```
pip install pandas scikit-learn nltk
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

def load_data(file_path):
    data = pd.read_csv('/content/DIABETES AND SYMPTOMS DATASET.csv')
    return data

def preprocess_data(data):
    symptom_columns = ['Symptom_1', 'Symptom_2', 'Symptom_3', 'Symptom_4', 'Symptom_5', 'Symptom_6', 'Symptom_7',
                       'Symptom_8', 'Symptom_9', 'Symptom_10', 'Symptom_11', 'Symptom_12', 'Symptom_13',
                       'Symptom_14', 'Symptom_15', 'Symptom_16', 'Symptom_17', 'Symptom_18', 'Symptom_19', 'Symptom_20']

    data['Symptoms'] = data[symptom_columns].apply(lambda row: ' '.join(row.dropna()), axis=1)
    vectorizer = TfidfVectorizer()
    X_tfidf = vectorizer.fit_transform(data['Symptoms'])
    y = data['Disease']
    return X_tfidf, y, vectorizer

def train_model(X, y):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
    model = RandomForestClassifier(n_estimators=100, random_state=42, class_weight='balanced')
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print("Accuracy:", accuracy_score(y_test, y_pred))
    return model
```

```
def preprocess_input(symptoms_input):
    symptoms_input_list = symptoms_input.lower().strip().split(",")
    symptoms_input_str = ' '.join(symptoms_input_list)
    return symptoms_input_str

def predict_disease(model, vectorizer, symptoms_input):
    symptoms_input_str = preprocess_input(symptoms_input)
    symptoms_input_tfidf = vectorizer.transform([symptoms_input_str])
    predicted_disease = model.predict(symptoms_input_tfidf)
    return predicted_disease[0]

file_path = '/content/DIABETES AND SYMPTOMS DATASET.csv'
data = load_data(file_path)
X, y, vectorizer = preprocess_data(data)
model = train_model(X, y)

def chat_response(user_input):
    user_input = user_input.lower()

    if user_input in ["hi", "hello"]:
        return "Hello! How can I assist you today?"

    elif user_input in ["i want to know what type of diabetes i have", "guess the type of diabetes", "guess the diabetes", "help me"]:
        return "Sure! Please tell me 3 or more symptoms you have (comma separated)."

    elif user_input in ["exit", "bye"]:
        return "Goodbye!"

    else:

        disease = predict_disease(model, vectorizer, user_input)
        return f"Based on your symptoms, you may have: {disease}"
```

```
from google.colab import output

# Register chatbot function for JavaScript to access
def chat_function(user_message):
    return chat_response(user_message)

output.register_callback('notebook.chat', chat_function)
```

```
from IPython.display import HTML, display

chatbot_html = '''
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Doc Bot Chat</title>
    <style>
        body { font-family: Arial, sans-serif; display: flex; justify-content: center; align-items: center; height: 100vh; margin: 0; background-color: #f3f4f6; color: #333; }
        #chat-container { width: 400px; background-color: #f9f9f9; border-radius: 10px; box-shadow: 0px 4px 8px rgba(0, 0, 0, 0.2); display: flex; flex-direction: column; }
        #chat-log { height: 400px; overflow-y: auto; padding: 10px; background-color: #ffffff; border-radius: 10px 10px 0 0; }
        .message { margin: 8px 0; padding: 10px 12px; border-radius: 8px; max-width: 80%; font-size: 14px; position: relative; }
        .user { background-color: #ADD8E6; color: #000; text-align: right; margin-left: auto; }
        .bot { background-color: #001f3f; color: white; text-align: left; margin-right: auto; }
        .bot-label { font-weight: bold; color: #001f3f; margin-bottom: 5px; }
        .user-label { font-weight: bold; color: #ADD8E6; margin-bottom: 5px; text-align: right; display: block; }
        #input-container { display: flex; padding: 10px; border-top: 1px solid #ddd; background-color: #f9f9f9; border-radius: 0 0 10px 10px; }
        #user-input { flex-grow: 1; padding: 10px; border: 1px solid #ddd; border-radius: 5px; background-color: #f3f3f3; color: #333; }
        #send-btn { padding: 10px; border: none; background-color: #007bff; color: white; border-radius: 5px; cursor: pointer; margin-left: 5px; }
        #loading-indicator { display: none; color: #555; text-align: center; }
    </style>
</head>
<body>
    <div id="chat-container">
        <div id="chat-log"></div>
        <div id="loading-indicator">Doc Bot is typing...</div>
        <div id="input-container">
            <input type="text" id="user-input" placeholder="Type a message..." />
            <button id="send-btn">Send</button>
        </div>
    </div>

    <script>
        const chatLog = document.getElementById('chat-log');
        const userInput = document.getElementById('user-input');
        const sendBtn = document.getElementById('send-btn');

        async function getChatbotResponse(userMessage) {
            document.getElementById("loading-indicator").style.display = "block";
            const response = await google.colab.kernel.invokeFunction('notebook.chat', [userMessage], {});
            document.getElementById("loading-indicator").style.display = "none";
            return response.data['text/plain'].replace(/['"]+/g, '');  // Remove quotations from response
        }

        function displayInitialBotMessages() {
            displayMessage("I am Doctor Bot or Doc Bot for short. I will answer which type of Diabetes you have depending on the symptoms you gave to me. If you want to end the chat, please type 'Bye'!", 'bot', true);
        }

        sendBtn.addEventListener('click', async () => {
            const userMessage = userInput.value.trim();
            if (!userMessage) return;

            displayMessage(userMessage, 'user');
            userInput.value = '';
```

```
            try {
                const botMessage = await getChatbotResponse(userMessage);
                displayMessage(botMessage, 'bot', true);
            } catch (error) {
                displayMessage("Error: Could not connect to Doc Bot.", 'bot', true);
            }
        });

        function displayMessage(message, sender, isBot = false) {
            const messageDiv = document.createElement('div');
            messageDiv.classList.add('message', sender);

            if (isBot && sender === 'bot') {
                const botLabel = document.createElement('div');
                botLabel.classList.add('bot-label');
                botLabel.textContent = "Doc Bot";
                chatLog.appendChild(botLabel);
            } else if (sender === 'user') {
                const userLabel = document.createElement('div');
                userLabel.classList.add('user-label');
                userLabel.textContent = "You";
                chatLog.appendChild(userLabel);
            }

            messageDiv.textContent = message;
            chatLog.appendChild(messageDiv);
            chatLog.scrollTop = chatLog.scrollHeight;
        }

        // Initial bot greeting message
        displayInitialBotMessages();
    </script>
</body>
</html>
'''

display(HTML(chatbot_html))
```