```
import tkinter as tk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
import nltk
nltk.download('punkt')
nltk.download('stopwords')

# Dummy data
doctors = {
  "Dr. Smith": ["10:00", "11:00"],
  "Dr. Alice": ["12:00", "13:00"],
  "Dr. Max": ["12:00", "14:00"]
}
available_dates = ["2025-05-01", "2025-05-02", "2025-05-03"]
appointments = []
bot_state = {}

# Preprocess input
def preprocess(text):
  tokens = word_tokenize(text.lower())
  return [t for t in tokens if t.isalpha() and t not in stopwords.words('english')]

# Chatbot logic
def chatbot_response(user_input):
  if "show appointments" in user_input.lower():
    if not appointments:
      return "No appointments found."
    return "\n".join([f"{a['name']} - {a['doctor']} at {a['time']} on {a['date']}" for a in
appointments])

  tokens = preprocess(user_input)

  if any(w in tokens for w in ["hi", "hello"]):
    return "Hello! How can I assist you today?"
  if "thank" in tokens:
    return "You're welcome!"
  if "sorry" in tokens:
    return "No problem."

  if ("don't book" in user_input.lower() or "do not book" in user_input.lower() or
    ("book" in tokens and ("don't" in tokens or "not" in tokens))):
```

```
        return "Okay, no appointment booked."

    if ("cancel appointment" in user_input.lower() or ("cancel" in tokens and
"appointment" in tokens)) and bot_state.get("step") is None:
        bot_state["step"] = "cancel_name"
        return "Please provide the patient's name to cancel the appointment."

    if bot_state.get("step") == "cancel_name":
        for appt in appointments:
            if appt["name"].lower() == user_input.lower():
                appointments.remove(appt)
                bot_state.clear()
                return "Appointment cancelled. Let me know when you would like to book."
        bot_state.clear()
        return "No matching appointment found to cancel."

    if ("reschedule appointment" in user_input.lower() or "reschedule" in tokens) and
bot_state.get("step") is None:
        bot_state["step"] = "reschedule_name"
        return "Please provide the patient's name to reschedule the appointment."

    if bot_state.get("step") == "reschedule_name":
        for appt in appointments:
            if appt["name"].lower() == user_input.lower():
                appointments.remove(appt)
                bot_state["name"] = appt["name"]
                bot_state["step"] = "date"
                return f"Sure, {appt['name']}. Please provide a new date: {',
'.join(available_dates)}"
        bot_state.clear()
        return "No appointment found for that name to reschedule."

    if ("book" in tokens or "appointment" in tokens) and bot_state.get("step") is None:
        bot_state["step"] = "name"
        return "Please provide your name to book an appointment."

    if bot_state.get("step") == "name":
        bot_state["name"] = user_input
        bot_state["step"] = "date"
        return f"Hi {user_input}, choose a date: {', '.join(available_dates)}"
```

```
  if bot_state.get("step") == "date":
    if user_input not in available_dates:
      return "Date not available. Try another."
    bot_state["date"] = user_input
    bot_state["step"] = "doctor"
    return f"Available doctors: {', '.join(doctors.keys())}"

  if bot_state.get("step") == "doctor":
    if user_input not in doctors:
      return "Doctor not available. Try again."
    bot_state["doctor"] = user_input
    bot_state["step"] = "time"
    return f"Available times: {', '.join(doctors[user_input])}"

  if bot_state.get("step") == "time":
    chosen_time = user_input
    doctor = bot_state["doctor"]
    date = bot_state["date"]

    # Check if already booked
    for appt in appointments:
      if appt["doctor"] == doctor and appt["date"] == date and appt["time"] ==
chosen_time:
        bot_state["step"] = "suggest_time"
        return f"{doctor} is already booked at {chosen_time} on {date}. Would you like to
schedule it for some other time?"

    if chosen_time not in doctors[doctor]:
      return "Time not available. Try again."

    bot_state["time"] = chosen_time
    bot_state["step"] = "confirm"
    return (f"Confirm appointment with {doctor} at {chosen_time} on {date}? (yes/no)")

  if bot_state.get("step") == "suggest_time":
    if "yes" in user_input.lower():
      doctor = bot_state["doctor"]
      date = bot_state["date"]
      booked = [a["time"] for a in appointments if a["doctor"] == doctor and a["date"] ==
date]
      available = [t for t in doctors[doctor] if t not in booked]
```

```
            if not available:
                bot_state.clear()
                return f"Sorry, no other times available for {doctor} on {date}."
            bot_state["step"] = "time"
            return f"Available times for {doctor} on {date}: {', '.join(available)}"
        else:
            bot_state.clear()
            return "Okay, appointment not scheduled. Let me know if you'd like to try again."

    if bot_state.get("step") == "confirm":
        if "yes" in user_input.lower():
            appointments.append({
                "name": bot_state["name"],
                "date": bot_state["date"],
                "doctor": bot_state["doctor"],
                "time": bot_state["time"]
            })
            bot_state.clear()
            return "Appointment confirmed!"
        else:
            bot_state.clear()
            return "Okay, no appointment booked."

    return "I can help book, cancel, reschedule appointments, or answer FAQs. You can also type 'show appointments'."

# UI setup
window = tk.Tk()
window.title("Appointment Chatbot")

chat_log = tk.Text(window, height=20, width=70)
chat_log.pack()

entry = tk.Entry(window, width=70)
entry.pack()

def send():
    user_input = entry.get()
    chat_log.insert(tk.END, f"You: {user_input}\n")
    response = chatbot_response(user_input)
    chat_log.insert(tk.END, f"Bot: {response}\n\n")
```

```
    entry.delete(0, tk.END)

button = tk.Button(window, text="Send", command=send)
button.pack()

chat_log.insert(tk.END, "Bot: Hello! How can I assist you today?\n\n")

window.mainloop()
```