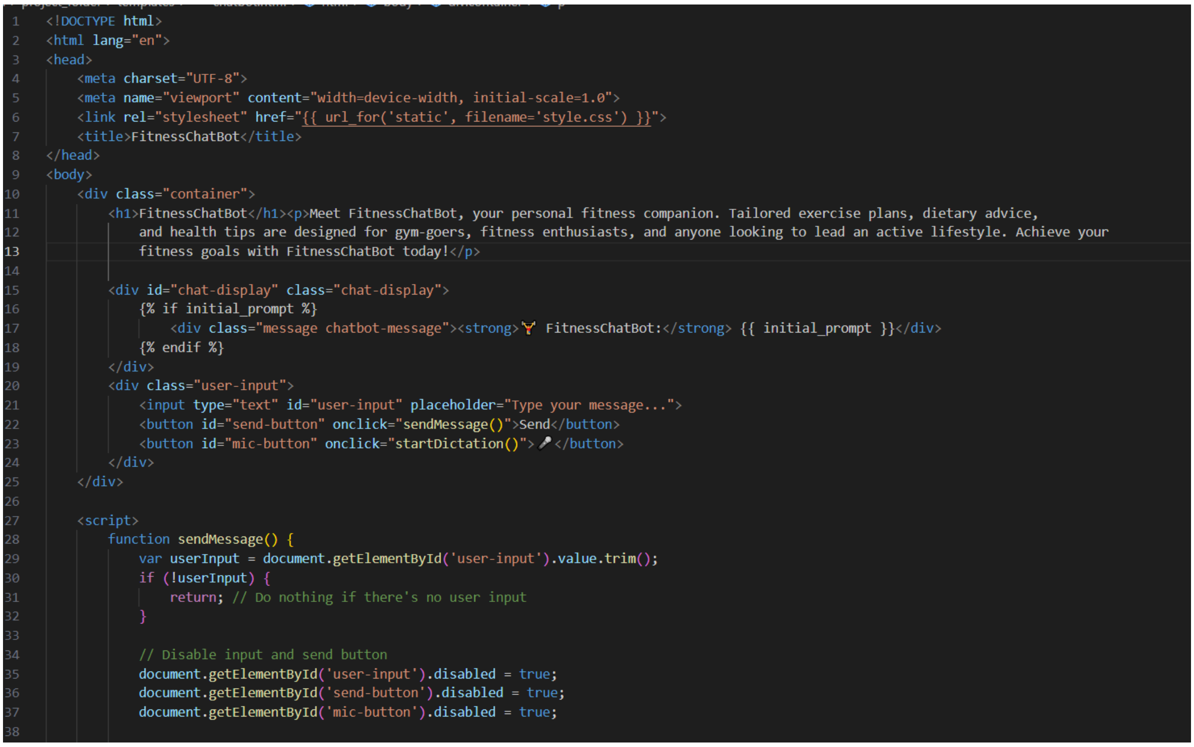


```python
    last_request_time = time.time()

    openai.api_key = api_key

    try:
        system_message = "You are a helpful assistant knowledgeable about fitness. Only provide responses related to fitness, exercises, workouts, nutrition, and health."
        response = openai.Completion.create(
            engine="gpt-3.5-turbo-instruct",
            prompt=f"{system_message}\n{prompt}",
            max_tokens=150,
            n=1,
            stop=None,
            temperature=0.7
        )
        return response.choices[0].text.strip()
    except OpenAIError as e:
        raise OpenAIError(f"Error encountered while communicating with OpenAI API: {str(e)}")

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

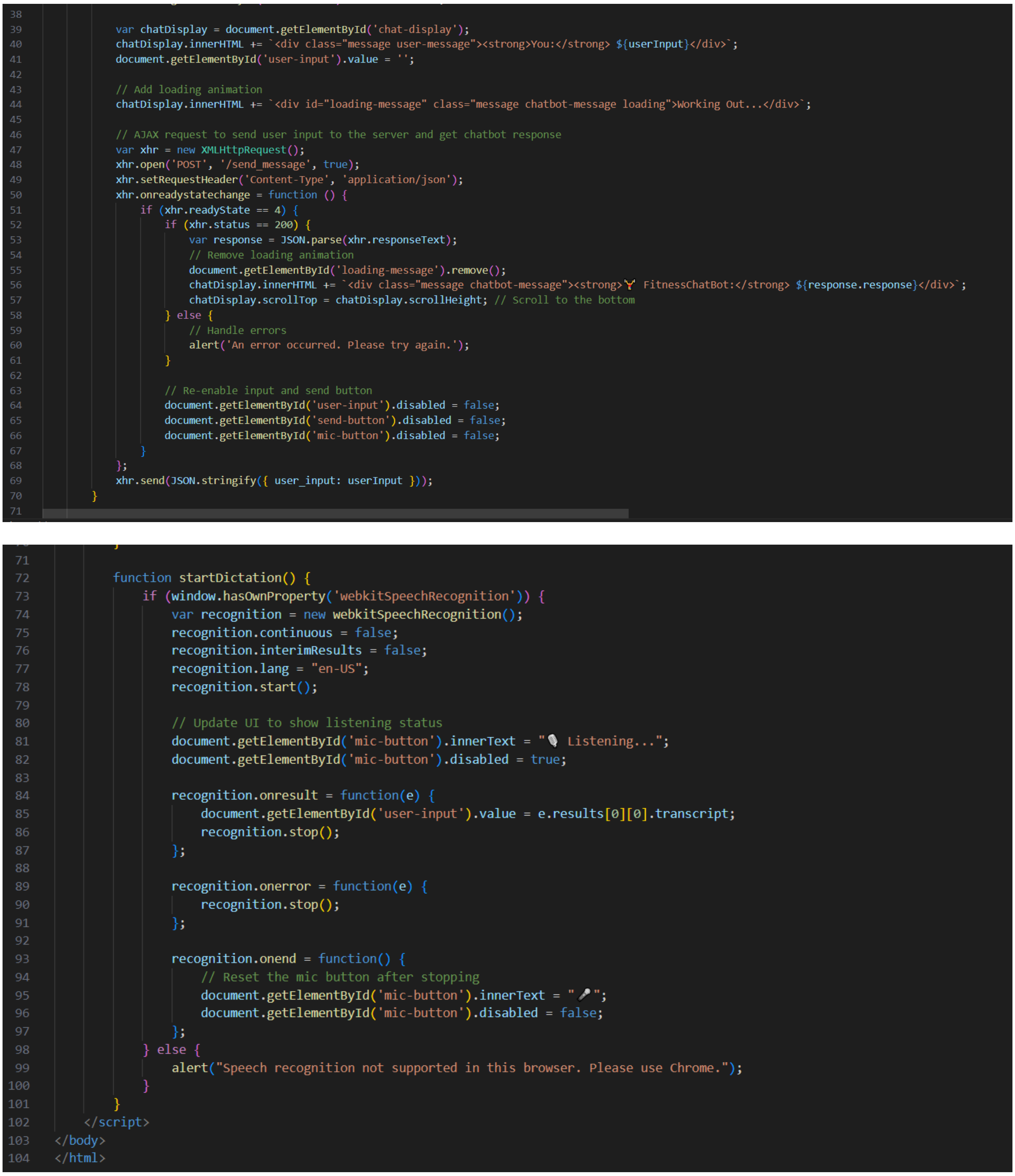
Figure 2.1 A screenshot of Python coding using Spyder

3.0 Web-based Interface

- Technologies Used: HTML for structure, CSS for styling.
- Interface Features: Text input box, send button, microphone button for speech recognition.



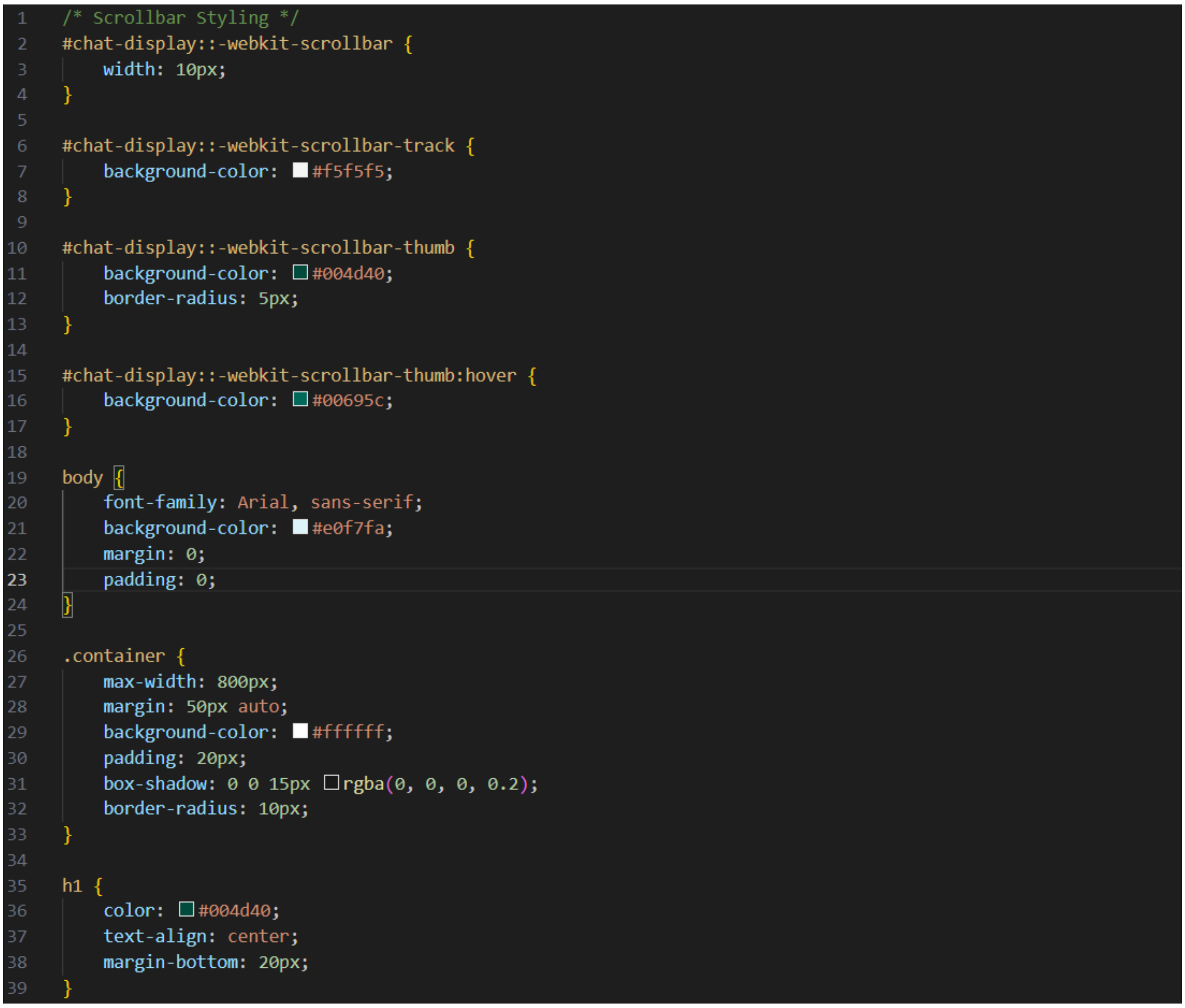
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
    <title>FitnessChatBot</title>
</head>
<body>
    <div class="container">
        <h1>FitnessChatBot</h1><p>Meet FitnessChatBot, your personal fitness companion. Tailored exercise plans, dietary advice,
            and health tips are designed for gym-goers, fitness enthusiasts, and anyone looking to lead an active lifestyle. Achieve your
            fitness goals with FitnessChatBot today!</p>

        <div id="chat-display" class="chat-display">
            {% if initial_prompt %}
                <div class="message chatbot-message"><strong>🏋 FitnessChatBot:</strong> {{ initial_prompt }}</div>
            {% endif %}
        </div>
        <div class="user-input">
            <input type="text" id="user-input" placeholder="Type your message...">
            <button id="send-button" onclick="sendMessage()">Send</button>
            <button id="mic-button" onclick="startDictation()">🎤</button>
        </div>
    </div>

    <script>
        function sendMessage() {
            var userInput = document.getElementById('user-input').value.trim();
            if (!userInput) {
                return; // Do nothing if there's no user input
            }

            // Disable input and send button
            document.getElementById('user-input').disabled = true;
            document.getElementById('send-button').disabled = true;
            document.getElementById('mic-button').disabled = true;

```



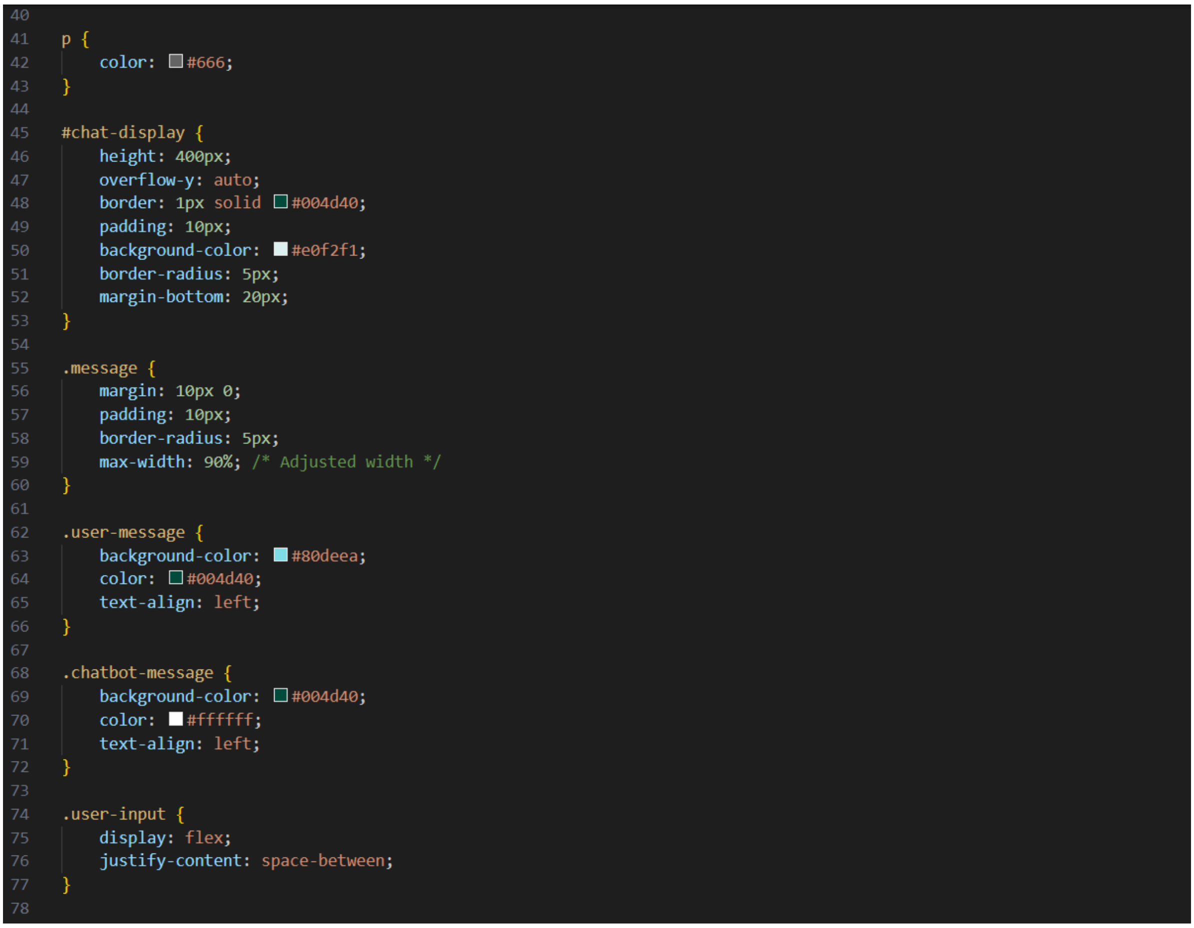
```javascript

            var chatDisplay = document.getElementById('chat-display');
            chatDisplay.innerHTML += `<div class="message user-message"><strong>You:</strong> ${userInput}</div>`;
            document.getElementById('user-input').value = '';

            // Add loading animation
            chatDisplay.innerHTML += `<div id="loading-message" class="message chatbot-message loading">Working Out...</div>`;

            // AJAX request to send user input to the server and get chatbot response
            var xhr = new XMLHttpRequest();
            xhr.open('POST', '/send_message', true);
            xhr.setRequestHeader('Content-Type', 'application/json');
            xhr.onreadystatechange = function () {
                if (xhr.readyState == 4) {
                    if (xhr.status == 200) {
                        var response = JSON.parse(xhr.responseText);
                        // Remove loading animation
                        document.getElementById('loading-message').remove();
                        chatDisplay.innerHTML += `<div class="message chatbot-message"><strong>🏋️ FitnessChatBot:</strong> ${response.response}</div>`;
                        chatDisplay.scrollTop = chatDisplay.scrollHeight; // Scroll to the bottom
                    } else {
                        // Handle errors
                        alert('An error occurred. Please try again.');
                    }

                    // Re-enable input and send button
                    document.getElementById('user-input').disabled = false;
                    document.getElementById('send-button').disabled = false;
                    document.getElementById('mic-button').disabled = false;
                }
            };
            xhr.send(JSON.stringify({ user_input: userInput }));
        }

```

```html

        function startDictation() {
            if (window.hasOwnProperty('webkitSpeechRecognition')) {
                var recognition = new webkitSpeechRecognition();
                recognition.continuous = false;
                recognition.interimResults = false;
                recognition.lang = "en-US";
                recognition.start();

                // Update UI to show listening status
                document.getElementById('mic-button').innerText = "🎙️ Listening...";
                document.getElementById('mic-button').disabled = true;

                recognition.onresult = function(e) {
                    document.getElementById('user-input').value = e.results[0][0].transcript;
                    recognition.stop();
                };

                recognition.onerror = function(e) {
                    recognition.stop();
                };

                recognition.onend = function() {
                    // Reset the mic button after stopping
                    document.getElementById('mic-button').innerText = "🎤";
                    document.getElementById('mic-button').disabled = false;
                };
            } else {
                alert("Speech recognition not supported in this browser. Please use Chrome.");
            }
        }
    </script>
</body>
</html>
```

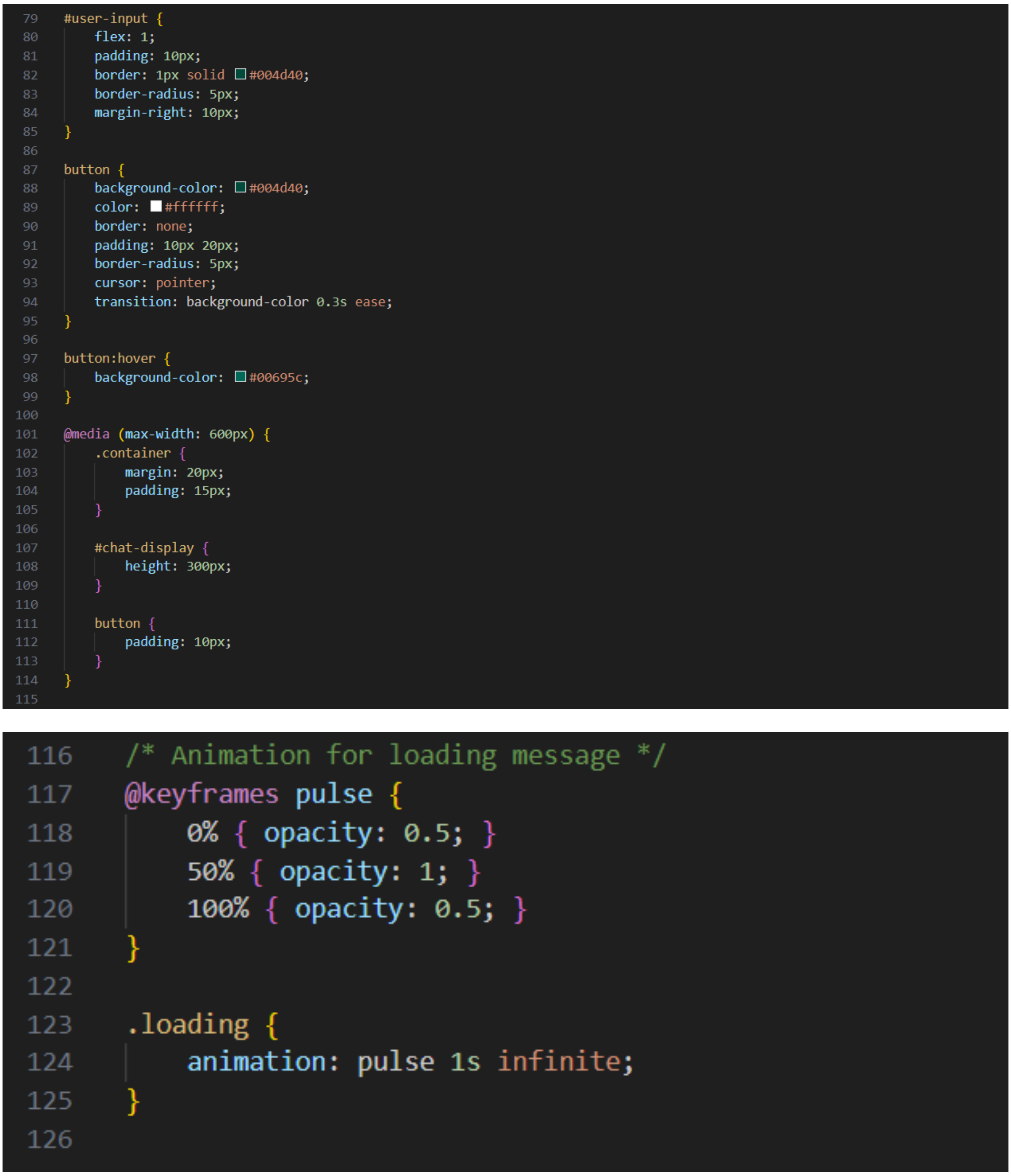
Figure 3.1 Screenshot of HTML coding using Visual Studio Code



```css
/* Scrollbar Styling */
#chat-display::-webkit-scrollbar {
    width: 10px;
}

#chat-display::-webkit-scrollbar-track {
    background-color: #f5f5f5;
}

#chat-display::-webkit-scrollbar-thumb {
    background-color: #004d40;
    border-radius: 5px;
}

#chat-display::-webkit-scrollbar-thumb:hover {
    background-color: #00695c;
}

body {
    font-family: Arial, sans-serif;
    background-color: #e0f7fa;
    margin: 0;
    padding: 0;
}

.container {
    max-width: 800px;
    margin: 50px auto;
    background-color: #ffffff;
    padding: 20px;
    box-shadow: 0 0 15px rgba(0, 0, 0, 0.2);
    border-radius: 10px;
}

h1 {
    color: #004d40;
    text-align: center;
    margin-bottom: 20px;
}
```



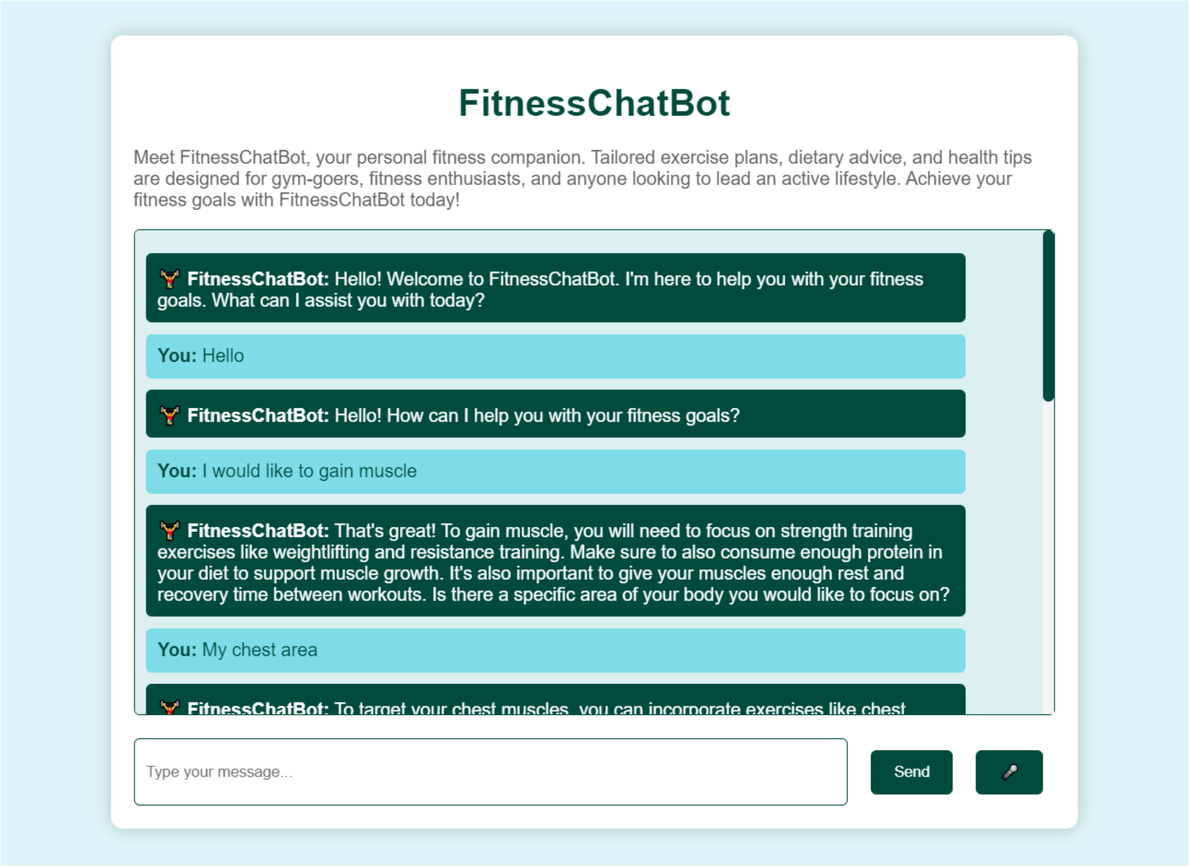
```css

p {
    color: #666;
}

#chat-display {
    height: 400px;
    overflow-y: auto;
    border: 1px solid #004d40;
    padding: 10px;
    background-color: #e0f2f1;
    border-radius: 5px;
    margin-bottom: 20px;
}

.message {
    margin: 10px 0;
    padding: 10px;
    border-radius: 5px;
    max-width: 90%; /* Adjusted width */
}

.user-message {
    background-color: #80deea;
    color: #004d40;
    text-align: left;
}

.chatbot-message {
    background-color: #004d40;
    color: #ffffff;
    text-align: left;
}

.user-input {
    display: flex;
    justify-content: space-between;
}

```



```css
#user-input {
    flex: 1;
    padding: 10px;
    border: 1px solid #004d40;
    border-radius: 5px;
    margin-right: 10px;
}

button {
    background-color: #004d40;
    color: #ffffff;
    border: none;
    padding: 10px 20px;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s ease;
}

button:hover {
    background-color: #00695c;
}

@media (max-width: 600px) {
    .container {
        margin: 20px;
        padding: 15px;
    }

    #chat-display {
        height: 300px;
    }

    button {
        padding: 10px;
    }
}

/* Animation for loading message */
@keyframes pulse {
    0% { opacity: 0.5; }
    50% { opacity: 1; }
    100% { opacity: 0.5; }
}

.loading {
    animation: pulse 1s infinite;
}
```

Figure 3.2 Screenshot of CSS coding using Visual Studio Code

4.0 Chatbot Interactions



FitnessChatBot

Meet FitnessChatBot, your personal fitness companion. Tailored exercise plans, dietary advice, and health tips are designed for gym-goers, fitness enthusiasts, and anyone looking to lead an active lifestyle. Achieve your fitness goals with FitnessChatBot today!

FitnessChatBot: Hello! Welcome to FitnessChatBot. I'm here to help you with your fitness goals. What can I assist you with today?

You: Hello

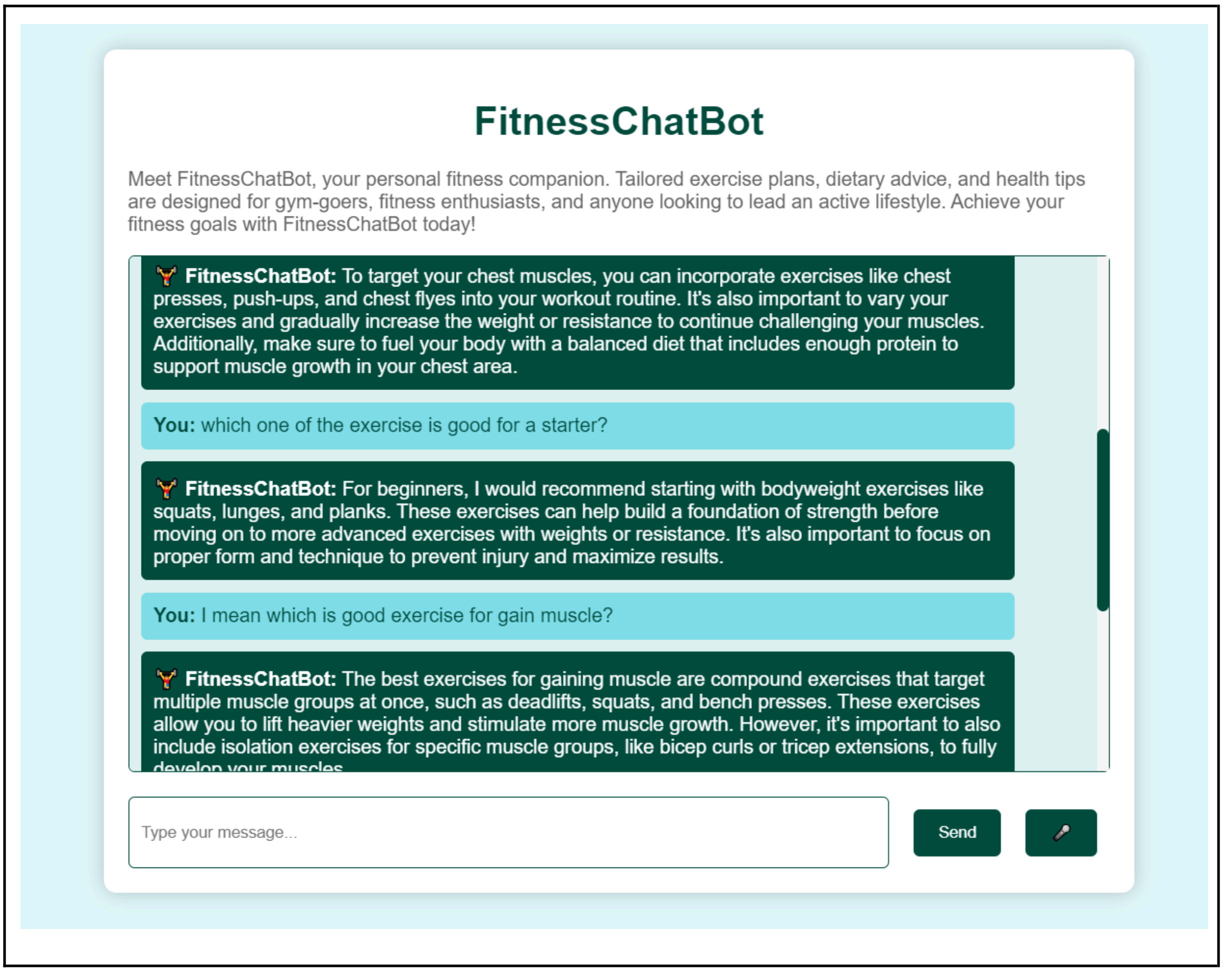
FitnessChatBot: Hello! How can I help you with your fitness goals?

You: I would like to gain muscle

FitnessChatBot: That's great! To gain muscle, you will need to focus on strength training exercises like weightlifting and resistance training. Make sure to also consume enough protein in your diet to support muscle growth. It's also important to give your muscles enough rest and recovery time between workouts. Is there a specific area of your body you would like to focus on?

You: My chest area

FitnessChatBot: To target your chest muscles, you can incorporate exercises like chest



FitnessChatBot

Meet FitnessChatBot, your personal fitness companion. Tailored exercise plans, dietary advice, and health tips are designed for gym-goers, fitness enthusiasts, and anyone looking to lead an active lifestyle. Achieve your fitness goals with FitnessChatBot today!

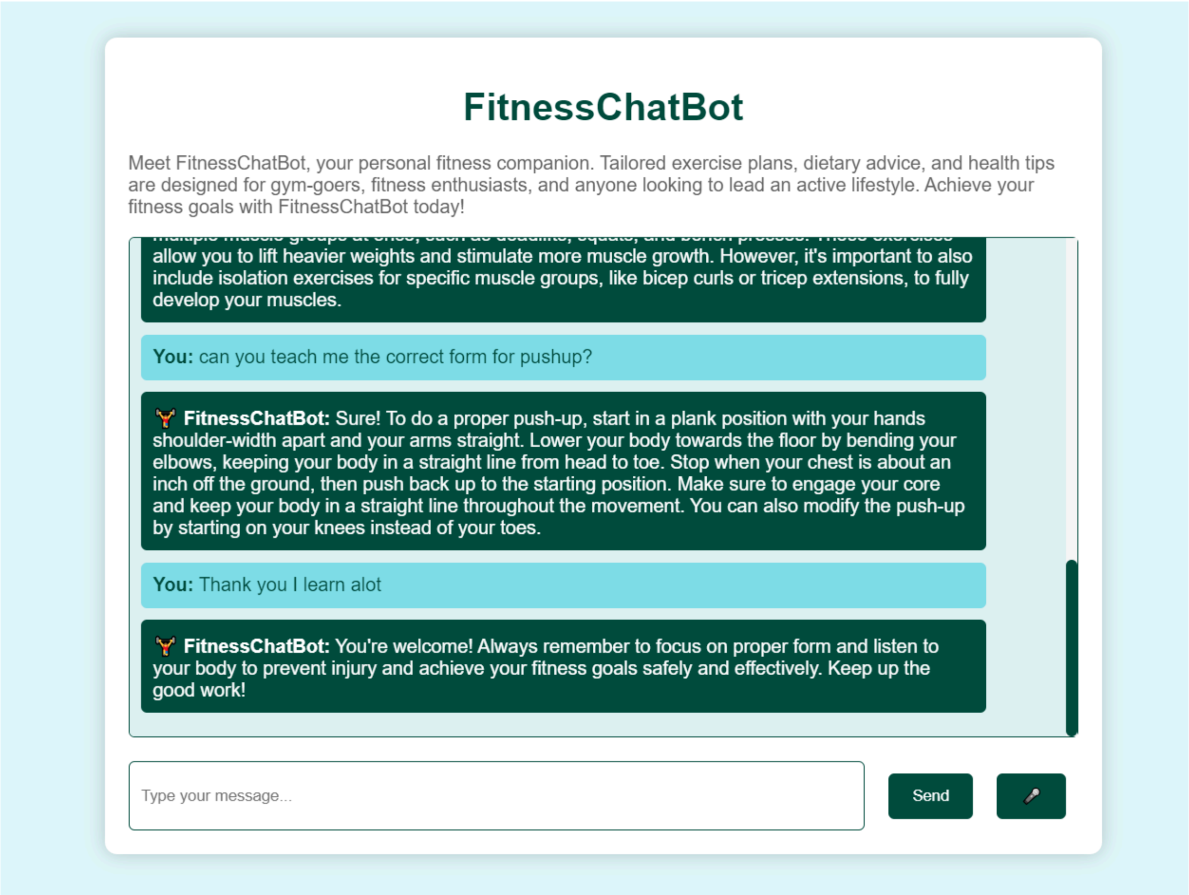
FitnessChatBot: To target your chest muscles, you can incorporate exercises like chest presses, push-ups, and chest flyes into your workout routine. It's also important to vary your exercises and gradually increase the weight or resistance to continue challenging your muscles. Additionally, make sure to fuel your body with a balanced diet that includes enough protein to support muscle growth in your chest area.

You: which one of the exercise is good for a starter?

FitnessChatBot: For beginners, I would recommend starting with bodyweight exercises like squats, lunges, and planks. These exercises can help build a foundation of strength before moving on to more advanced exercises with weights or resistance. It's also important to focus on proper form and technique to prevent injury and maximize results.

You: I mean which is good exercise for gain muscle?

FitnessChatBot: The best exercises for gaining muscle are compound exercises that target multiple muscle groups at once, such as deadlifts, squats, and bench presses. These exercises allow you to lift heavier weights and stimulate more muscle growth. However, it's important to also include isolation exercises for specific muscle groups, like bicep curls or tricep extensions, to fully develop your muscles



Meet FitnessChatBot, your personal fitness companion. Tailored exercise plans, dietary advice, and health tips are designed for gym-goers, fitness enthusiasts, and anyone looking to lead an active lifestyle. Achieve your fitness goals with FitnessChatBot today!

allow you to lift heavier weights and stimulate more muscle growth. However, it's important to also include isolation exercises for specific muscle groups, like bicep curls or tricep extensions, to fully develop your muscles.

You: can you teach me the correct form for pushup?

🏋 **FitnessChatBot:** Sure! To do a proper push-up, start in a plank position with your hands shoulder-width apart and your arms straight. Lower your body towards the floor by bending your elbows, keeping your body in a straight line from head to toe. Stop when your chest is about an inch off the ground, then push back up to the starting position. Make sure to engage your core and keep your body in a straight line throughout the movement. You can also modify the push-up by starting on your knees instead of your toes.

You: Thank you I learn alot

🏋 **FitnessChatBot:** You're welcome! Always remember to focus on proper form and listen to your body to prevent injury and achieve your fitness goals safely and effectively. Keep up the good work!

Figure 4.1 Screenshot of conversation with FitnessChatBot

5.0 Testing Process

- Test scenarios and cases: A variety of inputs include various fitness inquiries, accuracy tests for speech recognition, and stress tests for API response times.
- Test Results: FitnessChatBot handled a variety of user inputs with ease and responded appropriately. The accuracy of the speech recognition was satisfactory, and the API response times fell within reasonable bounds.