

The Smart Scheduler AI Agent

A Voice-Based Gemini-powered Meeting Scheduling Assistant

Objective

To build a free and intelligent AI voice agent that helps users schedule meetings through a natural back-and-forth voice-based conversation, while integrating Google Calendar to check real-time availability.

The project tests multi-turn dialogue management, tool integration, and reasoning capabilities of modern LLMs.

Key Features

- Conversational AI using Gemini 1.5 Flash
- Google Calendar API v3 integration to check actual time slots
- Voice input using SpeechRecognition (STT)
- Voice response using pyttsx3 (TTS)
- Maintains memory of meeting duration and time preference
- Fully local and free to run — no paid services
- Ignores vague inputs or suggests clarification
- Terminal/CLI-based working prototype

Tools & Stack

Component	Tool / API
Language Model	Google Gemini 1.5 Flash
Calendar Access	Google Calendar API v3
Text-to-Speech (TTS)	pyttsx3 (offline)
Speech-to-Text (STT)	Speech Recognition (CMU Sphinx / Google STT)
Programming Language	Python 3.10+
Auth & Secrets	OAuth2 + dotenv

Sample Conversation

You: I want to schedule a meeting

Agent: How long should the meeting be?

You: 1 hour

Agent: Which day and time do you prefer?

You: Tuesday afternoon

Agent: I found these available slots:

- 12:00 PM

- 3:30 PM

You: Let's go with 3:30 PM

Agent: Great. I'll save it to your calendar.

Architecture Overview

User ↔ [Voice Input] → SmartScheduler Agent ↔ Gemini API



Google Calendar API ← Available Slots



Voice Output (TTS)

Screenshots

Text based AI Scheduler agent

```
EXPLORER
MY_SCHEDULER_AI
  > __pycache__
  > .venv
  > .env
  > .gitignore
  > calendar_utils.py 3, U
  {} credentials.json U
  {} README.md U
  requirements.txt
  scheduler_agent.py 1
  test_agent.py 2, M
  test_test_agent.py U
  {} token.json U

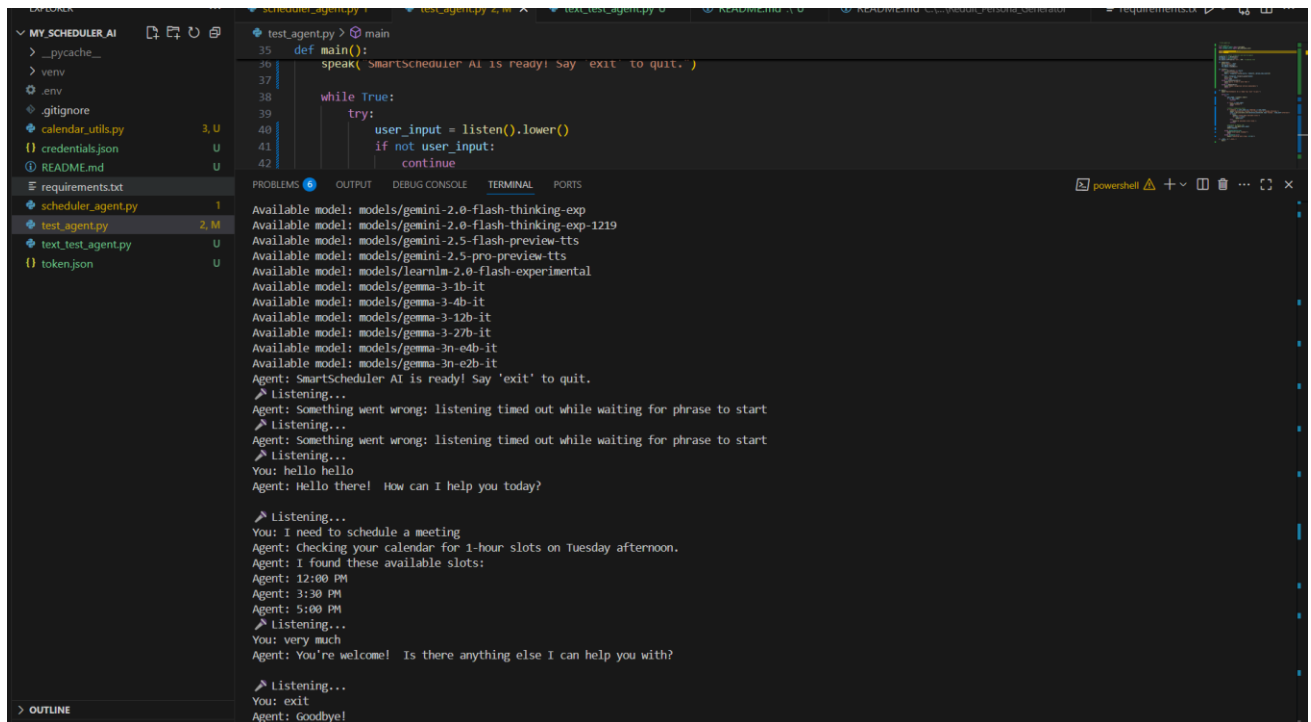
text_test_agent.py > main
1 # text_test_agent.py
2 from scheduler_agent import ask_agent
3 from calendar_utils import get_available_slots
4
5 def main():
6     print("SmartScheduler AI is ready! Type 'exit' to quit.")
7     print("You can ask things like: 'I need to schedule a meeting.'")
8
9     while True:
10        try:
11            user_input = input("You: ")
12            if user_input.lower() == "exit":
13                break
14
15        # Check if user wants to schedule a meeting
16        if "schedule" in user_input.lower() and "meeting" in user_input.lower():
17            print("Agent: Checking your calendar...")
18            slots = get_available_slots(duration_minutes=60, day='Tuesday', time_pref='afternoon') # You can customize
19            print("Agent: I found these available slots:")
20            for slot in slots:
21                print(f"  - {slot}")
22            continue

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Available model: models/gemma-3-27b-it
Available model: models/gemma-3n-e4b-it
Available model: models/gemma-3n-e2b-it
SmartScheduler AI is ready! Type 'exit' to quit.
You can ask things like: 'I need to schedule a meeting'
You: Hi
Agent: Hi there! How can I help you today?

You: I need to schedule a meeting
Agent: Checking your calendar...
Agent: I found these available slots:
  - 12:00 PM
  - 3:30 PM
  - 5:00 PM
You: Good. Thanks.
Agent: You're welcome! Is there anything else I can help you with?

You: exit
(venv) PS C:\Users\samuj\OneDrive\Desktop\my_projects\My_scheduler_AI>
```

Voice-based AI Scheduler agent



```
def main():
    speak('SmartScheduler AI is ready! Say \'exit\' to quit. ')
    while True:
        try:
            user_input = listen().lower()
            if not user_input:
                continue
```

Available model: models/gemini-2.0-flash-thinking-exp
Available model: models/gemini-2.0-flash-thinking-exp-1219
Available model: models/gemini-2.5-flash-preview-tts
Available model: models/gemini-2.5-pro-preview-tts
Available model: models/learnlm-2.0-flash-experimental
Available model: models/gemma-3-1b-it
Available model: models/gemma-3-4b-it
Available model: models/gemma-3-12b-it
Available model: models/gemma-3-27b-it
Available model: models/gemma-3n-e4b-it
Available model: models/gemma-3n-e2b-it
Agent: SmartScheduler AI is ready! Say 'exit' to quit.
Listening...
Agent: Something went wrong: listening timed out while waiting for phrase to start
Listening...
Agent: Something went wrong: listening timed out while waiting for phrase to start
Listening...
You: hello hello
Agent: Hello there! How can I help you today?
Listening...
You: I need to schedule a meeting
Agent: Checking your calendar for 1-hour slots on Tuesday afternoon.
Agent: I found these available slots:
Agent: 12:00 PM
Agent: 3:30 PM
Agent: 5:00 PM
Listening...
You: very much
Agent: You're welcome! Is there anything else I can help you with?
Listening...
You: exit
Agent: Goodbye!

Challenges Faced & Solved

- Gemini API versioning & model listing (404 fixed using correct model name)
- Calendar API authentication with token.json flow
- Speech-to-text accuracy tuning using thresholds
- Context management between LLM input and user preferences

Repository & Access

Click [here](#) to access GitHub repository.

Alternate link:- <https://github.com/Samartha21BRS1698/Smart-Scheduler-AI>

Project Status

- Fully working CLI prototype
 - Voice input/output integrated
 - Real Google Calendar slots
 - Bonus features like advanced time parsing and UI can be implemented.
-

Conclusion

This project demonstrates how LLMs can be combined with real-world tools like calendars, speech, and authentication to build intelligent AI agents. The Smart Scheduler AI agent is a practical assistant and a proof-of-concept for enterprise AI integration.

Project Experience

The Smart Scheduler AI Agent project was a hands-on exploration of building an intelligent, voice-based assistant capable of scheduling meetings through natural multi-turn conversation. Powered by Google's Gemini 1.5 Flash model, the agent integrated with the Google Calendar API to retrieve real-time availability and handled contextual queries such as preferred day, time, and duration. The development process deepened my understanding of prompt engineering, conversational memory, OAuth2 authentication, and real-world tool orchestration. Overcoming challenges like model version errors and calendar token handling added valuable experience in debugging API-driven workflows. Overall, this task sharpened my skills in building practical AI agents that bridge language models with external systems—skills that are increasingly critical in real-world AI applications.

Author

Samartha

samu36939@gmail.com

[LinkedIn](#)

[GitHub](#)