SOPHIA RESPONSE:

```
recognizer = speech_recognition.Recognizer()

# Initialize pyttsx3

speaker = tts.init()

# Set properties

voices = speaker.getProperty('voices')

speaker.setProperty('voice', voices[1].id)

todo_list = []

reminder_list = []

def generic_respond():

 ints = assistant._predict_class(message)
```

```
 intents = json.loads(open('intents.json').read())

 res = assistant._get_response(ints, intents)

 print(res)

 # Convert text to speech

 speaker.say(res)

 # Speak the text

 speaker.runAndWait()

def create_note():

 global recognizer

 generic_respond()

 done = False
```

```
20
 while not done:
 try:
 with
speech_recognition.Microphone()
as mic:

recognizer.adjust_for_ambient_noi
se(mic, duration=0.5)

recognizer.dynamic_energy_thresh
old = True
 audio = recognizer.listen(mic)
 note =
recognizer.recognize_google(audio
)
```

```
 note.lower()
 print(note)
 print("Give a file name")
 speaker.say("Give a file name")
 speaker.runAndWait()

recognizer.adjust_for_ambient_noise(mic, duration=0.5)

recognizer.dynamic_energy_threshold = True
 audio = recognizer.listen(mic)
 filename = recognizer.recognize_google(audio)
```

```
 filename = filename.lower()
 print(" " * 8 + filename)
 with open(filename, 'w') as anote:
 anote.write(note)
 done = True
 print(f"Your note {filename} has been created successfully")
 speaker.say(f"Your note {filename} has been created successfully")
 speaker.runAndWait()
 except speech_recognition.UnknownValueError:
```

```
 recognizer = speech_recognition.Recognizer()
 print("I didn't get you. Please try again!")
 speaker.say("I didn't get you. Please try again!")
 speaker.runAndWait()
def about():
 generic_respond()
```



```
def jokes():
 generic_respond()
 joke = pyjokes.get_joke(language='en', category='neutral')
```

```
 print(joke)
 speaker.say(joke)
def quit():
 generic_respond()
 sys.exit(0)
def greetings():
 generic_respond()
def add_reminders():
 global recognizer
 generic_respond()
 try:
 with
speech_recognition.Microphone()
as mic:
```

```
recognizer.adjust_for_ambient_noise(mic, duration=0.5)

recognizer.dynamic_energy_threshold = True
 audio = recognizer.listen(mic)
 reminder = recognizer.recognize_google(audio)
 reminder = reminder.lower()
 print(reminder)
 reminder_list.append(reminder)
 except speech_recognition.UnknownValueError:
```

```
 recognizer = speech_recognition.Recognizer()
 try:
 reminder_dict = json.load(open("reminders.json"))
 except json.JSONDecodeError:
 reminder_dict = dict()
 for i in range(len(reminder_dict), len(reminder_list)):
 reminder_dict[i] = reminder_list[i]
 json.dump(reminder_dict, open("reminders.json", 'w'))
22
def show_reminders():
 generic_respond()
```

```
 reminder_dict =
json.load(open("reminders.json"))
 for i in range(len(reminder_dict)):
 print(reminder_dict[str(i)])

speaker.say(str(reminder_dict[str(i
)]))
 speaker.runAndWait()
def add_todo():
 global recognizer
 todo_lst = []
 new = str()
 try:
 todo = open("todo.txt", "r+")
 raw_str = todo.read()
```

```
if len(raw_str) != 0:
# print("case 1")
if "," in raw_str:
# print("case 2")
todo_templst = raw_str.split(',')
for x in todo_templst:
todo_lst.append(str(x))
if "" in todo_lst:
todo_lst.remove('')
else:
# print("else 2.1")
todo_templst = raw_str
todo_lst.append(todo_templst)
else:
```

```
 # print("else")
 todo = open("todo.txt", "w")
 # print("#", todo_lst)
 except FileNotFoundError:
 todo = open("todo.txt", "w")
23
 todo.close()
 try:
 todo = open("todo.txt", "w")
 with
speech_recognition.Microphone()
as mic:

recognizer.adjust_for_ambient_noi
se(mic, duration=0.5)
```

```
recognizer.dynamic_energy_threshold = True
 speaker.say("Tell me what is your todo")
 print("Tell me what is your todo")
 speaker.runAndWait()
 audio = recognizer.listen(mic)
 todos = recognizer.recognize_google(audio)
 todos.lower()
 new = todos
 todo_lst.append(todos)
 # print(todo_lst)
```

```
except Exception as e:
print(e)
for i in range(len(todo_lst)):
temp = str(todo_lst[i])
print(temp)
todo.write('%s,' % temp)
todo.close()
speaker.say(f"Your new todo {new} has been added successfully")
print(f"Your new todo {new} has been added successfully")
speaker.runAndWait()
def shutdown():
global recognizer
```

```
generic_respond()
 try:
 with
speech_recognition.Microphone()
as mic:

recognizer.adjust_for_ambient_noi
se(mic, duration=0.5)

recognizer.dynamic_energy_thresh
old = True
 audio = recognizer.listen(mic)
 confirmation =
recognizer.recognize_google(audio
)
```

```
 confirmation.lower()
 print(confirmation)
 except
speech_recognition.UnknownValu
eError:
 recognizer =
speech_recognition.Recognizer()
 if confirmation in "yes":
 if WORKING_PLATFORM ==
"Windows":
 return os.system("shutdown /s /t
1")
 else:
 print("another platform!")
```

```
 else:
 speaker.say("Confirmation rejected")
 speaker.runAndWait()
 return 0
def restart():
 global recognizer
 generic_respond()
 try:
 with speech_recognition.Microphone() as mic:

recognizer.adjust_for_ambient_noise(mic, duration=0.5)
```

```
recognizer.dynamic_energy_threshold = True
 audio = recognizer.listen(mic)
 confirmation = recognizer.recognize_google(audio)
 confirmation.lower()
 print(confirmation)
 except speech_recognition.UnknownValueError:
 recognizer = speech_recognition.Recognizer()
 if confirmation in "yes":
```

```
 return os.system("shutdown /r /t 1")
 else:
 speaker.say("Confirmation rejected")
 speaker.runAndWait()
 return 0
25
def logout():
 global recognizer
 generic_respond()
 try:
 with speech_recognition.Microphone() as mic:
```

```
recognizer.adjust_for_ambient_noise(mic, duration=0.5)

recognizer.dynamic_energy_threshold = True
 audio = recognizer.listen(mic)
 confirmation = recognizer.recognize_google(audio)
 confirmation.lower()
 print(confirmation)
 except speech_recognition.UnknownValueError:
```

```
 recognizer = speech_recognition.Recognizer()
 if confirmation in "yes":
 return os.system("shutdown -l")
 else:
 speaker.say("Confirmation rejected")
 speaker.runAndWait()
 return 0
def hibernate():
 global recognizer
 generic_respond()
 try:
 with speech_recognition.Microphone()
```

```
as mic:

recognizer.adjust_for_ambient_noise(mic, duration=0.5)

recognizer.dynamic_energy_threshold = True
 audio = recognizer.listen(mic)
 confirmation = recognizer.recognize_google(audio)
 confirmation.lower()
 print(confirmation)
 except speech_recognition.UnknownValueError:
```

```
 recognizer =
speech_recognition.Recognizer()
 if confirmation in "yes":
 return os.system("shutdown /h")
 else:
26
 speaker.say("Confirmation
rejected")
 speaker.runAndWait()
 return 0
def lock_screen():
 global recognizer
 generic_respond()
 try:
```

```
 with
speech_recognition.Microphone()
as mic:

recognizer.adjust_for_ambient_noi
se(mic, duration=0.5)

recognizer.dynamic_energy_thresh
old = True
 audio = recognizer.listen(mic)
 confirmation =
recognizer.recognize_google(audio
)
 confirmation.lower()
 print(confirmation)
```

```
 except
speech_recognition.UnknownValu
eError:
 recognizer =
speech_recognition.Recognizer()
 if confirmation in "yes":
 return os.system("rundll32.exe
user32.dll, LockWorkStation")
 else:
 speaker.say("Confirmation
rejected")
 speaker.runAndWait()
 return 0
def queries():
 try:
```

```
 result =
wikipedia.summary(message,
sentences=5)

 print("According to wikipedia : ",
result)

 speaker.say("According to
wikipedia" + result)

 speaker.runAndWait()

 except Exception as e:

 print("Here's the search result")

 speaker.say("Here's the search
result")

 speaker.runAndWait()


webbrowser.open_new("www.goo
```

```
gle.com/search?q=" + message)
27
def name_call():
 pass
def health_assistant():
 getSeverityDict()
 getDescription()
 getprecautionDict()
 # getInfo()
 tree_to_code(clf, cols)
mappings = {
 "greetings": greetings,
 "create_note": create_note,
 "jokes": jokes,
```

"name_call": name_call,

"about": about,

"exit": quit,

"add_reminders": add_reminders,

"show_reminders": show_reminders,

"shut_down": shutdown,

"restart": restart,

"sleep": hibernate,

"log_out": logout,

"lock_screen": lock_screen,

"queries": queries,

"add_todo": add_todo,

"health_assistant":
health_assistant

}

assistant =
ModelAssistant('intents.json',
intent_methods=mappings)

assistant.load_model()

speaker.say("Hey!, I am here. How can I help you?")

speaker.save_to_file("Hi. I am Sophia. your personal Virtual Assistant", 'sophia.mp3')

28

print("Hey!, I am here. How can I help you?")

```
speaker.runAndWait()
speaker.runAndWait()
while True:
 try:
 with speech_recognition.Microphone() as mic:

recognizer.adjust_for_ambient_noise(mic, duration=0.5)

recognizer.dynamic_energy_threshold = True
 print("Listening...")
```

```
 speaker.say("Listening...")
 speaker.runAndWait()
 audio = recognizer.listen(mic)
 message =
recognizer.recognize_google(audio
)
 message = message.lower()
 print(message)
 assistant.request(message)
 except
speech_recognition.UnknownValu
eError:
 recognizer =
speech_recognition.Recognizer()
 print("Sorry, say it again!")
```

```
speaker.say("Sorry, say it again!")
speaker.runAndWait()
```