Add cover Customize layout

Jarvis-Nova

+ Add a property

Comments

s Add a comment...

Write, press 'space' for AI, '/' for commands...

Here is a brief explanation of my code:

IMPORTING LIBRARIES:

- `os` : For system-level operations (e.g., opening files, shutdown).
- `requests` : Used to call APIs (for DeepSeek AI).
- `speech_recognition` (`sr`): For converting voice to text.
- `webbrowser` : To open websites in your default browser.
- `smtplib` : Used for sending emails.
- `time` : For time/date-related functions (e.g., alarm).
- `wikipedia` : Fetches info from Wikipedia.
- `random` : For randomly choosing responses (makes Nova feel less robotic).
- `pyttsx3` : Text-to-speech engine to make Nova talk.

TEXT to SPEECH:

```python
def speak(text):
    print(f"Nova: {text}")
    engine = pyttsx3.init()
    voices = engine.getProperty('voices')
    engine.setProperty('voice', voices[1].id)  # optional: change voice
    engine.setProperty('rate', 170)
    engine.say(text)
    engine.runAndWait()
    engine.stop()
```

- Converts any `text` into **spoken voice**.
- Uses a **female voice** (`voices[1]`) and sets a speaking speed.
- Also prints the spoken text to console.

STARTUP AND SHUTDOWN RESPONSES:

Python Copy Caption

```python
startup_lines = [
    "Welcome back, Nova's here.",
    "All systems primed and ready,ma'am."
]

mid_responses = [
    "Right away.", "Looking it up now.", "Just a moment.", "Certainly.",
    "On it, ma'am.", "As you wish.", "Fetching that for you.",
    "Give me a sec...", "Here you go.", "Already ahead of you.",
    "Consider it done.", "Scanning now...", "Processing complete."
]

shutdown_lines = [
    "Going dark. Call me when you need me.",
    "System powering down. Until next time, ma'am.",
    "Shutting down. Nova out."
]
```

- These are **predefined voice lines** that make Nova feel **more dynamic and human-like**.
- Randomly selected and spoken at different stages.

DEEPSEEK API KEY:

```
DEEPSEEK_API_KEY = "PASTE YOUR API KEY HERE"
```

Your personal API key to talk to Deep Seek AI for fallback queries (like a Chat GPT backup).

LISTEN FOR WAKE WORD FUNCTION:

```
def listen_for_wake_word(wake_word="nova"):
    recognizer = sr.Recognizer()
    with sr.Microphone() as source:
        recognizer.adjust_for_ambient_noise(source, duration=0.3)

        if listening:
            speak("Listening for wake word")
        print("🎧 Listening for wake word...")  # always print, so user knows it's active

        try:
            audio = recognizer.listen(source, timeout=5, phrase_time_limit=7)
            heard = recognizer.recognize_google(audio).lower()
            print("Heard:", heard)

            if listening:
                speak(f"You said: {heard}")
            return heard
        except sr.WaitTimeoutError:
            print("Timeout: No voice detected.")
        except sr.UnknownValueError:
            print("Couldn't understand the audio.")
        except sr.RequestError:
            print("Speech service unavailable.")
        return None
```

- Uses your **microphone to listen for "nova"**.
- If it hears the word, it returns what was said.
- It handles **timeouts**, **no speech**, and **API errors** gracefully.

LISTEN FOR COMMAND FUNCTION:

```
def listen_command():
    recognizer = sr.Recognizer()
    with sr.Microphone() as source:
        recognizer.adjust_for_ambient_noise(source, duration=0.3)
        speak("Listening for command")
        print("🎤 Listening for command...")
        try:
            audio = recognizer.listen(source, timeout=5, phrase_time_limit=7)
            command = recognizer.recognize_google(audio)
            speak(f"You said: {command}")  # 🔊 Now Nova will speak the command
            print("You said:", command)
            return command.lower()
        except sr.UnknownValueError:
            speak("Sorry, I couldn't understand.")
            return ""
        except sr.WaitTimeoutError:
            speak("Timeout. I didn't hear anything.")
            return ""
        except sr.RequestError:
            speak("Sorry, there was a speech service error.")
            return ""
```

- After detecting the wake word, this function **listens for your command** (e.g., "open YouTube").
- Converts the audio to text and returns it.

PLAY MUSIC:

Python

Copy Caption ···

```python
def play_music():
    playlist_url = "https://open.spotify.com/playlist/6QT22ZcbGuaFZkRowjIGRE"
    try:
        webbrowser.open(playlist_url)
        speak("Opening your Spotify playlist.")
    except:
        speak("Sorry, I couldn't open the playlist.")
```

- Opens your Spotify playlist in the default browser.

WIKIPEDIA SEARCH:

Python Copy Caption

```python
def search_meaning(query):
    try:
        query = query.replace("what do you mean by", "").replace("who is", "").replace("s
        speak(f"Searching for {query}...")
        result = wikipedia.summary(query, sentences=2)
        print(result)
        speak("Here you go.")
        speak(result)
    except wikipedia.exceptions.DisambiguationError:
        speak("Hmm, there are multiple results. Please be more specific.")
    except wikipedia.exceptions.PageError:
        speak("Sorry, I couldn't find any results.")
```

- Strips common phrases like "what is", "who is", etc.
- Searches for the keyword on Wikipedia.
- Returns a brief 2-line summary.

EXECUTE COMMANDS:

Python Copy Caption ···

```python
def execute_command(command):
    command = command.lower()

    if "stop listening" in command:
        speak("Going silent. Say 'start listening' to wake me up.")
        global listening
        listening = False
        return

    if "start listening" in command:
        speak("Back online and listening.")
        listening = True
        return

    if "shutdown" in command:
        speak(random.choice(shutdown_lines))
        os.system("shutdown /s /t 1")

    elif "restart" in command:
        speak("Restarting.")
        os.system("shutdown /r /t 1")

    elif "notepad" in command:
        os.system("notepad")

    elif "music" in command:
        play_music()

    elif "open google" in command:
        webbrowser.open("https://www.google.com")
        speak("Opening Google.")

    elif "open youtube" in command:
        webbrowser.open("https://www.youtube.com")
        speak("Opening YouTube.")
```

```
    elif "send email" in command:
        send_email()

    elif "time" in command:
        now = time.strftime("%I:%M %p")
        speak(f"The time is {now}.")

    elif "date" in command:
        today = time.strftime("%A, %B %d, %Y")
        speak(f"Today is {today}.")

    elif "open chatgpt" in command:
        webbrowser.open("https://chat.openai.com")
        speak("Opening ChatGPT.")

    elif "search for" in command or "what do you mean by" in command or "who is" in comma
        search_meaning(command)

    elif "open website" in command:
        speak("Which website should I open?")
        site = listen_command().replace(" ", "")
        webbrowser.open(f"https://{site}")
        speak(f"Opening {site}.")

    elif "take a note" in command or "make a note" in command:
        speak("What should I note down?")
        note = listen_command()
        with open("nova_notes.txt", "a") as f:
            f.write(f"{time.ctime()} - {note}\n")
        speak("Note saved.")

    elif "open downloads" in command:
        downloads = os.path.join(os.path.expanduser("~"), "Downloads")
        os.startfile(downloads)
        speak("Opening Downloads folder.")

    elif "set alarm at" in command:
        try:
            alarm_time = command.replace("set alarm at", "").strip()
            alarm_hour, alarm_minute = map(int, alarm_time.split(":"))
```

```
            speak(f"Alarm set for {alarm_hour}:{alarm_minute}.")
            while True:
                now = time.localtime()
                if now.tm_hour == alarm_hour and now.tm_min == alarm_minute:
                    speak("Wake up! Alarm time.")
                    break
                time.sleep(20)
        except:
            speak("Sorry, I couldn't understand the alarm time.")

    else:
        response = ask_deepseek(command)
        speak(response)
```

- This is the **main action handler**.
- Based on your command, it performs tasks like:
  - `shutdown`, `restart`, `open notepad`, `play music`, `search`, etc.
- Handles stop/start listening mode
- Some commands like `search`, `note taking`, `setting alarm`, and `open website` are interactive.

MAIN LOOP

```python
def main_loop():
    global listening
    speak(random.choice(startup_lines))

    while True:
        if listening:
            print("🎧 Listening for wake word...")
            heard = listen_for_wake_word("nova")
            if heard and "nova" in heard.lower():
                print(f"You said: {heard}")
                # Removed verbal response
                command = listen_command()
                if command:
                    print(f"You said: {command}")
                    execute_command(command)
        else:
            print("🔴 Nova is silent. Say 'start listening' to wake me up.")
            heard = listen_for_wake_word("start listening")
            if heard and "start listening" in heard.lower():
                speak("Nova back online.")
                listening = True
```

- The core loop where Nova:
  - **Waits for the wake word** (like "Nova").
  - **Listens to the command**, and
  - **Executes the command** using `execute_command()`.
- If Nova is in silent mode (`listening = False`), it only listens for `"start listening"` and resumes.

START LOOP

```
if __name__ == "__main__":
    main_loop()
```

- Entry point of the program.
- This runs the main loop when you do `python jarvis.py`.