```
import os
import time

import pygame
import speech_recognition as sr
from dotenv import load_dotenv
from playsound import playsound

from swarms import OpenAIChat, OpenAITTS

# Load the environment variables
load_dotenv()

# Get the API key from the environment
openai_api_key = os.environ.get("OPENAI_API_KEY")

# Initialize the language model
llm = OpenAIChat(
    openai_api_key=openai_api_key,
)

# Initialize the text-to-speech model
tts = OpenAITTS(
    model_name="tts-1-1106",
    voice="onyx",
    openai_api_key=openai_api_key,
    saved_filepath="runs/tts_speech.wav",
)


# Initialize the speech recognition model
r = sr.Recognizer()



def play_audio(file_path):
    # Check if the file exists
    if not os.path.isfile(file_path):
        print(f"Audio file {file_path} not found.")
        return

    # Initialize the mixer module
    pygame.mixer.init()

    try:
        # Load the mp3 file
        pygame.mixer.music.load(file_path)

        # Play the mp3 file
        pygame.mixer.music.play()

        # Wait for the audio to finish playing
        while pygame.mixer.music.get_busy():
            pygame.time.Clock().tick(10)
    except pygame.error as e:
        print(f"Couldn't play {file_path}: {e}")
    finally:
        # Stop the mixer module and free resources
        pygame.mixer.quit()



while True:
    # Listen for user speech
    with sr.Microphone() as source:
        print("Listening...")
        audio = r.listen(source)

    # Convert speech to text
    try:
        print("Recognizing...")
        task = r.recognize_google(audio)
        print(f"User said: {task}")
    except sr.UnknownValueError:
        print("Could not understand audio")
        continue
    except Exception as e:
        print(f"Error: {e}")
        continue

    # Run the Gemini model on the task
```

```
print("Running GPT4 model...")

out = llm(task)

print(f"Gemini output: {out}")


# Convert the Gemini output to speech

print("Running text-to-speech model...")

out = tts.run_and_save(out)

print(f"Text-to-speech output: {out}")


# Ask the user if they want to play the audio

# play_audio = input("Do you want to play the audio? (yes/no): ")

# if play_audio.lower() == "yes":

# Initialize the mixer module

# Play the audio file


time.sleep(5)


playsound("runs/tts_speech.wav")
```