SuperU Voice Agent Task Report – Koushik BM

---

1. How the Code Works

The SuperU Voice Agent is a voice-based meeting scheduler that interacts with users through speech, collects meeting details, checks calendar availability, and schedules events in Google Calendar. Below is the breakdown of the process:

- **Speech Interaction**: The agent uses pyttsx3 for text-to-speech and pyaudio for recording user responses. It prompts the user for inputs and records their voice.
- **Transcription**: Recorded audio is transcribed to text using Groq's Whisper API.
- **Data Collection and Validation**:
  - The agent collects the user's email, meeting topic, and preferred time.
  - Email is cleaned and validated using regex.
  - Meeting time is parsed using dateutil.parser with timezone adjustments.
- **Confirmation**: After gathering all data, the agent repeats the inputs and asks the user for confirmation. If the user declines, the process restarts.
- **Calendar Integration**:
  - The agent checks for availability using the Google Calendar API.
  - If the slot is available and the user confirms, the meeting is created and stored in the calendar.
- **Looping for Multiple Meetings**: The agent offers the option to schedule another meeting. If the user declines, the session ends.

---

2. Changes Made from the Task

The original task instructions provided some flexibility, but the following are specific changes or alternatives I implemented:

- Used Google Calendar API instead of Composio: The task suggested integrating with Composio or similar tools. I chose to use the Google Calendar API directly, as it provides robust support for checking availability and scheduling events with OAuth2 authentication.
- Used Groq Whisper API for Speech-to-Text: Instead of a local or default speech-to-text engine, I integrated Groq's hosted Whisper large-v3-turbo model to provide accurate transcription of voice inputs.
- Used Local Voice Stack (pyttsx3 and PyAudio): For speech synthesis and input, I used pyttsx3 (for TTS) and pyaudio (for audio capture) instead of relying on any cloud-based voice agents.
- Added Input Confirmation Step: The original task required fallback and clarification, but I also implemented a final step that reads back all user-provided details and asks for confirmation before proceeding.

- Handled Timezone Adjustments: Implemented local time parsing and explicit UTC conversion to ensure accurate scheduling across time zones.
- Validated Meeting Time Constraints: Added logic to reject scheduling of meetings in the past or within 10 minutes of the current time, to ensure realistic scheduling.

---

3. Technologies Used

1. Python

- Base programming language used to build and orchestrate the voice agent logic.

2. Pyttsx3

- Text-to-speech engine used to vocalize agent prompts to the user.

3. PyAudio

- Used to record audio input from the user's microphone for transcription.

4. Groq Whisper API

- Converts audio to text using the Whisper large-v3-turbo model hosted by Groq.

5. Vocode (Custom Agent)

- RespondAgent from vocode.streaming.agent provides the backbone for defining and running conversational agents.

6. Google Calendar API

- Authenticates using OAuth2.
- Checks availability and creates calendar events using google-api-python-client.

7. Dateutil

- Used for parsing and manipulating natural language dates and times.
- Handles timezone conversion.

8. Dotenv

- Loads sensitive environment variables such as API keys from .env file.

9. Logging

- Provides structured log output for tracking and debugging.

---

4. Task Checklist with Code References

Task: Set up a Voice Agent to Talk to Users and Schedule Meetings

1. Vocode.dev Setup

- Agent defined using GroqVoiceAgent class in vocode_custom_agent.py.

- Integrated into main loop via agent = GroqVoiceAgent(GroqAgentConfig(type="custom")).

2. Agent Workflow Logic

- Email Collection: ask_and_transcribe("Please tell me your email address."...)
- Topic Collection: ask_and_transcribe("What is the topic of the meeting?"...)
- Time Collection: ask_and_transcribe("When would you like to schedule the meeting?"...)
- Time Parsing: parse_time() with timezone handling.
- Input Confirmation: speak(f"To confirm, your email is {email}...)`
- Retry on Invalid Input: 3 attempts in ask_and_transcribe()

3. Integration with Composio or Similar

- Google Calendar API is used as the integration tool.
- Availability Check: check_calendar_availability()
- Event Creation: create_google_event()
- Token management handled via token.json with OAuth2

4. Secure Data Handling

- API keys are stored in .env
- Tokens are not hardcoded
- Meeting details saved locally in meetings.txt for backup/debugging

---

The development of this vocode agent reflects a culmination of the skills and technologies I've gained through previous projects. Initially, the task called for the use of paid API services, but due to cost constraints, I sought more efficient alternatives. After careful consideration, I leveraged pyttsx3 from my Zeno.V project as the core speech synthesis tool. This provided a reliable and lightweight solution, aligning well with my objectives. Additionally, I integrated Groq, an open-source alternative to OpenAI's offerings, which not only met my technical requirements but also provided Whisper for robust speech-to-text functionality Groq was also used in my Language Learning project. This combination of tools allowed me to create a fully functional vocode agent without the need for costly API subscriptions, demonstrating the effectiveness of open-source solutions in achieving complex tasks efficiently.