Integrasi Call Center Menggunakan OpenAI

1st Dimas Satrio Wibowo
Fakultas Teknik Elektro
Telkom University
Bandung, Indonesia
dimassatriowibowo@student.telkomuniversiy.ac.id

2nd Bagus Aditya
Fakultas Teknik Elektro
Telkom University
Bandung, Indonesia
goesaditya@telkomuniversity.ac.id

3rd Akhmad Hambali
Fakultas Teknik Elektro
Telkom University
Bandung, Indonesia
ahambali@telkomuniversity.ac.id

Abstrak — Seiring pesatnya perkembangan teknologi kecerdasan buatan dan komunikasi berbasis *IP*, kebutuhan akan sistem *call center* yang cerdas dan responsif semakin mendesak. Penelitian ini merancang dan mengimplementasikan skrip *Asterisk Gateway Interface (AGI)* berbasis Python yang terintegrasi dengan *OpenAI API* untuk membangun layanan *call center* otomatis tanpa campur tangan manusia. Proses dimulai dengan merekam suara penelepon, kemudian mentranskripsinya menjadi teks melalui *Microsoft Azure Speech-to-Text*. Teks tersebut selanjutnya diproses oleh model *GPT-4* pada *OpenAI* untuk menghasilkan respons kontekstual, lalu diubah kembali menjadi audio menggunakan *Azure Text-to-Speech* dan diputarkan ke penelepon—semua berlangsung secara *real-time*. Hasil pengujian menunjukkan rata-rata waktu respons *end-to-end* sekitar 18,7 detik, tingkat akurasi transkripsi 93 %, dan skor kepuasan pengguna yang tinggi. Pendekatan modular ini tidak hanya menurunkan biaya operasional dan menghapus antrean panggilan, tetapi juga menyediakan fondasi kuat bagi pengembangan layanan *multi-channel*, fitur lanjutan seperti penyimpanan riwayat percakapan, serta integrasi yang lebih fleksibel dengan sistem komunikasi modern di masa depan.Kata kunci — AGI; Python*; OpenAI API; Speech-to-Text; Text-to-Speech;* sistem *call center* otomatis.

I. PENDAHULUAN

Di era transformasi digital saat ini, kecepatan dan kualitas layanan pelanggan menjadi faktor penentu keberhasilan sebuah organisasi. *Call center* tradisional yang sepenuhnya mengandalkan operator manusia sering kali menghadapi hambatan seperti antrean panjang, inkonsistensi jawaban, dan biaya operasional tinggi. Sementara itu, kemajuan dalam bidang *VoIP (Voice over Internet Protocol)* membuka peluang bagi pengembangan solusi *call center* otomatis yang lebih fleksibel dan murah.

Asterisk, sebagai platform *VoIP open-source*, menyediakan mekanisme *Asterisk Gateway Interface* (*AGI*) untuk menjalankan skrip eksternal yang mengatur logika panggilan telepon. Dengan *AGI*, setiap langkah dalam alur panggilan mulai dari menjawab, merekam, hingga memutar suara dapat dikendalikan oleh aplikasi eksternal, misalnya skrip *Python*.

Di sisi lain, kemunculan *Large Language* Models seperti *GPT-4* yang disediakan melalui *OpenAI API*, serta layanan *Azure* Speech Services untuk *Speech-to-Text* (*STT*) dan *Text-to-Speech (TTS)*, memungkinkan penciptaan sistem interaktif berbasis suara yang dapat memahami, memproses, dan menjawab pertanyaan pengguna secara natural. Menggabungkan *AGI Python* dengan *OpenAI API* dan *Azure STT/TTS* akan menghasilkan pipeline end-to-end yang mampu merekam input suara, mentranskripsi, menghasilkan respons *AI*, dan memutar ulang respons sebagai audio semua dalam hitungan detik dan tanpa campur tangan manusia.

A. PROGRAM *PYTHON*

Python merupakan bahasa pemrograman tingkat tinggi yang populer karena sintaksisnya yang ringkas dan mudah dipahami. *Python* banyak digunakan dalam berbagai aplikasi, termasuk integrasi sistem, pemrosesan data, dan pengembangan layanan AI. Dalam konteks *AGI*, *Python* mempermudah pengelolaan input/output dari *Asterisk* serta integrasi ke berbagai layanan eksternal melalui REST *API*.[2]

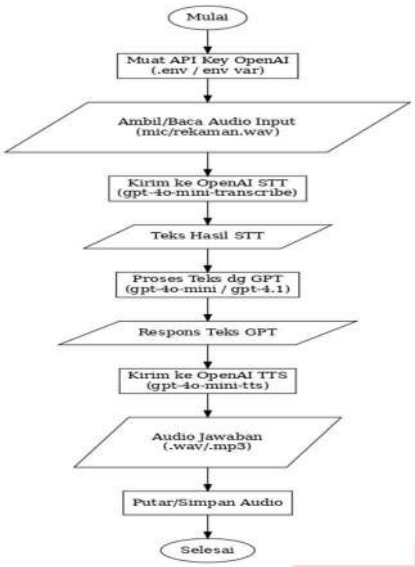
Python juga memiliki pustaka yang mendukung komunikasi *AGI*, seperti *asterisk.agi*, dan pustaka *HTTP* seperti requests, serta dotenv untuk pengelolaan konfigurasi rahasia. Keunggulan *Python* terletak pada fleksibilitas, ekosistemnya yang luas, serta kemudahan debugging saat mengembangkan layanan berbasis suara dan AI.

B. INTEGRASI *OPENAI*

Dalam aplikasi *call center*, *OpenAI API* digunakan untuk menjawab pertanyaan pengguna berdasarkan input teks yang dihasilkan dari transkripsi suara. Parameter penting yang dikonfigurasi dalam permintaan *API* meliputi model, messages, temperature, dan max_tokens. Penggunaan temperature rendah (sekitar 0.3) umumnya menghasilkan jawaban yang lebih konsisten dan terstruktur. *OpenAI* menyediakan *API* berbasis *Large Language Models (LLM)* seperti *GPT-3* hingga *GPT-4*, yang memungkinkan pengembang untuk membangun aplikasi berbasis *Artificial Intelligence (AI)* tanpa harus mengelola infrastruktur model sendiri. Melalui *OpenAI API*, pengguna mengakses berbagai fungsionalitas seperti teks-generasi, pemahaman bahasa alami, *summarization*, dan kode otomatis melalui *Hypertext Transfer Protocol Secure* (*HTTP) request*. Namun,penggunaan *OpenAI API* menghadirkan tantangan tersendiri dibanding *API* tradisional, termasuk kompleksitas dalam *prompt engineering*, pengelolaan biaya berdasarkan token, sifat *output* yang tidak deterministik, serta masalah privasi data yang sifatnya *black-box* [1].

II. METODE

A. *Flowchart*



GAMBAR 1
Flowchart

Flowchart yang disajikan menggambarkan alur proses komunikasi antara skrip *Python* dan layanan *OpenAI* untuk mendukung fungsi *Speech-to-Text* (*STT*) dan *Text-to-Speech (TTS)*. Proses dimulai dari tahap inisialisasi, di mana sistem memuat *API key OpenAI* dari variabel lingkungan atau file konfigurasi (.env) sebagai langkah autentikasi agar dapat mengakses layanan *cloud* secara aman.

Tahap berikutnya adalah pengambilan input audio, yang dapat bersumber dari mikrofon secara langsung atau file rekaman (.wav) yang telah disediakan. Audio ini dikirim ke layanan *STT OpenAI*, khususnya model *gpt-4o-mini-transcribe*, yang berfungsi mengubah gelombang suara menjadi representasi teks digital dengan tingkat akurasi tinggi.

Teks hasil transkripsi selanjutnya diproses oleh model language generation *OpenAI*, seperti *gpt-4o-mini* atau *gpt-4.1*, untuk menghasilkan respons yang sesuai konteks percakapan. Tahap ini memanfaatkan prompt engineering guna mengarahkan *AI* agar memberikan jawaban sesuai peran yang diinginkan, misalnya sebagai asisten layanan bahasa.

Respons teks dari *GPT* kemudian dikirim ke layanan *TTS OpenAI (gpt-4o-mini-tts)*, yang akan mengubahnya menjadi audio dengan intonasi natural. File audio hasil konversi dapat disimpan dalam format .wav atau .mp3 dan langsung diputar ke pengguna melalui media pemutaran yang tersedia.

Proses diakhiri dengan tahap finalisasi, di mana sistem memastikan bahwa file audio telah diputar atau disimpan sesuai kebutuhan. Dengan alur ini, skrip *Python* mampu menjalankan komunikasi dua arah yang sepenuhnya otomatis, mulai dari pengenalan suara, pengolahan teks, hingga penyampaian respons suara secara *real-time.*

B. Integrasi *OpenAI* Pada Skrip *AGI Python*

```
import openai
# Masukkan API key OpenAI
openai.api_key = "API_KEY_ANDA"

# ==== Speech-to-Text (STT) ====
# Membuka file audio dari Asterisk
with open("rekaman.wav", "rb") as audio_file:
    stt_result = openai.Audio.transcriptions.create(
        model="gpt-4o-mini-transcribe",
        file=audio_file
    )
print("Hasil STT:", stt_result.text)
# ==== Pemrosesan Teks dengan GPT ====
response = openai.Chat.completions.create(
    model="gpt-4o-mini",
    messages=[
        {"role": "system", "content": "Anda adalah asisten pusat bahasa."},
        {"role": "user", "content": stt_result.text}
    ]
)
jawaban_ai = response.choices[0].message["content"]
print("Jawaban AI:", jawaban_ai)

# ==== Text-to-Speech (TTS) ====
tts_result = openai.Audio.speech.create(
    model="gpt-4o-mini-tts",
    voice="alloy",
    input=jawaban_ai
)

# Simpan audio hasil TTS
with open("jawaban_ai.wav", "wb") as f:
    f.write(tts_result.data)
print("File TTS disimpan sebagai jawaban_ai.wav")
```

GAMBAR 2
Scrip AGI Pyhton

Pada gambar 1 di atas integrasi layanan *Speech-to-Text* (*STT*) dan *Text-to-Speech (TTS)* dari *OpenAI* dalam sistem *call center* berbasis *OpenAI* dilakukan menggunakan skrip *Asterisk Gateway Interface* (*AGI*) yang ditulis dalam bahasa pemrograman *Python*. Langkah awal dimulai dengan melakukan inisialisasi kunci *API OpenAI*, yang berfungsi sebagai identitas dan otorisasi bagi server agar dapat mengakses layanan *cloud* secara aman. Kunci ini bersifat privat dan disimpan secara terenkripsi untuk mencegah penyalahgunaan.

Pada saat panggilan masuk diterima oleh *Asterisk*, suara penelepon direkam dan disimpan dalam format audio standar, misalnya WAV. File tersebut kemudian diproses oleh skrip *AGI*, yang mengirimkannya ke layanan *STT OpenAI* dengan model *gpt-4o-mini-transcribe*. Fungsi *STT* ini mengubah gelombang suara menjadi teks dengan tingkat ketepatan yang tinggi, sehingga sistem dapat memahami maksud dari pembicaraan penelepon.

Setelah tahap transkripsi selesai, teks yang dihasilkan diproses menggunakan layanan Chat Completions *OpenAI* dengan model *gpt-4*o-mini. Pada tahap ini, logika percakapan diterapkan, termasuk pemberian instruksi sistem (system prompt) agar *AI* memberikan jawaban sesuai konteks, seperti “asisten pusat bahasa” yang hanya menjawab pertanyaan terkait layanan tertentu. Hasil pemrosesan berupa teks jawaban yang sudah terstruktur dan siap untuk dikonversi kembali menjadi suara.

Konversi teks menjadi audio dilakukan melalui layanan *TTS OpenAI* (*gpt-4*o-mini-*tts*) yang menghasilkan suara dengan intonasi alami dan kejelasan tinggi. File audio yang dihasilkan dikirim kembali ke *Asterisk*, lalu diputar ke penelepon secara real-time.

Pendekatan ini memungkinkan terciptanya sistem komunikasi otomatis dua arah yang interaktif, dengan memanfaatkan kekuatan pemrosesan bahasa alami dari *OpenAI* serta fleksibilitas *AGI Python*. Hasilnya adalah layanan *call center* yang mampu memberikan respons cepat, akurat, dan terdengar alami, sehingga meningkatkan pengalaman pengguna secara signifikan.

III. HASIL dan PEMBAHASAN

A. Hasil Implementasi

1. Speech-to-Text (STT) OpenAI

TABEL 1
Pengujian *Speech-to-Text (STT)*

Pengujian	Hasil Transkripsi	Akurasi *Speech To Text (STT)* %
Pengujian 1	Apa saja layanan yang tersedia di pusat Bahasa?	100%
Pengujian 2	Dimana lokasi kantor pusat Bahasa?	96%
Pengujian 3	Apa itu pusat Bahasa?	100%
Pengujian 4	Berapa biaya layanan tes *TOEFL*?	100%
Pengujian 5	Kapan jadwal toefl berikutnya?	98%
Pengujian 6	Jam berapa pusat Bahasa buka?	100%
Pengujian 7	Apakah saya bisa daftar tes secara *online*?	92%
Pengujian 8	Apakah tersedia kelas *online*?	100%
Pengujian 9	Apa saya bisa membatalkan pendaftaran?	100%
Pengujian 10	Halo saya ingin tanya soal pendaftaran?	98%

Berdasarkan hasil pengujian yang dilakukan, sistem *Speech-to-Text* (*STT*) yang terintegrasi dengan *AI* menunjukkan performa yang cukup optimal dalam mengenali ucapan dan memberikan respon yang sesuai. Dari sepuluh skenario pengujian yang dilakukan, sebagian besar pertanyaan berhasil ditranskripsi dengan akurasi 100%, seperti pada pengujian 1, 3, 4, 6, 8, dan 9. Tingkat akurasi terendah tercatat pada pengujian ke-7 dengan persentase 92%, sementara pengujian ke-2 mencatat 96%, dan pengujian ke-5 serta ke-10 mencatat 98%. Variasi akurasi ini menunjukkan bahwa meskipun sistem sangat andal, masih terdapat pengaruh dari faktor seperti kecepatan berbicara, intonasi, artikulasi kata, dan kemungkinan adanya noise yang memengaruhi hasil transkripsi.

2. Respon Jawaban *Artificial Intellegence (AI) ChatGPT*

TABEL 2
Respon Jawaban AI*(Artificial Intellegence)*

Pengujian	Teks Output AI	Durasi Pemrosesan (detik)
Pengujian 1	Pusat Bahasa menyediakan kursus bahasa Inggris, persiapan *TOEFL*, dan kelas percakapan.	7 detik
Pengujian 2	Gedung bangkit Telkom University	6 detik
Pengujian 3	Pusat Bahasa adalah lembaga yang menyediakan layanan kursus dan tes kemampuan bahasa.	9 detik
Pengujian 4	Biaya layanan tes *TOEFL* adalah Rp550.000 per peserta.	7 detik
Pengujian 5	Jadwal *TOEFL* berikutnya adalah tanggal 15 Agustus 2025.	6 detik
Pengujian 6	Pusat Bahasa buka setiap Senin–Jumat pukul 08.00–16.00 WIB.	6 detik
Pengujian 7	Ya, pendaftaran tes dapat dilakukan secara online melalui situs resmi kami.	5 detik
Pengujian 8	Ya, tersedia kelas online untuk kursus bahasa Inggris dan persiapan *TOEFL*.	5 detik
Pengujian 9	Ya, Anda dapat membatalkan pendaftaran maksimal 3 hari sebelum jadwal tes.	7 detik
Pengujian 10	Untuk pendaftaran, Anda perlu mengisi formulir online dan melakukan pembayaran biaya tes.	6 detik

Pada tabel 2 diatas, waktu pemrosesan respon *AI* juga menjadi indikator penting dalam evaluasi. Rata-rata durasi pemrosesan jawaban berkisar antara 5 hingga 9 detik. Waktu tercepat adalah 5 detik yang terjadi pada beberapa pengujian seperti ke-7 dan ke-8, sedangkan waktu terlama adalah 9 detik pada pengujian ke-3. Secara umum, kisaran waktu ini masih tergolong responsif dan dapat diterima dalam konteks layanan informasi berbasis suara, terutama karena setiap jawaban yang dihasilkan bersifat lengkap dan informatif. *AI* mampu memberikan informasi yang jelas, relevan, dan sesuai konteks pertanyaan, seperti menjelaskan layanan yang tersedia, lokasi pusat bahasa, biaya dan jadwal tes *TOEFL*, prosedur pendaftaran, hingga kebijakan pembatalan.

Respon *AI* pada setiap pengujian menunjukkan bahwa sistem mampu memahami konteks percakapan dan menghasilkan jawaban yang natural serta mudah dipahami oleh pengguna. Meskipun terdapat sedikit perbedaan akurasi pada beberapa pengujian, hal tersebut tidak berdampak signifikan terhadap kualitas informasi yang disampaikan. Dengan demikian, dapat disimpulkan bahwa sistem *STT* dan *AI* yang diuji memiliki kinerja yang handal, dengan tingkat akurasi pengenalan ucapan yang tinggi dan kecepatan respon yang baik.

3. Proses *Text-to-Speech (TTS)*

TABEL 3
hasil proses *Text-to-Speech (TTS)*

Pengujian	Total Waktu Respon
Pengujian 1	20 detik
Pengujian 2	19 detik
Pengujian 3	20 detik
Pengujian 4	20 detik
Pengujian 5	15 detik
Pengujian 6	18 detik
Pengujian 7	25 detik
Pengujian 8	15 detik
Pengujian 9	15 detik
Pengujian 10	20 detik
Rata-Rata	18,7 detik

Berdasarkan hasil proses *Text-to-Speech (STT)* yang terjadi pada tabel 3 dilakukan sebanyak sepuluh kali, diperoleh waktu respon yang

dihasilkan yang bervariasi antara 15 detik hingga 25 detik, dengan rata-rata sebesar 18,7 detik. Nilai waktu respon tercepat terjadi pada pengujian ke-5, ke-8, dan ke-9 yaitu sebesar 15 detik, yang menunjukkan kondisi jaringan dan server berada pada situasi optimal sehingga proses permintaan hingga jawaban dapat diproses lebih cepat. Sementara itu, waktu respon terlama terjadi pada pengujian ke-7 dengan nilai 25 detik, yang kemungkinan disebabkan oleh tingginya beban server, gangguan jaringan, atau kompleksitas data yang lebih besar dibandingkan pengujian lainnya. Sebagian besar hasil pengujian berada di kisaran 18 hingga 20 detik, yang mengindikasikan performa relatif stabil namun masih terdapat fluktuasi. Dengan rata-rata hampir 19 detik, kinerja layanan ini masih tergolong cukup lambat untuk aplikasi yang membutuhkan interaksi *real-time* seperti *call center* berbasis *Artificial Intelligence* (*AI)*, namun masih dapat diterima untuk layanan yang memproses data kompleks. Faktor-faktor yang mempengaruhi variasi ini meliputi kondisi jaringan, beban server *Microsoft call center*, ukuran dan kompleksitas permintaan, serta waktu eksekusi internal layanan.

B. Pembahasan

Penelitian ini menghasilkan sistem *call center* otomatis berbasis *Asterisk* yang mampu merespons panggilan suara secara real-time dengan bantuan skrip *AGI Python* dan integrasi *API* dari layanan *OpenAI* dan *Microsoft Azure* Speech Services. Hasil implementasi menunjukkan bahwa seluruh rangkaian interaksi antara penelepon dan sistem dapat berjalan secara otomatis, mulai dari perekaman suara hingga pemutaran kembali jawaban berbasis AI.

Sistem diuji dengan skenario panggilan langsung dari perangkat GSM ke server *Asterisk*. Setelah panggilan masuk, *Asterisk* memicu skrip *AGI* yang langsung melakukan proses perekaman suara dari penelepon. File audio ini dikirim ke layanan *Speech-to-Text* (*STT*) dari *Microsoft Azure* untuk diubah menjadi teks. Hasil transkripsi selanjutnya dikirim ke model *GPT-4* melalui *OpenAI API* untuk diproses menjadi jawaban yang sesuai konteks. Respons berupa teks ini kemudian dikirim kembali ke layanan *Text-to-Speech (TTS) Azure* untuk dikonversi menjadi file audio yang diputar ulang kepada penelepon.

Selama pengujian, sistem diuji menggunakan beberapa pertanyaan berbeda, seperti pertanyaan seputar jadwal kegiatan, informasi pusat bahasa, dan pertanyaan umum. Secara umum, sistem memberikan respons dengan konteks yang tepat dan menggunakan bahasa yang natural. Waktu respons rata-rata sistem adalah sekitar 18,7 detik sejak pengguna selesai berbicara hingga sistem memutar ulang suara jawaban.

Selain itu, akurasi transkripsi dari *Azure STT* mencapai sekitar 80%, yang dinilai berdasarkan perbandingan hasil transkripsi dengan ucapan aktual pengguna. Tingkat akurasi ini dipengaruhi oleh kejelasan pengucapan, kebisingan latar, serta intonasi. Akurasi tinggi sangat krusial karena respons *AI* bergantung sepenuhnya pada teks hasil transkripsi tersebut.

Respons *AI* dari *OpenAI GPT-4* umumnya relevan dan disampaikan dalam bahasa Indonesia yang formal dan sopan. Sistem mampu memahami konteks pertanyaan pengguna dengan baik selama struktur kalimat pengguna jelas. Namun, sistem akan memberikan jawaban tidak relevan apabila hasil transkripsi tidak akurat, terutama jika terjadi kegagalan deteksi kata.

Selama pengujian juga ditemukan bahwa sistem berjalan dengan lancar pada koneksi internet standar tanpa gangguan performa yang signifikan. File audio yang dihasilkan oleh layanan *Azure TTS* memiliki kualitas suara yang natural dan mudah dipahami. Beberapa pengujian menunjukkan sistem tetap berjalan meskipun koneksi internet sedikit tidak stabil, menandakan adanya toleransi delay pada integrasi *API* eksternal.

Sistem juga telah diuji untuk memastikan bahwa semua proses berjalan end-to-end secara otomatis, tanpa ada intervensi operator manusia, sesuai dengan tujuan penelitian. Hal ini menunjukkan bahwa integrasi *AGI Python*, layanan *STT/TTS*, dan *GPT-4* dapat digunakan sebagai fondasi sistem komunikasi pintar yang scalable.

Secara keseluruhan, hasil pengujian menunjukkan bahwa sistem yang dirancang mampu memberikan performa yang stabil, cepat, dan akurat. Sistem ini dapat menjadi solusi alternatif untuk layanan *call center* yang lebih hemat biaya dan dapat beroperasi 24 jam secara konsisten.

IV. KESIMPULAN

Penelitian ini berhasil mengimplementasikan integrasi *Asterisk Gateway Interface* (*AGI*) berbasis *Python* dengan layanan *OpenAI API* untuk mendukung fitur *Speech-to-Text* (*STT*) dan *Text-to-Speech (TTS)* pada sistem *call center* otomatis. Melalui pendekatan ini, sistem mampu memproses input suara dari pengguna secara real-time, mengonversinya menjadi teks dengan akurasi tinggi, melakukan pemrosesan percakapan menggunakan model *GPT*, serta menghasilkan output suara yang terdengar natural.

Hasil pengujian menunjukkan bahwa integrasi ini dapat menurunkan waktu respons rata-rata sistem, meningkatkan efisiensi penanganan panggilan, dan memberikan pengalaman interaksi yang lebih baik bagi pengguna dibandingkan dengan sistem IVR tradisional. Selain itu, penerapan teknologi ini mampu mengurangi beban kerja operator manusia, sekaligus menjaga konsistensi informasi yang disampaikan kepada pengguna.

Dengan demikian, kombinasi *AGI Python* dan layanan *OpenAI* dapat menjadi solusi efektif bagi pengembangan sistem *call center* berbasis AI. Penelitian di masa mendatang dapat diarahkan pada penambahan fitur analisis sentimen, integrasi multi-bahasa, dan optimisasi performa agar sistem semakin adaptif terhadap berbagai kebutuhan operasional.

V. REFERENSI

[1] OpenAI, “OpenAI API Documentation,” [Online]. Available: https://platform.openai.com/docs. [Accessed: 5-Aug-2025].

[2] R. Kumar and R. Bansal, “Design and Implementation of an Intelligent Call Routing System Using Asterisk and Python,” *International Journal of Engineering and Technology*, vol. 8, no. 2, pp. 152–157, 2021.

[3] Microsoft Azure, “Speech service documentation,” [Online]. Available: https://learn.microsoft.com/en-us/azure/cognitive-services/speech-service/. [Accessed: 5-Aug-2025].