PATEL KASHYAP KALPESHKUMAR (Artificial Intelligence Intern)

Develop a chatbot equipped with sentiment analysis capabilities. The chatbot will analyze the sentiment of the user's input. The sentiment analysis component will determine whether the user's message expresses a positive, negative, or neutral sentiment. This project combines natural language processing (NLP) techniques, machine learning algorithms To Find the Sentiment Of User

Code:

Python (Backend)

```python
from flask import Flask, render_template, request, jsonify
import json
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import make_pipeline
import speech_recognition as sr
import pyttsx3

# Expanded training dataset
training_data = [
    ("I love this product", "positive"),
    ("This is the best purchase I've ever made", "positive"),
    ("I'm so happy with this", "positive"),
    ("The product quality is excellent", "positive"),
    ("The service was quick and good", "positive"),
    ("I am very happy with the service", "positive"),
    ("Just now I got the rewards for my work", "positive"),
    ("The delivery was quick and efficient", "positive"),
    ("This product exceeded my expectations", "positive"),
    ("I hate this product", "negative"),
    ("This is the worst purchase I've ever made", "negative"),
    ("I'm so disappointed with this", "negative"),
    ("The product quality is terrible", "negative"),
    ("The service was slow and bad", "negative"),
    ("I am very unhappy with the service", "negative"),
    ("Just now I faced issues with my order", "negative"),
    ("The delivery was delayed", "negative"),
    ("This product did not meet my expectations", "negative"),
    ("It's okay, nothing special", "neutral"),
```

```
    ("Not bad, could be better", "neutral"),
    ("It's an average product", "neutral"),
    ("The delivery was quick, but the product is not great", "neutral"),
    ("I need help with this", "neutral"),
    ("How can I add this to my resume?", "neutral"),
    ("I have mixed feelings about this product", "neutral"),
    ("The service was neither good nor bad", "neutral"),
    ("The product is neither good nor bad", "neutral"),
    ("I don't feel strongly about this", "neutral"),
    # New training data related to education
    ("The lecture was very informative", "positive"),
    ("I enjoyed the online course", "positive"),
    ("I don't understand this topic", "neutral"),
    ("The exam was very difficult", "negative"),
    ("The professor is very helpful", "positive"),
    # New training data related to news and current affairs
    ("The recent policy changes are beneficial", "positive"),
    ("I am concerned about the economic downturn", "negative"),
    ("The news coverage was biased", "negative"),
    ("I am indifferent to the election results", "neutral"),
    ("The new technology has great potential", "positive"),
    # New training data related to technology
    ("This software is very user-friendly", "positive"),
    ("I am experiencing issues with this app", "negative"),
    ("The update improved performance significantly", "positive"),
    ("I am not impressed with the new features", "negative"),
    ("The interface is clean and intuitive", "positive"),
    # New training data related to general knowledge
    ("I learned something new today", "positive"),
    ("I find this topic boring", "negative"),
    ("The book was very enlightening", "positive"),
    ("I don't have any opinion on this", "neutral"),
    ("The documentary was well-made", "positive")
]


# Preprocess and vectorize the text data
def preprocess_data(training_data):
    texts, labels = zip(*training_data)
    return texts, labels

texts, labels = preprocess_data(training_data)

# Create a model pipeline
model = make_pipeline(TfidfVectorizer(), MultinomialNB())
```

```
# Train the model
model.fit(texts, labels)

# Function to predict sentiment
def predict_sentiment(text):
    return model.predict([text])[0]

app = Flask(__name__)

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

@app.route('/get_response', methods=['POST'])
def get_response():
    user_input = request.json.get("message")
    sentiment = predict_sentiment(user_input)
    response = {
        'message': user_input,
        'sentiment': sentiment
    }
    return jsonify(response)

# Voice assistant functions
recognizer = sr.Recognizer()

@app.route('/voice_input', methods=['GET'])
def voice_input():
    with sr.Microphone() as source:
        recognizer.adjust_for_ambient_noise(source)
        audio = recognizer.listen(source)
        try:
            command = recognizer.recognize_google(audio)
            sentiment = predict_sentiment(command)
            response = {
                'message': command,
                'sentiment': sentiment
            }
            return jsonify(response)
        except sr.UnknownValueError:
            return jsonify({"error": "Sorry, I did not understand that."})
        except sr.RequestError:
            return jsonify({"error": "Could not request results from the speech
recognition service."})
```

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

HTML+CSS(Frontend):

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>KARY Chatbot</title>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.3/css/all.min.css">
    <style>
        :root {
            --primary-color: #4CAF50;
            --secondary-color: #007BFF;
            --background-color: #f5f5f5;
            --text-color: #333;
        }

        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            background-color: var(--background-color);
            color: var(--text-color);
            margin: 0;
            padding: 0;
            display: flex;
            justify-content: center;
            align-items: center;
            min-height: 100vh;
            transition: background-color 0.3s, color 0.3s;
        }

        .container {
            max-width: 800px;
            width: 95%;
            background-color: #fff;
            border-radius: 15px;
            box-shadow: 0 4px 20px rgba(0, 0, 0, 0.1);
            overflow: hidden;
            display: flex;
            flex-direction: column;
            height: 90vh;
        }
```

```css
.header {
    background-color: var(--primary-color);
    padding: 20px;
    text-align: center;
    color: white;
    position: relative;
}

.header h1 {
    margin: 0;
    font-size: 24px;
}

.dark-mode-toggle {
    position: absolute;
    top: 10px;
    right: 10px;
    background: none;
    border: none;
    color: white;
    font-size: 20px;
    cursor: pointer;
}

.chatbox {
    flex: 1;
    padding: 20px;
    overflow-y: auto;
    display: flex;
    flex-direction: column;
    gap: 15px;
}

.message {
    display: flex;
    align-items: flex-start;
    max-width: 80%;
}

.message .avatar {
    width: 40px;
    height: 40px;
    border-radius: 50%;
    margin-right: 10px;
```

```
        background-color: #ddd;
        display: flex;
        align-items: center;
        justify-content: center;
        font-weight: bold;
    }

    .message .content {
        padding: 12px;
        border-radius: 18px;
        position: relative;
    }

    .message .timestamp {
        font-size: 0.7em;
        color: #888;
        margin-top: 5px;
    }

    .user {
        align-self: flex-end;
    }

    .user .content {
        background-color: var(--secondary-color);
        color: white;
    }

    .bot .content {
        background-color: #F1F1F1;
        color: black;
    }

    .typing-indicator {
        display: flex;
        padding: 10px;
        background-color: #F1F1F1;
        border-radius: 18px;
        width: fit-content;
    }

    .typing-indicator span {
        height: 10px;
        width: 10px;
        background-color: #666;
```

```
        border-radius: 50%;
        display: inline-block;
        margin: 0 2px;
        animation: typing 1s infinite;
    }

    @keyframes typing {
        0% { transform: translateY(0); }
        50% { transform: translateY(-5px); }
        100% { transform: translateY(0); }
    }

    .input-box {
        display: flex;
        border-top: 1px solid #ccc;
        padding: 10px;
    }

    .input-box textarea {
        flex: 1;
        padding: 12px;
        border: 1px solid #ccc;
        border-radius: 20px;
        outline: none;
        resize: none;
        font-family: inherit;
    }

    .input-box button {
        padding: 12px 20px;
        background-color: var(--primary-color);
        border: none;
        color: white;
        cursor: pointer;
        border-radius: 20px;
        margin-left: 10px;
        transition: background-color 0.3s;
    }

    .input-box button:hover {
        background-color: #45a049;
    }

    .input-box .microphone-button {
        background-color: #FF5722;
```

```css
}

.input-box .microphone-button.recording {
    animation: recordingAnimation 1s infinite;
}

@keyframes recordingAnimation {
    0% { background-color: #FF5722; }
    50% { background-color: #FF8A50; }
    100% { background-color: #FF5722; }
}

.suggestions-container {
    display: flex;
    flex-wrap: wrap;
    gap: 10px;
    padding: 15px;
    border-top: 1px solid #ccc;
}

.suggestion {
    background-color: #f0f0f0;
    padding: 10px 15px;
    border-radius: 20px;
    cursor: pointer;
    transition: background-color 0.3s, color 0.3s;
}

.suggestion:hover {
    background-color: var(--primary-color);
    color: white;
}

.suggestion.selected {
    background-color: var(--primary-color);
    color: white;
}

.suggestion i {
    margin-right: 5px;
}

/* Dark mode styles */
body.dark-mode {
    --background-color: #222;
```

```
            --text-color: #f5f5f5;
        }

        body.dark-mode .container {
            background-color: #333;
        }

        body.dark-mode .bot .content {
            background-color: #444;
            color: #f5f5f5;
        }

        body.dark-mode .input-box textarea {
            background-color: #444;
            color: #f5f5f5;
            border-color: #555;
        }

        body.dark-mode .suggestion {
            background-color: #444;
            color: #f5f5f5;
        }

        @media (max-width: 600px) {
            .container {
                width: 100%;
                height: 100vh;
                border-radius: 0;
            }

            .header h1 {
                font-size: 20px;
            }

            .message {
                max-width: 90%;
            }
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="header">
            <h1>KARY Chatbot</h1>
            <p>Hello Kashyap,</p>
```

```
            <p>How can I assist you today?</p>
            <button class="dark-mode-toggle" onclick="toggleDarkMode()">
                <i class="fas fa-moon"></i>
            </button>
        </div>
        <div class="chatbox" id="chatbox"></div>
        <div class="input-box">
            <textarea id="user-input" placeholder="Enter a prompt here" rows="1"
oninput="autoResize(this)"></textarea>
            <button onclick="sendMessage()"><i class="fas fa-paper-
plane"></i></button>
            <button id="microphone-button" class="microphone-button"
onclick="sendVoiceMessage()"><i class="fas fa-microphone"></i></button>
        </div>
        <div class="suggestions-container" id="suggestions-container"></div>
    </div>

    <script>
        let suggestions = [
            { text: "This software is very user-friendly", icon: "fas fa-pen" },
            { text: "I am not impressed with the new features", icon: "fas fa-
book" },
            { text: "I am indifferent to the election results", icon: "fas fa-
phone" },
            { text: "I am experiencing issues with this app", icon: "fas fa-user-
circle"},
            { text: "I find this topic boring", icon: "fas fa-laptop" }
        ];

        function displaySuggestions() {
            const suggestionsContainer = document.getElementById('suggestions-
container');
            suggestionsContainer.innerHTML = '';

            suggestions.forEach(suggestion => {
                const suggestionElement = document.createElement('div');
                suggestionElement.classList.add('suggestion');
                suggestionElement.innerHTML = `<i class="${suggestion.icon}"></i>
${suggestion.text}`;
                suggestionElement.addEventListener('click', () => {
                    sendMessage(suggestion.text);
                    const selectedSuggestion =
document.querySelector('.suggestion.selected');
                    if (selectedSuggestion) {
                        selectedSuggestion.classList.remove('selected');
```

```
                    }
                    suggestionElement.classList.add('selected');
                });
                suggestionsContainer.appendChild(suggestionElement);
            });
        }

        displaySuggestions();

        function toggleDarkMode() {
            document.body.classList.toggle('dark-mode');
            const darkModeToggle = document.querySelector('.dark-mode-toggle i');
            darkModeToggle.classList.toggle('fa-moon');
            darkModeToggle.classList.toggle('fa-sun');
        }

        function autoResize(textarea) {
            textarea.style.height = 'auto';
            textarea.style.height = textarea.scrollHeight + 'px';
        }

        function sendMessage(userInput = document.getElementById('user-
input').value.trim()) {
            if (userInput === '') return;

            appendMessage('user', userInput);
            document.getElementById('user-input').value = '';
            autoResize(document.getElementById('user-input'));

            // Show typing indicator
            showTypingIndicator();

            // Send message to the backend
            fetch('/get_response', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify({ message: userInput })
            })
            .then(response => response.json())
            .then(data => {
                removeTypingIndicator();
                appendMessage('bot', generateBotResponse(data));
            })
```

```
            .catch(error => {
                console.error('Error:', error);
                removeTypingIndicator();
                appendMessage('bot', 'Sorry, there was an error processing your
request.');
            });
        }

        function sendVoiceMessage() {
            const microphoneButton = document.getElementById('microphone-
button');
            microphoneButton.classList.add('recording');

            fetch('/voice_input', {
                method: 'GET'
            })
            .then(response => response.json())
            .then(data => {
                microphoneButton.classList.remove('recording');

                if (data.error) {
                    appendMessage('bot', data.error);
                    return;
                }

                appendMessage('user', data.message);
                appendMessage('bot', generateBotResponse(data));
            })
            .catch(error => {
                console.error('Error:', error);
                microphoneButton.classList.remove('recording');
                appendMessage('bot', 'Sorry, there was an error processing your
voice input.');
            });
        }

        function generateBotResponse(data) {
            if (data.sentiment === 'positive') {
                return "I'm glad to hear that! Your message seems positive.";
            } else if (data.sentiment === 'negative') {
                return "I'm sorry to hear that. Your message seems negative.";
            } else {
                return "Thank you for sharing that information. Your message
seems neutral.";
            }
```

```
        }

        function appendMessage(sender, content) {
            const chatbox = document.getElementById('chatbox');
            const messageDiv = document.createElement('div');
            messageDiv.classList.add('message', sender);

            const avatar = document.createElement('div');
            avatar.classList.add('avatar');
            avatar.textContent = sender === 'user' ? 'K' : 'B';

            const messageContent = document.createElement('div');
            messageContent.classList.add('content');
            messageContent.textContent = content;

            const timestamp = document.createElement('div');
            timestamp.classList.add('timestamp');
            timestamp.textContent = new Date().toLocaleTimeString([], { hour: '2-
digit', minute: '2-digit' });

            messageDiv.appendChild(avatar);
            messageDiv.appendChild(messageContent);
            messageContent.appendChild(timestamp);

            chatbox.appendChild(messageDiv);
            chatbox.scrollTop = chatbox.scrollHeight;
        }

        function showTypingIndicator() {
            const chatbox = document.getElementById('chatbox');
            const typingDiv = document.createElement('div');
            typingDiv.classList.add('message', 'bot', 'typing-indicator');
            typingDiv.innerHTML = '<span></span><span></span><span></span>';
            chatbox.appendChild(typingDiv);
            chatbox.scrollTop = chatbox.scrollHeight;
        }

        function removeTypingIndicator() {
            const typingIndicator = document.querySelector('.typing-indicator');
            if (typingIndicator) {
                typingIndicator.remove();
            }
        }
    </script>
</body>
```

```
</html>
```