This script builds a graphical user interface (GUI) for interacting with the chatbot created. Let's go through each component and function in detail:

1. **Importing Libraries**:
   - The script imports necessary libraries like `nltk` , `pickle` , `numpy` , `tkinter` , and `keras` .
   - These libraries are used for natural language processing (NLP), loading the pre-trained chatbot model, handling GUI components, and performing other required tasks.
2. **Loading Pre-trained Model and Data**:
   - It loads the pre-trained chatbot model from the saved file using Keras's `load_model()` function.
   - Additionally, it loads the words, classes, and intents data from the pickle files saved during training.
3. **Cleaning and Processing User Input**:
   - The `clean_up_sentence()` function tokenizes and lemmatizes the user's input sentence. This prepares the input for the model to make predictions.
   - The `bag_of_words()` function converts the cleaned input sentence into a bag of words representation. This is similar to what was done during model training.
4. **Making Predictions**:
   - The `predict_class()` function predicts the intent of the user's input by passing the bag of words representation to the pre-trained model.
   - It sets a threshold (`ERROR_THRESHOLD`) to filter out predictions with low confidence scores.
   - The function returns a list of intents along with their probabilities.
5. **Generating Responses**:
   - The `getResponse()` function selects a response based on the predicted intent.
   - It randomly selects a response from the corresponding intent's response list in the intents data loaded from the JSON file.
   - If the predicted intent is not found in the intents data, it falls back to a default response.
6. **Creating the GUI**:
   - The GUI is created using the `tkinter` library, which provides widgets for building desktop applications.
   - It creates a main window (`root`) with a fixed size and title.
   - Components like `Text` (for displaying chat history), `Scrollbar` , `Button` , and another `Text` (for entering messages) are created and configured.
   - The `send()` function is bound to the Send button, which sends the user's message, predicts a response, and updates the chat history accordingly.
7. **Sending Messages**:
   - The `send()` function is called when the user clicks the Send button.
   - It retrieves the message entered by the user, clears the input field, and displays the user's message in the chat history.
   - It then predicts a response using the chatbot model and displays the response in the chat history.
   - Finally, it disables the chat history to prevent user input and scrolls to the bottom to show the latest message.
8. **Running the GUI**:
   - The `root.mainloop()` function starts the GUI event loop, which waits for user interactions like button clicks and updates the GUI accordingly.

This script integrates the chatbot model with a graphical user interface, allowing users to interact with the chatbot in a more user-friendly way. It encapsulates the functionalities of the chatbot within a visually appealing interface, enhancing the user experience.

In [5]:

```python
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import pickle
import numpy as np
```

In [6]:

```python
from keras.models import load_model
model = load_model('C:/Users/anike/OneDrive/Desktop/Projects/Machine Learning/Chatbot/chatbot_model.h5')
import json
import random
# Load intents data from JSON file
with open('C:/Users/anike/OneDrive/Desktop/Projects/Machine Learning/Chatbot/chat_json.json', encoding='
    intents = json.load(file)
words = pickle.load(open('words.pkl','rb'))
classes = pickle.load(open('classes.pkl','rb'))
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_m
etrics` will be empty until you train or evaluate the model.
```

In [7]:

```
def clean_up_sentence(sentence):
    # tokenize the pattern - splitting words into array
    sentence_words = nltk.word_tokenize(sentence)
    # stemming every word - reducing to base form
    sentence_words = [lemmatizer.lemmatize(word.lower()) for word in sentence_words]
    return sentence_words
```

In [8]:

```
# return bag of words array: 0 or 1 for words that exist in sentence
def bag_of_words(sentence, words, show_details=True):
    # tokenizing patterns
    sentence_words = clean_up_sentence(sentence)
    # bag of words - vocabulary matrix
    bag = [0]*len(words)
    for s in sentence_words:
        for i,word in enumerate(words):
            if word == s:
                # assign 1 if current word is in the vocabulary position
                bag[i] = 1
                if show_details:
                    print ("found in bag: %s" % word)
    return(np.array(bag))
```

In [9]:

```
def predict_class(sentence):
    # filter below  threshold predictions
    p = bag_of_words(sentence, words,show_details=False)
    res = model.predict(np.array([p]))[0]
    ERROR_THRESHOLD = 0.25
    results = [[i,r] for i,r in enumerate(res) if r>ERROR_THRESHOLD]
    # sorting strength probability
    results.sort(key=lambda x: x[1], reverse=True)
    return_list = []
    for r in results:
        return_list.append({"intent": classes[r[0]], "probability": str(r[1])})
    return return_list

def getResponse(ints, intents_json):
    intent = ints[0]['intent']
    for intent_obj in intents_json['intents']:
        if intent_obj['intent'] == intent:
            result = random.choice(intent_obj['responses'])
            return result
    # If the intent is not found, return a default response
    return random.choice(intents_json['default_intent']['responses'])
```

In [10]:

```
#Creating tkinter GUI
import tkinter
from tkinter import *

def send():
    msg = EntryBox.get("1.0",'end-1c').strip()
    EntryBox.delete("0.0",END)

    if msg != '':
        ChatBox.config(state=NORMAL)
        ChatBox.insert(END, "You: " + msg + '\n\n')
        ChatBox.config(foreground="#446665", font=("Verdana", 12 ))

        ints = predict_class(msg)
        res = getResponse(ints, intents)

        ChatBox.insert(END, "Bot: " + res + '\n\n')

        ChatBox.config(state=DISABLED)
        ChatBox.yview(END)


root = Tk()
root.title("Chatbot")
root.geometry("400x500")
root.resizable(width=FALSE, height=FALSE)
```

Out[10]:

```
''
```

In [13]:

```
# Import the necessary modules
import tkinter as tk
from tkinter import Scrollbar, Text, Button

# Function to send a message
def send():
    # Get the message from the EntryBox
    msg = EntryBox.get("1.0",'end-1c').strip()
    # Clear the EntryBox
    EntryBox.delete("0.0",tk.END)

    if msg != '':
        # Display user message in ChatBox
        ChatBox.config(state=tk.NORMAL)
        ChatBox.insert(tk.END, "You: " + msg + '\n\n')
        ChatBox.config(foreground="#446665", font=("Arial", 12))

        # Get response from the chatbot and display in ChatBox
        ints = predict_class(msg)
        res = getResponse(ints, intents)
        ChatBox.insert(tk.END, "Bot: " + res + '\n\n')

        # Disable ChatBox after displaying message
        ChatBox.config(state=tk.DISABLED)
        # Scroll ChatBox to the bottom
        ChatBox.yview(tk.END)

# Create the main window
root = tk.Tk()
root.title("Chatbot")
root.geometry("400x500")
root.resizable(width=False, height=False)

# Create ChatBox
ChatBox = Text(root, bd=0, bg="white", height="8", width="50", font="Arial")
ChatBox.config(state=tk.DISABLED)

# Bind scrollbar to ChatBox
scrollbar = Scrollbar(root, command=ChatBox.yview, cursor="heart")
ChatBox['yscrollcommand'] = scrollbar.set

# Create Button to send message
SendButton = Button(root, font=("Verdana",12,'bold'), text="Send", width="12", height=5,
                    bd=0, bg="#f9a602", activebackground="#3c9d9b",fg='#000000',
                    command=send)

# Create the box to enter message
EntryBox = Text(root, bd=0, bg="white", width="29", height="5", font="Arial")

# Place all components on the screen
scrollbar.place(x=376, y=6, height=386)
ChatBox.place(x=6, y=6, height=386, width=370)
EntryBox.place(x=128, y=401, height=90, width=265)
SendButton.place(x=6, y=401, height=90)

root.mainloop()
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
```