INTEGRASI PUPPETEER DAN WHISPER OPEN AI UNTUK PENGEMBANGAN BOT NOTULA PADA PLATFORM GOOGLE MEET

Disusun Oleh:

N a m a : Muhammad Ihsan Syafiul Umam
NIM : 20523151

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
2024

HALAMAN PENGESAHAN DOSEN PEMBIMBING

INTEGRASI PUPPETEER DAN WHISPER OPEN AI UNTUK PENGEMBANGAN BOT NOTULA PADA PLATFORM GOOGLE MEET

TUGAS AKHIR JALUR MAGANG

Disusun Oleh:

N a m a : Muhammad Ihsan Syafiul Umam
NIM 20523217

Yogyak ta 1 uli 2024
Pe nb bi g,

(M oh d ris S.Kom. M.Kom)

HALAMAN PENGESAHAN DOSEN PENGUJI

INTEGRASI PUPPETEER DAN WHISPER OPEN AI UNTUK PENGEMBANGAN BOT NOTULA PADA PLATFORM GOOGLE MEET

TUGAS AKHIR JALUR MAGANG

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia
Yogyakarta, Juli 2024

Tim Penguji

Moh. Idris, S.Kom., M.Kom. _____

Kholid Haryono., S.T., M.Kom. _____

Irving Vitra Paputungan., S.T., M.Sc. _____

Mengetahui,
Ketua Program Studi Informatika – Program Sarjana
Fakultas Teknologi Industri
Universitas Islam Indonesia

(Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama	:	Muhammad Ihsan Syafiul Umam
NIM	:	20523217

Tugas akhir dengan judul:

INTEGRASI PUPPETEER DAN WHISPER OPEN AI UNTUK PENGEMBANGAN BOT NOTULA PADA PLATFORM GOOGLE MEET

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir *ini adalah hasil karya* saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 4 Juli 2024

Muhammad Ihsan Syafiul Umam

HALAMAN PERSEMBAHAN

Puji syukur penulis panjatkan kepada Allah SWT atas segala nikmat yang telah diberikan dalam perjalanan menyelesaikan tugas akhir ini yang berjudul “Integrasi Puppeteer dan Whisper Open AI Untuk Pengembangan Bot Notula Pada Platform Google Meet”. Penulis juga mengucapkan terima kasih yang sebesar-besarnya kepada orang tua, keluarga, dan teman dekat yang selalu memberikan dukungan, semangat, dan doa dalam perjalanan penulisan laporan tugas akhir ini. Dukungan dan kasih sayang yang diberikan menjadi pendorong utama dalam menghadapi berbagai tantangan selama proses penulisan. Semoga laporan tugas akhir ini dapat memberikan manfaat dan kontribusi yang positif bagi pengembangan teknologi.

HALAMAN MOTO

“Dan, carilah pada apa yang telah dianugerahkan Allah kepadamu (pahala) negeri akhirat, tetapi janganlah kamu lupakan bagianmu di dunia. Berbuat baiklah (kepada orang lain) sebagaimana Allah telah berbuat baik kepadamu dan janganlah kamu berbuat kerusakan di bumi. Sesungguhnya Allah tidak menyukai orang-orang yang berbuat kerusakan.”

-QS. Al-Qasas: 77

KATA PENGANTAR

Puji syukur atas kehadirat Allah SWT yang telah memberikan kesempatan kepada sehingga penulis dapat menyelesaikan laporan tengah magang di Widya Robotics dengan baik. Dalam menulis laporan ini, penulis secara terbuka menerima masukan, bimbingan maupun dukungan. Tujuan disusunya laporan magang ini adalah bukti bahwa penulis telah melaksanakan magang dan menyelesaikan kegiatan magang di Widya Robotics. Penulis mengucapkan terima kasih kepada seluruh pihak yang mendukung penulis selama proses pembuatan laporan ini.

Pihak-pihak yang terkait diantaranya sebagai berikut:

1. Kepada Allah SWT yang telah memberikan nikmat dan karunia Nya untuk dapat melaksanakan magang dengan baik.
2. Kedua orang tua yang selalu memberikan dukungan berupa doa, dukungan, dan materi.
3. Bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc. selaku Ketua Jurusan Informatika dan Bapak Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D. selaku Ketua Program Studi Informatika – Program Sarjana.
4. Bapak Moh. Idris, S.Kom., M.Kom., selaku dosen pembimbing yang telah memberikan arahan dan masukan kepada penulis dalam menyusun laporan ini.
5. Bapak Norma Dani Risdiandita, selaku VP Engineering yang telah memberikan pembelajaran bermanfaat selama melaksanakan magang di Widya Robotics.
6. Seluruh karyawan dan karyawati Widya Robotics yang telah menerima dan mendukung penulis selama melaksanakan magang.

Sekali lagi penulis mengucapkan terima kasih kepada semua pihak yang telah memberikan dukunganNya. Jika terdapat kekurangan pada laporan ini penulis memohon maaf atas kesalahan tersebut dan menjadi pelajaran bagi penulis untuk melakukan perbaikan kedepannya.

Yogyakarta, 7 Juli 2024

Muhammad Ihsan Syafiul Umam

SARI

Keterbatasan waktu dan tingkat kesalahan manual sering kali mengakibatkan ketidakakuratan dan ketidakefisienan dalam pencatatan notula rapat secara daring. Penelitian ini mengembangkan bot notula otomatis untuk Google Meet menggunakan teknologi Puppeteer dan Whisper. Pengguna hanya perlu memberikan URL rapat Google Meet, lalu bot secara otomatis bergabung dalam rapat, merekam audio, dan mentranskripsikan menjadi teks. Mengintegrasikan Whisper AI untuk transkripsi dan teknik diarization tambahan, bot ini dapat mengidentifikasi berbagai pembicara dengan akurat. Dikembangkan dengan metodologi prototyping untuk perbaikan berkelanjutan, sistem ini terdiri dari dua layanan terpisah untuk skalabilitas. Pengujian awal menunjukkan hasil yang menjanjikan, mengindikasikan potensi bot untuk memberikan transkripsi berkualitas tinggi dalam berbagai kondisi. Secara signifikan, penggunaan bot ini dapat meningkatkan otomatisasi dokumentasi rapat, mengurangi kesalahan manusia, dan meningkatkan produktivitas. Proyek ini dikerjakan sebagai bagian dari tugas magang di Widya Robotics.

Kata kunci: Bot Notula, Whisper, Speaker Diarization, Google Meet, Puppeteer

GLOSARIUM

Service	Mengacu pada komponen perangkat lunak yang dirancang untuk melakukan fungsi spesifik dalam suatu sistem.
Framework	Struktur konseptual atau kerangka kerja yang menyediakan pendekatan umum untuk membangun dan mengembangkan aplikasi atau sistem perangkat lunak.
Scallable	Kemampuan suatu sistem untuk menangani peningkatan beban kerja atau volume data tanpa mengalami penurunan kinerja yang signifikan atau kegagalan sistem.
Cloud Storage	Layanan penyimpanan data yang disediakan melalui internet, memungkinkan pengguna untuk menyimpan dan mengakses *file* secara fleksibel dari berbagai Lokasi.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pada era digital yang berkembang pesat saat ini, otomatisasi menjadi salah satu kebutuhan utama dalam berbagai sektor bisnis. Otomatisasi tidak hanya membantu meningkatkan efisiensi operasional tetapi juga mengurangi beban kerja manual sehingga memungkinkan karyawan untuk fokus pada tugas-tugas yang lebih strategis. Salah satu area yang mendapat perhatian khusus adalah otomatisasi proses pencatatan notula dalam pertemuan atau rapat, terutama yang dilakukan secara daring melalui platform seperti Google Meet.

Widya Robotics, perusahaan teknologi yang merupakan bagian dari Widya Indonesia Group, telah menunjukkan dedikasinya dalam menghadirkan solusi teknologi berbasis data dan robotika. Sebagai perusahaan pertama yang lahir di dalam grup tersebut, Widya Robotics menawarkan rangkaian produk dan layanan inovatif yang didukung oleh kecerdasan buatan, otomatisasi, dan teknologi robotika. Produk-produk ini dirancang untuk membantu klien mencapai tujuan bisnis mereka melalui solusi yang efisien dan efektif.

Widya Robotics telah berhasil mengembangkan beragam teknologi canggih, salah satunya adalah teknologi Vision Intelligence (VI). Teknologi ini merupakan kecerdasan buatan yang mampu mengenali berbagai objek melalui penggunaan kamera. Dengan pemanfaatan teknologi ini, Widya Robotics berhasil menciptakan beberapa produk unggulan seperti aplikasi presensi dengan pengenalan wajah, aplikasi penghitung objek, dan sistem pengawasan kerja karyawan. Di samping itu, Widya Robotics juga merancang perangkat keras dan aplikasi khusus untuk kebutuhan industri, seperti Widya Load Scanner yang dapat mengukur *volume* muatan truk dengan lebih efisien. Mereka juga menciptakan Gas Monitoring, sebuah perangkat Internet of Things (IoT) yang mampu mendeteksi kadar berbagai jenis gas pada proyek terowongan setelah proses peledakan berlangsung.

Salah satu kontribusi penting dalam inovasi di Widya Robotics adalah pengembangan bot notula meeting di Google Meet. Bot ini dikembangkan untuk menjawab kebutuhan otomatisasi dalam pencatatan notula selama rapat. Dalam pengembangan bot ini, teknologi Puppeteer digunakan untuk mengotomatisasi interaksi dengan antarmuka Google Meet. Dengan adanya bot notula, pengguna tidak perlu lagi mencatat pokok-pokok penting dalam rapat secara manual dan membuat ringkasannya. Hal ini tidak hanya menghemat waktu tetapi juga memastikan akurasi dan konsistensi dalam pencatatan.

Puppeteer digunakan karena kemampuan untuk secara otomatis mensimulasikan interaksi antara browser dan halaman web. Salah satu metode pengambilan data pada suatu halaman web secara otomatis adalah web scraping, yang memudahkan proses simulasi masuknya bot ke Google Meet, melakukan perekaman audio pertemuan, dan melakukan *scraping* data *subtitle* bawaan Google Meet. Sebenarnya, Puppeteer adalah *library* Nodejs yang memiliki kemampuan untuk mengotomatisasi banyak hal yang biasa dilakukan secara manual di *browser*, seperti pengambilan data, interaksi dengan halaman dengan klik tombol, *screenshoot* halaman, dan masih banyak lagi (Srivastava, 2021). Karena itu, Puppeteer adalah alat yang sangat penting untuk membuat bot notula Google Meet. Selain itu Puppeteer juga biasa digunakan oleh seorang Quality Assurance untuk melakukan *automated testing* pada suatu aplikasi web, hal ini ditujukan untuk mengefisiensi waktu dan menghindari kesalahan berulang dalam proses *testing* (Nurkholis & Yunial, 2023).

Selain itu, penelitian ini juga melibatkan proses *diarization* menggunakan Whisper dari OpenAI. *Diarization* adalah proses memisahkan dan mengidentifikasi pembicara yang berbeda dalam rekaman audio (Wang & Purushotam, 2023). Dengan menggunakan Whisper, bot notula dapat secara akurat mengenali dan memisahkan suara dari berbagai peserta meeting, sehingga menghasilkan catatan yang lebih terstruktur dan informatif.

Selama magang di Widya Robotics, penulis berkesempatan untuk terlibat dalam beberapa proyek, antara lain Task Remainder Bot Discord, aplikasi penghitungan elektabilitas calon legislatif, Bot Notula Google Meet, dan Load Scanner Dashboard. Dari berbagai proyek tersebut, penulis memilih untuk fokus pada proyek Bot Notula Google Meet dalam tugas akhir ini. Pilihan ini didasarkan pada kontribusi penulis yang lebih dominan pada proyek ini, serta penggunaan berbagai teknologi baru yang belum pernah penulis gunakan sebelumnya, seperti teknologi audio processing, Puppeteer, dan RabbitMQ.

Laporan ini merangkum seluruh aktivitas yang berlangsung selama masa magang di Widya Robotics dan merupakan bagian dari laporan akhir untuk menyelesaikan program magang pada semester pertama tahun akademik 2023/2024.

1.2 Ruang Lingkup

Pengembangan bot notula Google Meet dilaksanakan selama tiga bulan dimulai dari September 2023 hingga November 2023. Proyek ini dikerjakan oleh tiga orang yang terdiri dari dua orang karyawan magang sebagai *backend engineer* termasuk penulis dan satu orang

supervisor. Kontribusi penulis akan mencakup beberapa aspek penting dalam pengembangan sistem sebagai berikut:

a. Riset arsitektur protokol komunikasi pada Google Meet.
b. Riset dan implementasi puppeteer pada proyek bot notula Google Meet.
c. Integrasi Google Object Storage untuk penyimpanan hasil *file* audio meeting.
d. Integrasi *service* Bot Notula Engine dan *service* Audio Processing menggunakan RabbitMQ.
e. Integrasi Whisper Open AI untuk melakukan *multi-speaker diarization*.

1.3 Tujuan

Tujuan dari pengembangan bot notula pada platform Google Meet adalah untuk mencapai efisiensi dan akurasi dalam pencatatan rapat. Bot notula ini dirancang untuk mengotomatisasi proses pencatatan rapat, memungkinkan bot untuk memasuki dan berinteraksi dengan Google Meet tanpa memerlukan intervensi manual. Integrasi dengan Puppeteer memungkinkan bot untuk mengikuti rapat secara konsisten dan efisien. Selain itu, dengan memanfaatkan teknologi *diarization* dari Whisper OpenAI, bot notula mampu memisahkan dan mengenali berbagai pembicara dalam rapat. Hal ini bertujuan untuk menghasilkan transkripsi yang lebih akurat dan mengurangi kesalahan yang sering terjadi dalam pencatatan manual.

Bot notula ini dirancang untuk bekerja secara real-time, menyediakan notula yang segera tersedia setelah rapat selesai. Hal ini memungkinkan pengambilan keputusan yang lebih cepat dan meningkatkan produktivitas tim, karena hasil rapat dapat langsung diakses dan digunakan. Lebih lanjut, bot notula Google Meet merupakan salah satu fitur yang nantinya akan diintegrasikan ke dalam aplikasi Widya Notulensi, sebuah aplikasi pencatatan rapat otomatis yang mendukung berbagai platform meeting online seperti Zoom, Google Meet, dan Microsoft Teams. Dengan integrasi ini, diharapkan aplikasi Widya Notulensi dapat memberikan solusi pencatatan rapat yang komprehensif dan andal untuk berbagai kebutuhan organisasi.

1.4 Manfaat

Manfaat dari integrasi Whisper OpenAI dan Puppeteer dalam pengembangan bot notula pada platform Google Meet adalah sebagai berikut:

a. Otomatisasi Pencatatan Rapat: Integrasi Puppeteer memungkinkan otomatisasi dalam memasuki dan berinteraksi dengan Google Meet, sehingga bot dapat secara efisien mengikuti rapat tanpa memerlukan campur tangan manual.

b. Akurasi dan Kualitas Transkripsi: Whisper OpenAI menyediakan teknologi *diarization* yang mampu memisahkan dan mengenali berbagai pembicara dalam rapat sehingga menghasilkan transkripsi yang akurat dan mengurangi kesalahan.
c. Peningkatan Produktivitas: Bot notula yang terintegrasi dapat bekerja secara real-time, menghasilkan notula yang segera tersedia setelah rapat selesai. Produktivitas menjadi meningkat karena keputusan dapat segera diambil secepatnya.

1.5 Sistematika Penulisan

Sistematika penulisan disusun untuk mempermudah pembaca dalam memahami isi laporan secara menyeluruh. Berikut adalah susunan sistematika penulisan laporan:

a. BAB I: Pendahuluan

   Bab ini membahas tentang latar belakang, ruang lingkung, tujuan, manfaat, dan sistematika penulisan.
b. BAB II: Landasan Teori dan Tinjauan Pustaka

   Bab ini membahas tentang teori – teori yang menjadi dasar integrasi Whisper Open AI dan Puppeteer dalam pengembangan bot notula pada Platform Google Meet.
c. BAB III: Pelaksanaan Magang

   Bab ini membahas tentang kegiatan yang berisi implementasi dari proses kegiatan magang.
d. BAB IV: Refleksi Pelaksanaan Magang

   Bab ini membahas tentang refleksi dan hasil yang diperoleh selama masa magang pada proyek pengembangan Bot Notula Google Meet.
e. BAB V: Penutup

   Bab ini membahas tentang kesimpulan dan saran dari materi yang telah dibahas pada bab sebelumnya.

BAB II
LANDASAN TEORI DAN TINJAUAN PUSTAKA

2.1 Bot Notula

Bot Notula terdiri dari dua kata, yaitu "bot" dan "notula." "Bot" diartikan sebagai aplikasi perangkat lunak yang telah diberi instruksi sebelumnya melalui kode program untuk melakukan tugas secara otomatis dan repetitif (Setiawan, 2021). Sementara itu, "notula" merujuk pada catatan singkat yang dibuat berdasarkan sebuah rapat atau pertemuan. Notula berisi kegiatan yang dilakukan, peserta yang hadir dalam rapat, serta kesimpulan pembicaraan yang diambil dari masing-masing peserta rapat (KBBI VI Daring, n.d.). Sehinga dapat disimpulkan bahwa bot notula adalah sebuah perangkat lunak yang dapat melakukan tugas pencatatan informasi rapat meliputi isi percakapan setiap peserta, peserta yang hadir dalam rapat, serta kesimpulan yang didapatkan secara otomatis.

Dalam melakukan pencatatan sebuah rapat atau pertemuan, pada umumnya dilakukan dengan menulis detail-detail penting secara manual, baik menggunakan perangkat lunak seperti aplikasi catatan atau di atas kertas. Pendekatan ini sering kali menyebabkan ketidakefisienan waktu dan sumber daya, karena pencatatan manual dapat mengalihkan perhatian peserta rapat dari diskusi yang sedang berlangsung. Oleh karena itu, penggunaan bot untuk pencatatan rapat menjadi solusi yang sangat diperlukan.

Penggunaan bot dalam pencatatan rapat menawarkan efisiensi waktu dan sumber daya yang signifikan. Bot dapat melakukan pencatatan secara real-time, memungkinkan peserta rapat untuk fokus sepenuhnya pada diskusi tanpa terganggu oleh keharusan mencatat. Selain itu, bot dapat meminimalkan kesalahan manusia dalam pencatatan, seperti kesalahan penulisan atau kehilangan informasi penting, dengan memastikan bahwa semua poin penting dicatat dengan akurat dan konsisten.

Bot juga memiliki kemampuan multitasking yang tinggi, sehingga dapat menangani berbagai tugas secara simultan, seperti merekam percakapan, mencatat kehadiran, dan bahkan mengirim ringkasan rapat kepada peserta setelah rapat selesai. Hal ini secara keseluruhan meningkatkan produktivitas rapat. Data yang dikumpulkan oleh bot dapat dianalisis untuk mendapatkan wawasan lebih dalam mengenai pola diskusi, frekuensi topik tertentu, dan kontribusi individu. Dengan pembelajaran mesin, bot dapat meningkatkan kemampuannya untuk mengidentifikasi dan merangkum poin-poin penting dari rapat secara lebih efektif.

Selain itu, catatan rapat yang dihasilkan oleh bot dapat disimpan secara digital, memudahkan akses dan pencarian di masa depan. Ini memastikan bahwa semua informasi yang dicatat mudah diakses oleh peserta rapat atau pihak berkepentingan lainnya. Dalam beberapa industri, pencatatan rapat yang akurat dan dapat diaudit merupakan keharusan untuk kepatuhan terhadap regulasi. Bot dapat menghasilkan *log* yang terstruktur dan dapat diverifikasi, yang penting untuk kepatuhan dan audit di masa mendatang.

Dengan adanya catatan rapat yang cepat dan akurat, tim dapat lebih mudah merujuk kembali pada diskusi sebelumnya dan memastikan bahwa semua anggota tim memiliki pemahaman yang sama mengenai keputusan yang telah diambil. Hal ini meningkatkan kolaborasi dan mengurangi kesalahpahaman. Oleh karena itu, dengan mempertimbangkan faktor-faktor ini, penggunaan bot untuk pencatatan rapat dapat secara signifikan meningkatkan efisiensi, akurasi, dan produktivitas dalam lingkungan kerja.

2.2 Google Meet

Google Meet adalah layanan konferensi video yang dikembangkan oleh Google, merupakan bagian dari Google Workspace. Layanan ini memungkinkan pengguna untuk melakukan panggilan video dan rapat virtual dengan kualitas tinggi. Fitur utama Google Meet meliputi berbagi layar, integrasi dengan Google Calendar, dan pembuatan tautan rapat yang mudah diakses tanpa perlu mengunduh perangkat lunak tambahan. Karena penggunaan nya yang mudah dan simple, Google meet cukup banyak digunakan baik di dunia kerja maupun pendidikan seperti kegitatan rapat internal perusahaan atau kegiatan belajar mengajar di institusi pendidikan(Alturki & Aldraiweesh, 2022).

2.3 Puppeteer

Puppeteer adalah pustaka untuk Node.js yang memungkinkan penggunanya mengendalikan *browser* Chrome atau Chromium melalui DevTools Protocol. Dengan Puppeteer, dapat dilakukan berbagai operasi pada browser secara otomatis, seperti membuka halaman web, mengisi formulir, dan mengambil *screenshot*. Secara default, Puppeteer berjalan dalam mode "headless", yang berarti *browser* berjalan tanpa tampilan grafis. Namun, jika diperlukan, Puppeteer dapat dikonfigurasi untuk berjalan dalam mode "headful", di mana browser akan ditampilkan sepenuhnya seperti biasa (Puppeteer n.d.). Hal ini sangat berguna untuk keperluan pengujian, pengambilan data, dan otomasi tugas-tugas web lainnya. Puppeteer juga dapat digunakan untuk melakukan *testing* pada aplikasi berbasis browser secara otomatis (Nurkholis & Yunial, 2023).

Dengan menggunakan Puppeteer, proses masuk ke dalam rapat Google Meet dapat disimulasikan sebagai bot, sehingga memungkinkan pelaksanaan tugas-tugas otomasi yang berulang dan sistematis. Dalam konteks pencatatan rapat pada Goole Meet, Puppeteer dapat digunakan untuk mengotomatisasi berbagai langkah yang biasanya dilakukan secara manual. Bot yang diotomatisasi dengan Puppeteer dapat melakukan proses perekaman audio dari rapat, yang sangat penting untuk proses *audio diarization* pada langkah selanjutknya. Selain itu, bot ini juga dapat mengambil *live subtitle* yang disediakan secara bawaan oleh Google Meet, yang berguna untuk menghasilkan transkrip otomatis dari percakapan dalam rapat.

2.4 Whisper

Whisper adalah model pengenalan suara otomatis (ASR) yang dikembangkan oleh OpenAI sebagai terobosan signifikan dalam teknologi pengenalan suara (Amorese et al., 2023). Model ini menggabungkan jaringan saraf konvolusi (CNN) dan arsitektur Transformer. CNN digunakan untuk mengekstrak fitur spasial dari sinyal audio, seperti frekuensi dan amplitudo, pada berbagai tingkat hierarki. Proses ini membantu dalam menangkap fitur audio penting sebelum meneruskannya ke Transformer. Arsitektur Transformer, yang dikenal dengan mekanisme *self-attention*-nya, sangat baik dalam memahami ketergantungan jarak jauh dalam ucapan, memungkinkan model untuk mengenali pola temporal yang kompleks dan hubungan kontekstual antara kata-kata dalam kalimat dan wacana yang lebih luas. Selain itu model ini juga menggunakan sejumlah besar data yang *weakly supervised* dari berbagai bahasa, mencakup berbagai aksen dan intonasi. Data pelatihan yang digunakan terdiri dari korpus audio yang besar dan beragam, yang dikumpulkan dari berbagai sumber di web. Data ini termasuk transkripsi otomatis yang diperoleh dari *file* subtitle yang menyertai audio, yang difilter menggunakan heuristik otomatis untuk memastikan kualitas tanpa tinjauan manusia secara manual, sehingga disebut *weakly supervised learning*. Pelatihan model ini pada data audio multibahasa dengan beragam aksen dan intonasi meningkatkan kemampuan generalisasi yang kuat, memungkinkan pengenalan ucapan yang efisien di berbagai bahasa dan lingkungan dengan banyak pembicara. (Lyu et al., 2024).

Whisper adalah model terbaru dalam rangkaian model Convolutional Transformer End-to-End ASR, yang memiliki struktur mirip dengan model Wav2Vec2 dan WavLM. Keunikan Whisper terletak pada skala pelatihan modelnya. Whisper dilatih menggunakan 680.000 jam data audio multibahasa yang disertai dengan transkripsi, yang dikumpulkan dari internet. Transkripsi *ground truth* untuk data pelatihan diambil dari *file subtitle* yang menyertai audio tersebut dan difilter menggunakan berbagai heuristik untuk memastikan kualitasnya, sambil

tetap memperhatikan skala dataset yang besar. Pendekatan ini dikenal sebagai pengawasan lemah (*weakly supervised*), karena setiap transkripsi ground truth tidak diulas oleh manusia tetapi difilter secara otomatis (Amorese et al., 2023).

2.5 Speaker Diarization

Speaker Diarization adalah teknik pemrosesan suara yang bertujuan untuk mengidentifikasi *speaker* dalam audio dan menentukan kapan masing-masing *speaker* berbicara (Park et al., 2022). Proses ini melibatkan analisis karakteristik suara individu, segmentasi sinyal audio asli, dan klasifikasi berdasarkan fitur unik dari setiap orang yang berbicara. *Speaker Diarization* tidak hanya membedakan antara berbagai pembicara tanpa memerlukan informasi sebelumnya tentang mereka, tetapi juga dapat diterapkan dalam berbagai konteks dunia nyata, seperti siaran media, percakapan konferensi, media sosial online, dan proses pengadilan. *Diarization* juga membantu meningkatkan akurasi pengenalan suara dalam pengaturan dengan banyak pembicara (Wang & Purushotam, 2023).

Salah satu pustaka yang cukup populer untuk melakukan *speaker diarization* adalah pyannote.audio yang menggunakan bahasa pemrograman python berbasis *framework* PyTorch. Pyannote.audio dipublikasikan di platform Hugging Face sehingga diharuskan untuk mencantumkan *access token* yang dapat ditemukan di laman Hugging Face. Berikut adalah contoh penggunaan payannote.audio untuk melakukan *diarization*:

```
from pyannote.audio import Pipeline
pipeline = Pipeline.from_pretrained(
    "pyannote/speaker-diarization-3.1",
    use_auth_token="HUGGINGFACE_ACCESS_TOKEN_GOES_HERE")

# send pipeline to GPU (when available)
import torch
pipeline.to(torch.device("cuda"))

# apply pretrained pipeline
diarization = pipeline("audio.wav")

# print the result
for turn, _, speaker in diarization.itertracks(yield_label=True):
    print(f"start={turn.start:.1f}s stop={turn.end:.1f}s
speaker_{speaker}")
```

Gambar 2.1 Contoh Penggunaan Pyannote.audio

Kode pada Gambar 2.1 menggunakan pustaka Pyannote.audio untuk melakukan *diarization* pembicara pada berkas audio. *Pipeline* yang telah dilatih sebelumnya diinisialisasi dan diterapkan pada berkas audio.wav, dengan menggunakan GPU untuk mempercepat

pemrosesan jika tersedia. Hasil *diarization* kemudian ditampilkan, menunjukkan waktu mulai dan berhenti setiap segmen pembicara serta identitas pembicara tersebut.

Dengan menggabungkan teknik *automatic speech recognition* dan *speaker diarization* memungkinkan untuk mengenali konten ucapan secara akurat, juga menentukan kapan *speaker* berbeda berbicara, sehingga dapat memecahkan masalah siapa yang mengatakan apa dan kapan. Teknologi ini memiliki aplikasi yang luas, termasuk transkripsi pertemuan dengan banyak peserta, analisis program siaran dengan banyak pembicara, dan pembuatan catatan rapat sederhana hingga pengadilan. (Lyu et al., 2024).

2.6 DOM, Node, dan Mutation Observer dalam JavaScript

DOM (Document Object Model) adalah sebuah cara pada bahasa JavaScript untuk melihat dan mengubah dokumen pada HTML. DOM memiliki sebuah *interface* dan platform yang berfungsi sebagai pemanggilan konten, struktur, dan *style* pada program HTML dan script. Pada dasarnya, DOM adalah sebuah representasi dari dokumen HTML dalam bentuk pohon, di mana setiap *node* merepresentasikan elemen atau bagian dari dokumen. *Node* dapat memiliki anak (*child*) dan atribut (*attribute*), dan dapat diubah menggunakan JavaScript (Arwin Dermawan et al., 2022). Berikut adalah contoh sebuah dokumen html dan bagaimana struktur DOM dari dokumen tersebut pada Gambar 2.2 dan Gambar 2.3.Gambar 2.3

```
<html>
  <head>
    <title>Contoh Dokumen</title>
  </head>
  <body>
    <h1>Selamat Datang!</h1>
    <p>Ini adalah paragraf pertama.</p>
    <p>Ini adalah paragraf kedua.</p>
  </body>
</html>
```

Gambar 2.2 Dokumen HTML

Berdasarkan dokumen HTML tersebut DOM akan merepresentasikan struktur dokumen ini sebagai pohon seperti pada Gambar 3.2

```
- html
  - head
    - title
  - body
    - h1
    - p
    - p
```

Gambar 2.3 Representasi DOM HTML

Setiap *node* dalam pohon DOM merepresentasikan elemen atau bagian dari dokumen. *Node* dapat memiliki anak (*child*) dan atribut (*attribute*).

Node adalah unit dasar dalam pohon DOM (Document Object Model). DOM merepresentasikan struktur dokumen HTML dalam bentuk pohon, di mana setiap *node* mewakili elemen, teks, atau atribut dari dokumen tersebut. Element *Node* merupakan jenis *node* yang mewakili elemen HTML seperti ‘ ’, ‘ ’, atau ‘’. Setiap elemen *node* memiliki *tag name* yang mencerminkan jenis elemen HTML-nya. Sebagai contoh, dalam elemen ‘ Ini adalah paragraf pertama ’ adalah element *node* dengan *tag name* **‘**p’ (Arwin Dermawan et al., 2022).

Mutation Observer adalah API yang disediakan oleh *browser* terbaru untuk mendeteksi perubahan DOM pada suatu halaman (Iqbal, 2014). Mutation Observer sangat penting untuk mendeteksi perubahan DOM pada suatu halaman untuk dimanfaatkan dalam berbagai hal, dalam konteks pengembangan bot notula, Mutation Observer digunakan untuk mendeteksi kapan peserta rapat di Google Meet berbicara dengan mengawasi element html *microphone* yang mengalami perubahan dan mendeteksi keluar masuk peserta dalam rapat.

2.7 RabbitMQ

RabbitMQ adalah perantara yang menghubungkan berbagai layanan dalam arsitektur *microservice*, yang semakin populer seiring dengan perkembangan metodologi agile dan teknologi baru. Dalam konteks *microservice*, RabbitMQ membantu mengurangi beban dan waktu pengiriman pada aplikasi web server dengan mengalihkan tugas-tugas yang biasanya memerlukan banyak waktu dan sumber daya. Sistem antrian pesan memungkinkan server web untuk merespons permintaan dengan cepat tanpa harus menjalankan prosedur kompleks yang memakan waktu. Protokol pesan AMQP (Advanced Message Queuing Protocol) digunakan untuk mengelola penerbit (*publishers*) yang menghasilkan pesan dan konsumen (*consumers*) yang mengunduh serta memproses pesan tersebut. Peran utama *message broker* seperti

RabbitMQ adalah memastikan bahwa pesan dari *publisher* (pengirim pesan) sampai ke *consumer* (penerima pesan) yang tepat melalui komponen kunci yaitu *exchange* (Ćatović et al., 2022).

2.8 NestJs

NestJS adalah kerangka kerja (*framework*) untuk membangun aplikasi Node.js yang efisien dan modular. NestJS menggunakan TypeScript sebagai bahasa utama, tetapi juga mendukung JavaScript murni. Kerangka kerja ini dibangun di atas pustaka Express dan mengadopsi konsep-konsep dari arsitektur berbasis modul, yang terinspirasi dari Angular (Zima & Barszcz, 2024).

2.9 Meetbot

Meetbot dalam konteks ini merupakan salah satu proyek *open source* milik Balena Cloud, proyek ini pada dasarnya sudah diarsipkan dan tidak dikembangkan lebih lanjut karena sulit untuk dilakukan *maintanence* terhadap perubahan tampilan pengguna dan adopsi fitur Google Meet yang semakin meningkat (Gupta et al., 2022). Proyek ini sudah memiliki pondasi dan struktur kode yang cukup baik untuk menangani beberapa aktivitas inti seperti *login* menggunakan google dan masuk ke dalam rapat Google Meet, sehingga penulis cukup melakukan sedikit perubahan untuk menyesuaikan tampilan pengguna Google Meet terbaru. Sehingga masalah terkait perubahan tampilan pengguna pada Google Meet sudah dapat diatasi dengan cara menyesuaikan dengan DOM element pada halaman Google Meet terbaru.

2.10 Tinjauan Pustaka

Penulisan tugas akhir ini mengacu pada beberapa penelitian terdahulu seperti yang tercantum pada Tabel 2.1.

Tabel 2.1 Tinjauan Pustaka

No.	Peneliti	Judul	Tahun	Tujuan	Metode	Hasil
1.	Ke Ming Lyu, Ren Yuan Lyu, dan Hsien Tsung Chang	Real-time multilingual speech recognition and speaker diarization system based on Whisper segmentation	2024	Mengembangkan sistem pengenalan ucapan multibahasa real-time dan diarization pembicara menggunakan	Pengujian model dan sistem	Sistem menunjukkan tingkat kesalahan diarization kata (WDER) 2,68% untuk dua pembicara,

No.	Peneliti	Judul	Tahun	Tujuan	Metode	Hasil
				model Whisper dari OpenAI, dengan fokus pada ucapan Mandarin beraksen Taiwan dan pergantian pembicara yang sering		11,65% untuk tiga pembicara, dan 6,96% secara keseluruhan, sebanding dengan model non-real-time (Lyu et al., 2024)
2.	Terry Amorese, Claudia Greco, Marialucia Cuciniello, Rosa Milo, Olga Sheveleva, Neil Glackin	Automatic speech recognition (ASR) with Whisper: Testing Performances in Different Languages	2023	Menguji kinerja transkripsi model Automatic Speech Recognition (ASR) Whisper dari OpenAI untuk mendukung sistem diagnosis depresi otomatis melalui transkripsi data audio dari subjek di berbagai negara (Inggris, Italia, Rusia)	Pengujian model Whisper	Whisper menunjukkan kinerja optimal dalam pengenalan kata yang benar dengan tingkat kesalahan yang rendah dalam DEL, SUB, INS, dan WER untuk setiap bahasa yang diuji (Amorese et al., 2023)
3.	Tae Jin Park, Naoyuki Kanda,	A Review of Speaker Diarization: Recent	2022	Meninjau perkembangan historis dan kemajuan terbaru	Studi Literatur	Paper ini menyajikan tinjauan komprehensif

No.	Peneliti	Judul	Tahun	Tujuan	Metode	Hasil
	Dimitrios Dimitriadis, Kyu J Han, Shinji Watanabe, Shrikanth Narayanan	Advances with Deep Learning		dalam algoritma diarization pembicara, khususnya dengan pendekatan neural, serta mengkaji integrasi sistem diarization dengan aplikasi pengenalan ucapan dan tren teknologi deep learning		tentang perkembangan neural diarization pembicara, menunjukkan bagaimana integrasi dengan pengenalan ucapan dan teknologi deep learning dapat saling melengkapi, serta memfasilitasi kemajuan lebih lanjut menuju diarization pembicara yang lebih efisien (Park et al., 2022)
4.	Srushti Nitin Ghadge	AI-Powered Information Retrieval in Meeting Records and Transcripts Enhancing	2024	Membandingkan metode pencarian tradisional dengan pencarian berbasis AI yang	Membandingkan akurasi dan kecepatan algoritma pencarian berbasis AI dengan metode	Algoritma pencarian berbasis text lebih akurat dan cepat dibandingkan pendekatan

No.	Peneliti	Judul	Tahun	Tujuan	Metode	Hasil
		Efficiency and User Experience		terintegrasi dengan video dan transkrip, serta mengevaluasi efisiensi pencarian dan kepuasan pengguna	pencarian konvensional	tradisional menggunakan video(Nitin Ghadge, 2024)
5.	Naoyuki Kanda, Guoli Ye, Yu Wu, Yashesh Gaur, Xiaofei Wang, Zhong Meng, Zhuo Chen, Takuya Yoshioka	Large-Scale Pre-Training of End-to-End Multi-Talker ASR for Meeting Transcription with Single Distant Microphone	2022	Menyelidiki pengenalan ucapan otomatis (ASR) untuk transkripsi rapat dengan ucapan tumpang tindih menggunakan satu mikrofon jarak jauh (SDM) melalui pendekatan dua langkah: pelatihan awal dengan data simulasi skala besar dan penyetelan dengan data rapat nyata skala kecil	Studi literatur, analisis, kebutuhan, perancangan sistem, implementasi, pengujian sistem, kesimpulan dan saran	Model SOT ASR mencapai tingkat kesalahan kata (WER) 21,2% pada set evaluasi AMI-SDM, melebihi hasil WER 36,4% dari model sebelumnya dengan informasi batas ucapan oracle, dan lebih baik dari model ASR tunggal yang disetel pada audio beamformed

No.	Peneliti	Judul	Tahun	Tujuan	Metode	Hasil
						(Kanda et al., 2021)

Dari beberapa penelitian terdahulu, terlihat bahwa banyak upaya telah dilakukan untuk meningkatkan pengenalan ucapan otomatis (ASR) dan diarization pembicara dalam berbagai konteks. Dalam penelitian ini, pengembangan bot notula Google Meet memiliki beberapa kesamaan dan perbedaan dengan penelitian sebelumnya.

Terry Amorese et al. (2023) menguji kinerja model ASR Whisper dalam berbagai bahasa untuk mendukung sistem diagnosis depresi otomatis. Sementara fokus mereka adalah pada aplikasi medis, kesamaan terletak pada penggunaan model Whisper untuk pengenalan ucapan lintas bahasa. Bot notula ini juga memanfaatkan kemampuan lintas bahasa dari Whisper untuk menghasilkan transkripsi yang akurat.

Penelitian Tae Jin Park et al. (2022) meninjau kemajuan terbaru dalam algoritma diarization pembicara dengan pendekatan neural. Bot notula ini juga memanfaatkan teknologi diarization untuk memisahkan dan mengenali berbagai pembicara dalam rapat, menunjukkan kesamaan dalam tujuan peningkatan efisiensi diarization pembicara.

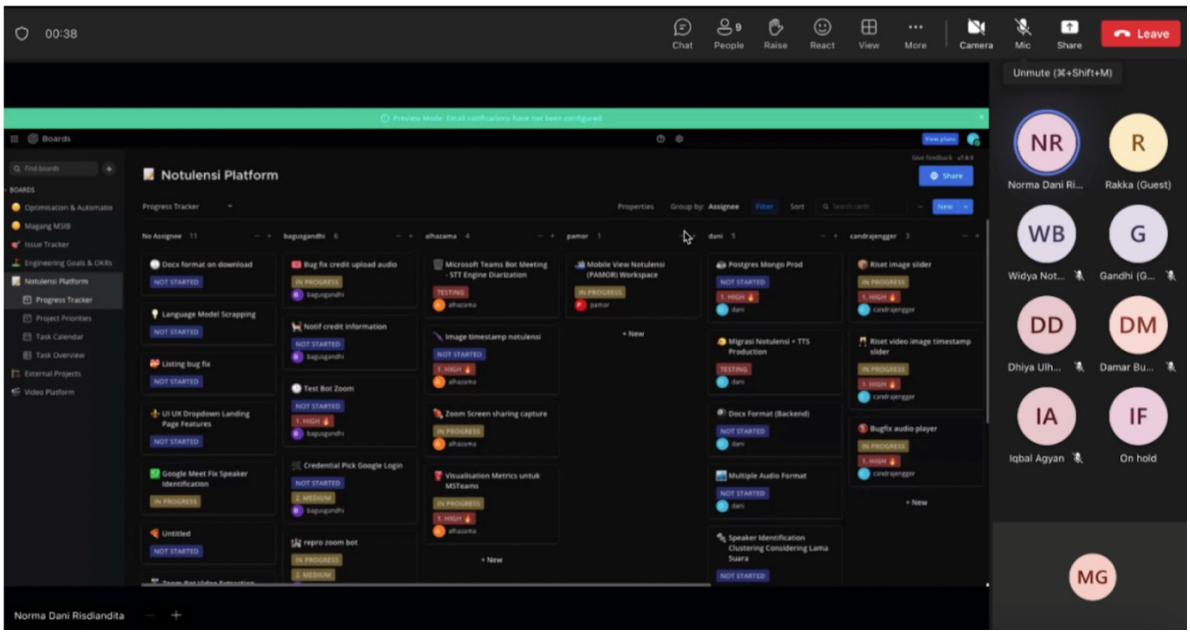
Namun, terdapat perbedaan signifikan dalam metode dan alat yang digunakan. Bot notula ini menggunakan RabbitMQ untuk komunikasi antar layanan dan Google Cloud Storage untuk penyimpanan *file* audio, yang tidak ditemukan dalam penelitian sebelumnya. Selain itu, penggunaan Puppeteer untuk otomatisasi interaksi dengan Google Meet adalah inovasi tersendiri yang tidak dibahas dalam penelitian lain.

Dengan demikian, proyek bot notula Google Meet ini tidak hanya mengikuti tren dan kemajuan dalam pengenalan ucapan otomatis dan diarization pembicara, tetapi juga memperkenalkan inovasi dalam hal integrasi teknologi dan peningkatan efisiensi dalam konteks pencatatan rapat otomatis. Integrasi Puppeteer untuk otomatisasi interaksi di Google Meet memungkinkan bot untuk mengikuti rapat tanpa campur tangan manual, meningkatkan efisiensi dan akurasi pencatatan rapat.

BAB III
PELAKSANAAN MAGANG

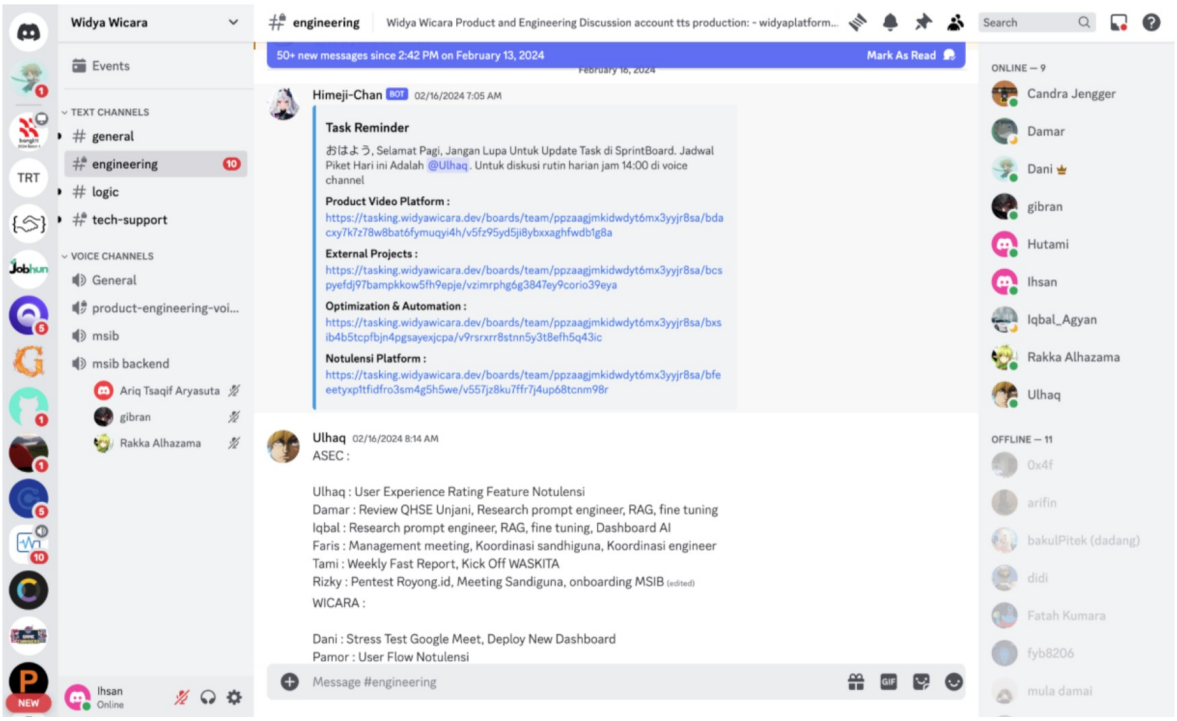
3.1 Manajemen Proyek

Penulis mendapatkan kesempatan untuk terlibat sebagai Backend Developer dalam pengembangan produk aplikasi di Widya Robotics. Selama masa magang, penulis dibimbing oleh beberapa mentor yang memiliki peran serupa dan ditempatkan di tim pengembang yang fokus pada penggunaan teknologi kecerdasan buatan dalam pengolahan audio. Selama periode magang, penulis menjalani jam kerja selama 8 jam sehari, dimulai dari pukul 08.00 hingga 17.00 dengan waktu istirahat pada pukul 12.00 hingga 13.00. Untuk koordinasi pemberian tugas, rutin diadakan pertemuan antara penulis dan mentor pada akhir setiap periode pengerjaan tugas. Berikut tangkapan layar rapat harian antara penulis, supervisor, dan staff pada Gambar 3.1.

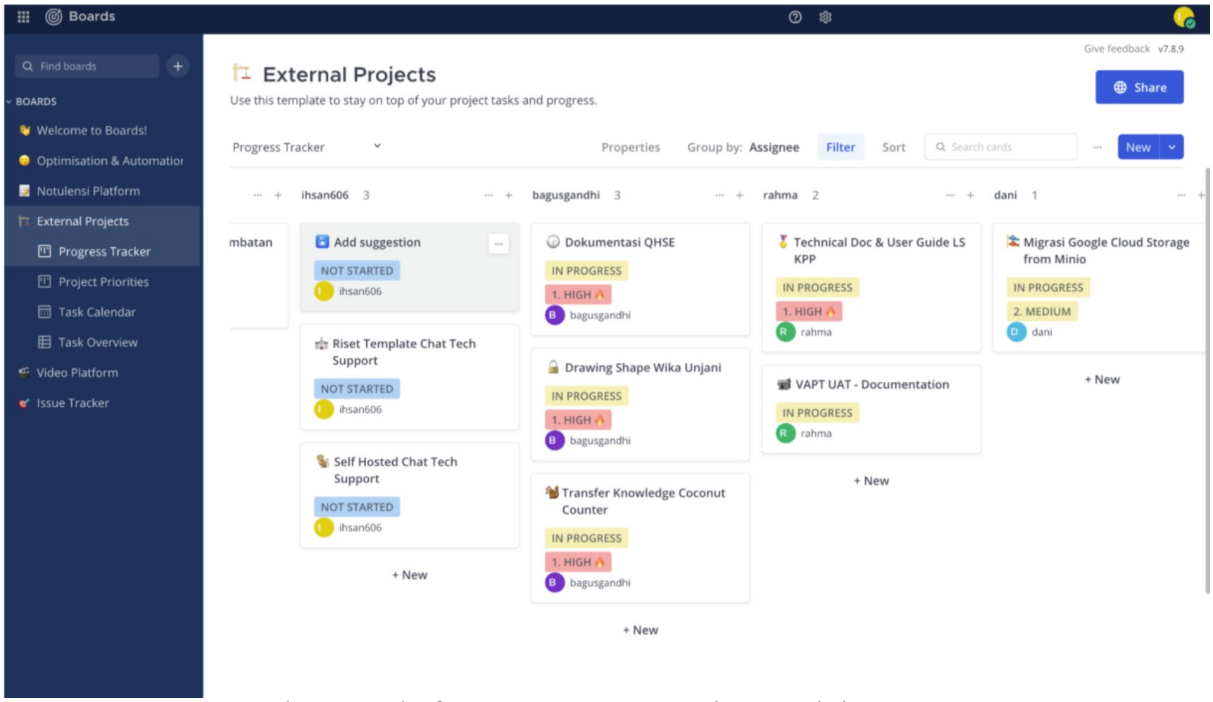


Gambar 3.1 Tangkapan Layar *Daily Meeting*

Pada Gambar 3.1 menunjukkan rapat harian yang dilakukan untuk membahas laporan progres pengerjaan tugas dan penugasan tugas baru oleh supevisor yang rutin dilakukan untuk memastikan kendala pengerjaan dapat disampaikan agar segera dibantu oleh senior staff di bidangnya. Selain rapat harian, penulis juga menggunakan platform Discord dan Mattermost untuk sarana komunikasi dan pengelola tugas selama magang, berikut tangkapan layar dari masing-masing platform pada Gambar 3.2 dan Gambar 3.3.



Gambar 3.2 Platform Discord untuk Komunikasi Tim



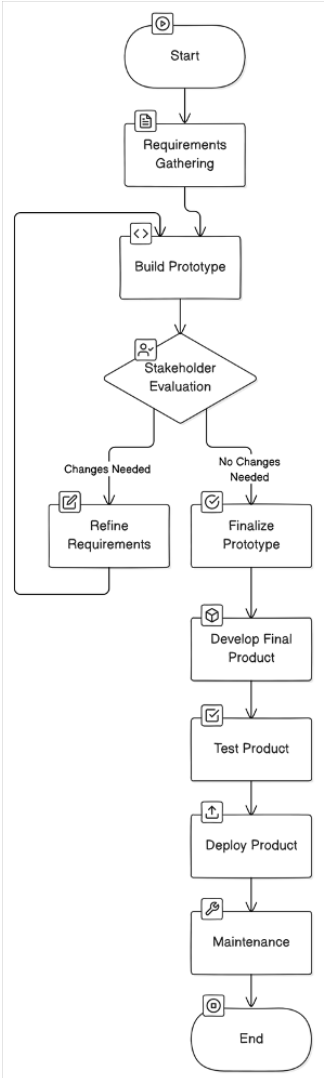
Gambar 3.3 Platform Mattermost untuk Pengelolaan Tugas

Gambar 3.2 menunjukan interaksi antar anggota tim untuk melaporkan pekerjaan yang akan dilakukan di hari tersebut. Proses ini dilakukan setelah Bot discord yang telah terhubung dengan basisdata Mattermost memberi pengingat jadwal pemandu acara *daily meeting*. Sedangkan Gambar 3.3 menunjukan daftar tugas pada aplikasi Mattermost yang telah ditugaskan kepada setiap anggota tim baik magang maupun karyawan, dengan aplikasi

Mattermost semua perkembangan task dapat dipantau oleh siapapun di internal tim sehingga dapat memastikan semua task dilaksanakan dengan baik.

Para mentor menekankan beberapa aspek penting yang sangat dihargai dalam dunia kerja, terutama komunikasi. Penulis diharuskan untuk selalu berkomunikasi mengenai perkembangan maupun hambatan dalam menyelesaikan tugas yang diberikan setiap hari. Hal ini bertujuan untuk memastikan setiap tugas dapat diselesaikan dengan baik dan komunikasi antar anggota tim tetap berjalan lancar. Pada kegiatan magang ini, penulis ditempatkan sebagai Backend Developer Intern dengan tugas mengerjakan issue/task yang telah diberikan oleh mentor, membantu pengembangan aplikasi bersama tim, serta mengisi logbook dan laporan mingguan.

Di Widya Robotics, penggunaan model System Development Life Cycle (SDLC) *prototyping* sangat penting. Lingkungan perusahaan rintisan menuntut kecepatan dalam perilisan produk agar bisa mendapatkan umpan balik dari pengguna secepat mungkin (Firmansyah et al., 2021). Hal ini memungkinkan produk yang dikembangkan untuk lebih cepat beradaptasi dan lincah dalam memenuhi kebutuhan pengguna. SDLC *prototyping* dimulai dengan pengumpulan kebutuhan awal dari *stakeholder* untuk memahami kebutuhan dasar sistem yang akan dibangun (De Cerff et al., 2018). Fokus utama adalah pada kebutuhan fungsional dan non-fungsional yang kritis. Berdasarkan kebutuhan yang telah dikumpulkan, pengembang membuat prototipe awal. Prototipe ini merupakan representasi sementara dari sistem yang akan dibangun, yang mencakup fitur dasar dan antarmuka pengguna yang sederhana. Pengguna atau *stakeholder* kemudian diminta untuk mengevaluasi prototipe awal dan memberikan umpan balik. Proses ini sangat penting untuk mengidentifikasi masalah dan menentukan perubahan yang diperlukan. Berdasarkan umpan balik yang diterima, pengembang melakukan perbaikan dan penyempurnaan pada prototipe. Proses ini berulang sampai prototipe memenuhi kebutuhan pengguna dan siap untuk dikembangkan menjadi produk akhir. Penggunaan SDLC *prototyping* di Widya Robotics memungkinkan pengembangan produk yang cepat dan adaptif, sehingga memenuhi kebutuhan pengguna dengan lebih baik.



Gambar 3.4 Metode *Prototyping* di Widya Robotics

1. Pengumpulan Kebutuhan

   Tahap ini melibatkan pengumpulan informasi dari stakeholder untuk memahami kebutuhan dasar sistem yang akan dibangun. Fokus utama adalah pada kebutuhan fungsional dan non-fungsional yang kritis. Proses ini biasanya dilakukan melalui wawancara, survei, dan analisis dokumen.

2. Pembuatan Prototipe Awal

   Berdasarkan kebutuhan yang telah dikumpulkan, pengembang membuat prototipe awal. Prototipe ini merupakan representasi sementara dari sistem yang akan dibangun, yang mencakup fitur dasar dan antarmuka pengguna yang sederhana. Prototipe awal ini memungkinkan pengembang dan pengguna untuk memahami dan mengevaluasi kebutuhan sistem.

3. Evaluasi dan Umpan Balik Pengguna

   Pengguna atau stakeholder diminta untuk mengevaluasi prototipe awal dan memberikan umpan balik. Proses ini sangat penting untuk mengidentifikasi masalah dan menentukan perubahan yang diperlukan. Umpan balik pengguna membantu pengembang memahami apakah sistem sudah memenuhi kebutuhan pengguna.

4. Penyempurnaan Prototipe

   Berdasarkan umpan balik yang diterima, pengembang melakukan perbaikan dan penyempurnaan pada prototipe. Proses ini bisa berulang beberapa kali hingga prototipe memenuhi ekspektasi pengguna. Penyempurnaan ini mencakup penambahan fitur baru, perbaikan antarmuka, dan peningkatan performa.

5. Pengembangan Produk Akhir

   Setelah prototipe disempurnakan dan divalidasi, pengembang mulai bekerja pada pengembangan sistem final. Pada tahap ini, semua fitur dan fungsionalitas yang diperlukan diimplementasikan secara penuh, sesuai dengan spesifikasi yang telah disepakati.

6. Pengujian dan Implementasi

   Sistem yang telah dikembangkan kemudian diuji secara menyeluruh untuk memastikan semua fitur bekerja dengan baik dan tidak ada bug yang signifikan. Setelah pengujian selesai dan sistem dinyatakan siap, sistem diimplementasikan di lingkungan produksi.

7. Pemeliharaan

   Setelah sistem diimplementasikan, tahap pemeliharaan dilakukan untuk memastikan sistem tetap berjalan dengan baik. Ini meliputi perbaikan bug, pembaruan sistem, dan peningkatan performa berdasarkan umpan balik yang terus berlanjut dari pengguna.

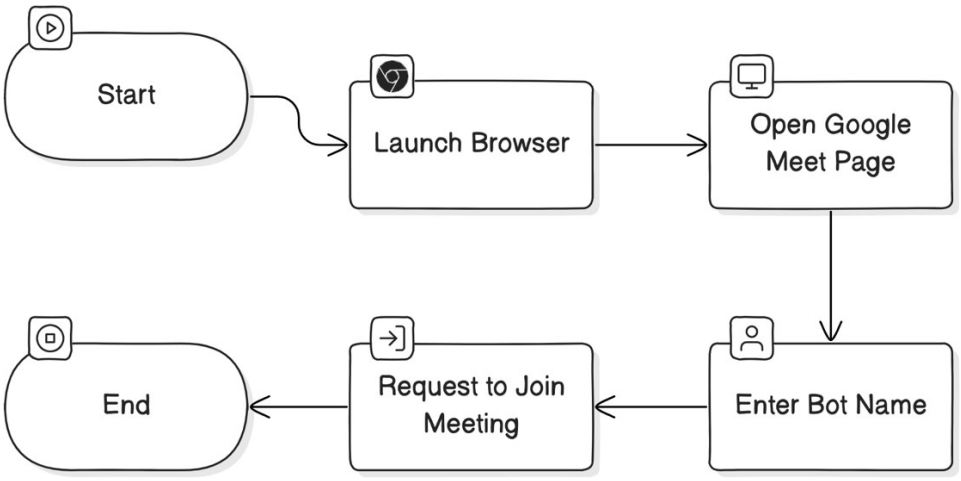
3.2 Pelaksanaan Proyek

Proyek ini merupakan salah satu bagian dari pengembangan fitur di platform Widya Notulensi. Widya Notulensi adalah aplikasi *Software as a Service* (SaaS) yang dapat melakukan transkripsi rapat di berbagai platform rapat online seperti Zoom, Microsoft Teams, dan Google Meet. Pada saat penulis bergabung sebagai staf magang, Widya Notulensi hanya terintegrasi dengan Zoom, dan penulis mendapatkan tugas untuk melanjutkan pengembangan integrasi pada platform Google Meet atas perintah VP Engineering. Untuk mengakomodasi kebutuhan tersebut, diputuskan bahwa akan dibuat dua layanan (*service)* terpisah, yaitu Bot Notula Engine dan Audio Processing. Google Meet *bot engine* dikembangkan menggunakan *framework* Nestjs selain karena sudah menjadi standar yang digunakan ketika akan membangun aplikasi *backend* di Widya Robotics juga karena sifatnya yang modular sehingga

aplikasi menjadi lebih mudah ketika dilakukan pemeliharaan setelah tahap pengembangan. *Service* Audio Processing dikembangkan menggunakan bahasa pemrograman python mempertimbangkan banyaknya pustaka yang siap digunakan untuk melakukan pemrosesan *audio* seperti pyannote.audio yang dipublikasikan di Hugging Face, sebuah platform untuk mendistribusikan pustaka, alat, maupun model kecerdesan buatan.

3.2.1 Implementasi Puppeteer pada Bot Notula Google Meet

Dalam pengembangan Bot Notula untuk Google Meet, Puppeteer diimplementasikan menggunakan *framework* NestJs. Puppeteer adalah pustaka Nodejs yang memungkinkan pengendalian *browser* Chrome atau Chromium secara otomatis melalui Protokol DevTools . Implementasi ini bertujuan untuk mengotomatisasi proses masuk, berpartisipasi, dan merekam audio rapat di Google Meet. Untuk mengembangkan bot notula yang mampu menangani aktivitas dasar otomatisasi Google Meet, penulis merujuk pada salah satu proyek *open source* milik Balena Cloud yang bernama *meetbot,* proyek sudah diarsipkan dan tidak dikembangkan lebih lanjut karena sulit untuk dilakukan *maintanence* terhadap perubahan tampilan pengguna dan adopsi fitur Google Meet yang semakin meningkat. Proyek ini sudah memiliki pondasi dan struktur kode yang cukup baik untuk menangani beberapa aktivitas inti seperti *login* menggunakan google dan masuk ke dalam rapat Google Meet, sehingga penulis cukup melakukan sedikit perubahan untuk menyesuaikan tampilan pengguna Google Meet terbaru. Untuk dapat lebih memahami alur puppeteer digunakan untuk automasi masuk ke dalam rapat dapat melihat alur diagram pada gambar.



Gambar 3.5 Diagram Alur Proses Automasi Bergabung dengan Rapat Google Meet

Diagaram pada Gambar 3.5 menggambarkan langkah-langkah dasar yang dilakukan oleh bot untuk dapat bergabung ke dalam pertemuan Google Meet secara otomatis menggunakan

puppeteer. Proses ini melibatkan beberapa tahapan kunci seperti peluncuran *browser*, navigasi ke halaman Google Meet, pengisian nama bot, dan pengiriman permintaan untuk bergabung ke dalam pertemuan. Untuk dapat menangani semua proses tersebut di proyek berbasis Nestjs yang telah dipersiapkan sebelumnya, perlu dilakukan beberapa proses sebagai berikut:

1. Instalasi Puppeteer dan Puppeteer Extra

   Framework NestJs berjalan di lingkungan NodeJs, sehingga proses instalasi pustaka tersebut dapat dilakukan menggunakan beberapa *package manager* seperti NPM dan Yarn. Berikut adalah perintah instalasinya:

```
npm install puppeteer puppeteer-extra puppeteer-extra-plugin-stealth
puppeteer-extra-plugin-user-preferences
```

Gambar 3.6 Perintah Instalasi Puppeteer dan Puppeteer Extra

Gambar 3.6 akan melakukan instalasi pustaka Puppeteer dan Puppeteer Extra yang digunakan untuk menambahkan plugin tambahan seperti Stealth Plugin dan User Preferences Plugin.

2. Membuat Module MeetBot di NestJs

   Setiap aplikasi NestJs memiliki satu modul utama, yang berfungsi sebagai pusat dari semua modul lain yang akan diimport ke dalamnya. Berikut kutipan kode Modul utama pada NestJs:

```
@Module({ imports: [], controllers: [], providers: [], }) export class
AppModule {}
```

Gambar 3.7 Modul Utama NestJs

Pada Gambar 3.7 tampak bahwa class AppModule merupakan class kosong, hal itu didesain agar class utama hanya menjadi modul yang akan menjalankan modul-modul yang lain dengan cara melakukan import ke dalam *class* tersebut. Dalam pengembangan Bot Notula Google Meet, langkah pertama adalah membuat module MeetBot yang akan mengatur seluruh logika dan fungsi terkait dengan otomatisasi Google Meet. Modul dapat dibuat dengan mudah melalui *command line interface* (CLI) dengan perintah berikut:

```
nest generate module meet-bot
```

Gambar 3.8 Perintah Membuat Modul MeetBot

Perintah pada Gambar 3.8 akan membuat *file* meet-bot.module.ts di folder meet-bot dan juga akan melakukan import modul meet-bot ke modul utama, sehingga modul utama akan menjadi seperti kutipan kode berikut:

```
import { MeetBotModule } from './meet-bot/meet-bot.module';


@Module({
imports: [MeetBotModule],
controllers: [],
providers: [], })
export class AppModule {}
```

Gambar 3.9 Kutipan Kode Modul Utama Setelah Import Modul MeetBot

Kode Program pada Gambar 3.9 menunjukkan ‘MeetBotModule’ yang berhasil dibuat menggunakan perintah pada Gambar 3.8 telah diimpor ke dalam ‘AppModule’ secara otomatis, sehingga tidak perlu melakukan impor secara manual.

3. Implementasi Proyek Meetbot dari Balena Cloud

Setelah membuat modul MeetBot proses implementasi proyek Meetbot dari Balena Cloud yang dijadikan rujukan oleh penulis dapat dilakukan, mencakup pembuatan *class* MeetBot dan menuliskan beberapa funsionalitas dasar seperti masuk ke dalam rapat Google Meet tanpa melakukan *login* dan masuk ke dalam rapat, berikut adalah potongan kode yang dapat mengakomodasi fungsionalitas tersebut:

```
1. async joinMeet() {
2.   if (this.page instanceof Page === false) {
3.     throw new Error('Init page before join to meeting’);
4.   }
5.   try {
6.     await this.page.goto(this.url + '?hl=en', {
7.         waitUntil: 'networkidle0',
8.         timeout: 0,
9.     });
10.    const inputNameField =
11.        await this.page.waitForSelector(
12.          'input[type="text"]',
13.          { timeout: 0 },
14.         );
15.    this.logger.debug(`Entering name as ${Env.BOT_NAME}`);
16.    await inputNameField.type(Env.BOT_NAME, { delay: 10 });
17.    await this.page.keyboard.press('Enter');
18.    this.logger.debug(`Asked to join at ${new Date()}`);
19.
20.   }catch(()=> {
21.     return null;
22.   })
23. }
```

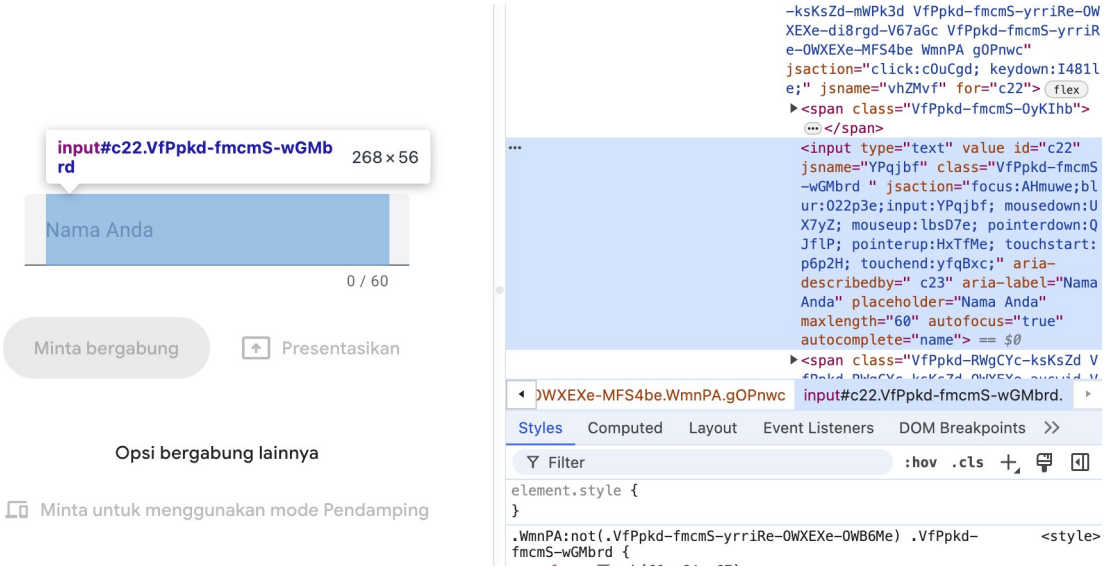
Gambar 3.10 Kutipan Kode untuk Bergabung ke Google Meet Menggunakan Puppeteer

Berdasarkan Gambar 3.10 pada awal fungsi ‘joinMeet’ tersebut dideklarasikan sebagai fungsi *asynchronous* menggunakan ‘async’, yang memungkinkan penggunaan await di dalamnya untuk menangani operasi *asynchronous*. Ini penting dalam konteks web *scraping* atau automasi *browser*, dimana sering harus menunggu berbagai proses selesai sebelum melanjutkan ke langkah berikutnya. Pada baris 2 dan 3, terdapat pengecekan untuk memastikan bahwa objek this.page telah diinisialisasi dan merupakan *instance* dari *class* ‘Page’ yang disediakan oleh pustaka Puppeteer. Jika ‘this.page’ bukan *instance* dari ‘Page’, fungsi akan melemparkan sebuah *error* dengan pesan "Init page before join to meeting". Ini adalah langkah pencegahan untuk memastikan bahwa semua prasyarat sudah terpenuhi sebelum mencoba bergabung ke rapat.

Setelah memastikan bahwa halaman sudah diinisialisasi dengan benar, langkah berikutnya adalah membuka URL Google Meet yang telah ditentukan. URL ini memiliki

parameter ‘?hl=en’ untuk memastikan bahwa halaman ditampilkan dalam bahasa Inggris, yang mungkin lebih mudah untuk ditangani oleh bot. Fungsi ini memulai blok ‘try’ untuk menangani potensi *error* yang mungkin terjadi selama proses navigasi ke halaman Google Meet. Puppeteer kemudian diperintahkan untuk membuka URL meeting yang ditentukan ‘(this.url + '?hl=en')’. Opsi ‘waitUntil: networkidle0’ digunakan untuk memastikan bahwa halaman dianggap telah dimuat ketika tidak ada lagi jaringan yang aktif selama sekian waktu, dan ‘timeout: 0’ digunakan untuk menonaktifkan batas waktu, sehingga proses ini akan menunggu sampai halaman benar-benar dimuat tanpa batas waktu tertentu.

Setelah halaman berhasil dimuat, langkah berikutnya adalah mencari elemen input untuk mengisi nama pengguna yang akan bergabung ke meeting. Hal ini dilakukan dengan menunggu elemen input teks muncul di halaman, lalu mengetikkan nama bot ke dalamnya. Fungsi waitForSelector digunakan untuk menunggu sampai elemen input teks muncul di halaman dengan menggunakan selektor ‘input[type="text"]’. Selektor input[type="text"]’ tersebut dapat ditentukan dengan melihat elemen HTML pada halaman Google Meet dengan cara melakukan *inspect element*, berikut adalah tangkapan layar halaman Google Meet yang menampilkan kolom *input* untuk mengisikan nama peserta dan elemen HTML nya pada Gambar 3.11.



Gambar 3.11 Tangkapan Layar Elemen HTML Input Nama Peserta

Setelah elemen input ditemukan, objeknya disimpan dalam variabel ‘inputNameField’. Pada titik ini, fungsi mencatat log bahwa bot akan memasukkan nama yang diambil dari variabel ‘Env.BOT_NAME’ menggunakan ‘this.logger.debug’.

Setelah itu, nama bot diketikkan ke dalam elemen input bertipe *text* dengan sedikit penundaan antara setiap karakter untuk meniru input manusia menggunakan metode type. Setelah nama diketikkan, tombol 'Enter' ditekan untuk mengajukan permintaan bergabung ke meeting menggunakan metode keyboard.press. Waktu permintaan bergabung kemudian dicatat dalam log.

Jika terjadi kesalahan selama proses ini, blok 'catch' akan menangani *error* tersebut dengan mengembalikan nilai null, yang menunjukkan bahwa proses bergabung ke meeting gagal. Blok 'catch' menangkap *error* apapun yang terjadi dalam blok 'try', dan jika terjadi *error*, fungsi mengembalikan null untuk menunjukkan kegagalan.

Dengan penjelasan ini, setiap baris kode di dalam fungsi joinMeet diuraikan secara rinci, menggambarkan tujuan dan fungsionalitas dari masing-masing bagian kode. Hal ini memungkinkan pemahaman yang mendalam mengenai bagaimana bot diinisialisasi, bergabung ke meeting Google Meet, dan menangani potensi kesalahan yang terjadi selama proses tersebut.

4. Konfigurasi *Browser* Puppeteer

Untuk mendukung automasi di Google Meet, diperlukan penyesuaian konfigurasi *browser*. Tujuannya adalah untuk memastikan bahwa bot dapat beroperasi tanpa terdeteksi sebagai otomatisasi. Konfigurasi ini mencakup penggunaan *plugin* stealth untuk menghindari deteksi, serta pengaturan otomatisasi akses media seperti mikrofon dan kamera tanpa memunculkan *prompt* izin. Selain itu, penyesuaian bahasa dan ukuran jendela membantu mensimulasikan lingkungan pengguna nyata. Pengaturan jalur unduhan otomatis juga penting untuk menyimpan rekaman audio secara efisien, memastikan proses berjalan lancar tanpa intervensi manual. Dengan demikian, bot dapat berfungsi secara optimal dan mendukung tujuan pengembangan notula otomatis.

```
import puppeteer from 'puppeteer-extra';
import StealthPlugin from 'puppeteer-extra-plugin-stealth';
import    userPreferencesPlugin    from    'puppeteer-extra-plugin-user-
preferences';
import { getAudioDirectory } from './utils';

puppeteer.use(StealthPlugin());
puppeteer.use(
  userPreferencesPlugin({
   userPrefs: { download: {
     prompt_for_download: false,
     default_directory: getAudioDirectory(), },
   profile: {
    default_content_setting_values: { automatic_downloads: 1 }, },
    },
  }),
);
export async function newBrowser() {
  const puppeteerLaunchOptions = {
    args: [ '--use-fake-ui-for-media-stream',
           '--use-fake-device-for-media-stream',
           '--use-file-for-fake-audio-capture=/path/to/audio.wav',
           '--allow-file-access', '--lang=en', '--no-sandbox',
           '--window-size=1920,1080'],
    env: { LANG: 'en', },
    headless: false, handleSIGINT: false, handleSIGTERM: false, };
  const browser = await puppeteer.launch(puppeteerLaunchOptions);



  await browser.defaultBrowserContext()
      .overridePermissions('https://meet.google.com', [
      'microphone','camera', 'notifications', ]);
  return browser;
}
```

Gambar 3.12 Kutipan Kode Konfigurasi *Browser*

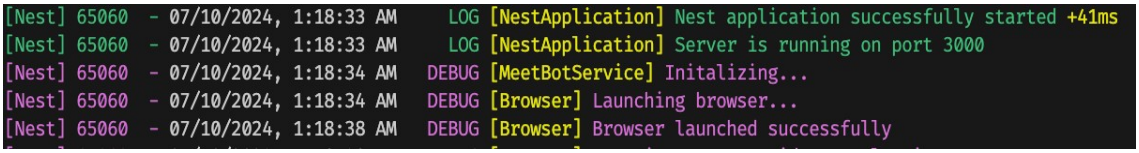
Berikut adalah penjelasan kutipan kode pada Gambar 3.12, Pertama, mengimpor beberapa modul penting, termasuk puppeteer-extra, yang merupakan versi modifikasi dari

Puppeteer untuk mendukung *plugin*. *Plugin* stealth digunakan untuk menyembunyikan bahwa *browser* dikendalikan oleh bot, membantu menghindari deteksi oleh situs web. *Plugin* 'user-preferences' diaktifkan dengan pengaturan yang mengarahkan unduhan ke direktori khusus yang ditentukan oleh fungsi 'getAudioDirectory()'. Ini mencegah munculnya *prompt* unduhan dan memungkinkan pengunduhan otomatis. Di dalam fungsi 'newBrowser()', penulis mendefinisikan konfigurasi peluncuran *browser* menggunakan PuppeteerLaunchOptions. Berikut penjelasan masing-masing konfigurasi:

a. '--use-fake-ui-for-media-stream' dan '--use-fake-device-for-media-stream': Opsi ini memungkinkan akses otomatis ke media stream (mikrofon dan kamera) tanpa perlu interaksi pengguna.
b. '--use-file-for-fake-audio-capture': Menentukan *file* audio yang digunakan sebagai input mikrofon palsu, yang berguna untuk simulasi input audio selama pengujian.
c. '--allow-file-access': Mengizinkan akses ke sistem file, penting untuk membaca file audio yang diperlukan.
d. '--lang=en': Mengatur bahasa *browser* ke bahasa Inggris, yang membantu menjaga konsistensi antarmuka pengguna.
e. '--no-sandbox': Menjalankan *browser* tanpa *sandbox* untuk meningkatkan kinerja. Namun, perlu diperhatikan bahwa ini dapat mengurangi keamanan.
f. '--window-size=1920,1080': Menentukan ukuran jendela browser untuk memastikan tampilan UI sesuai dengan yang diinginkan.

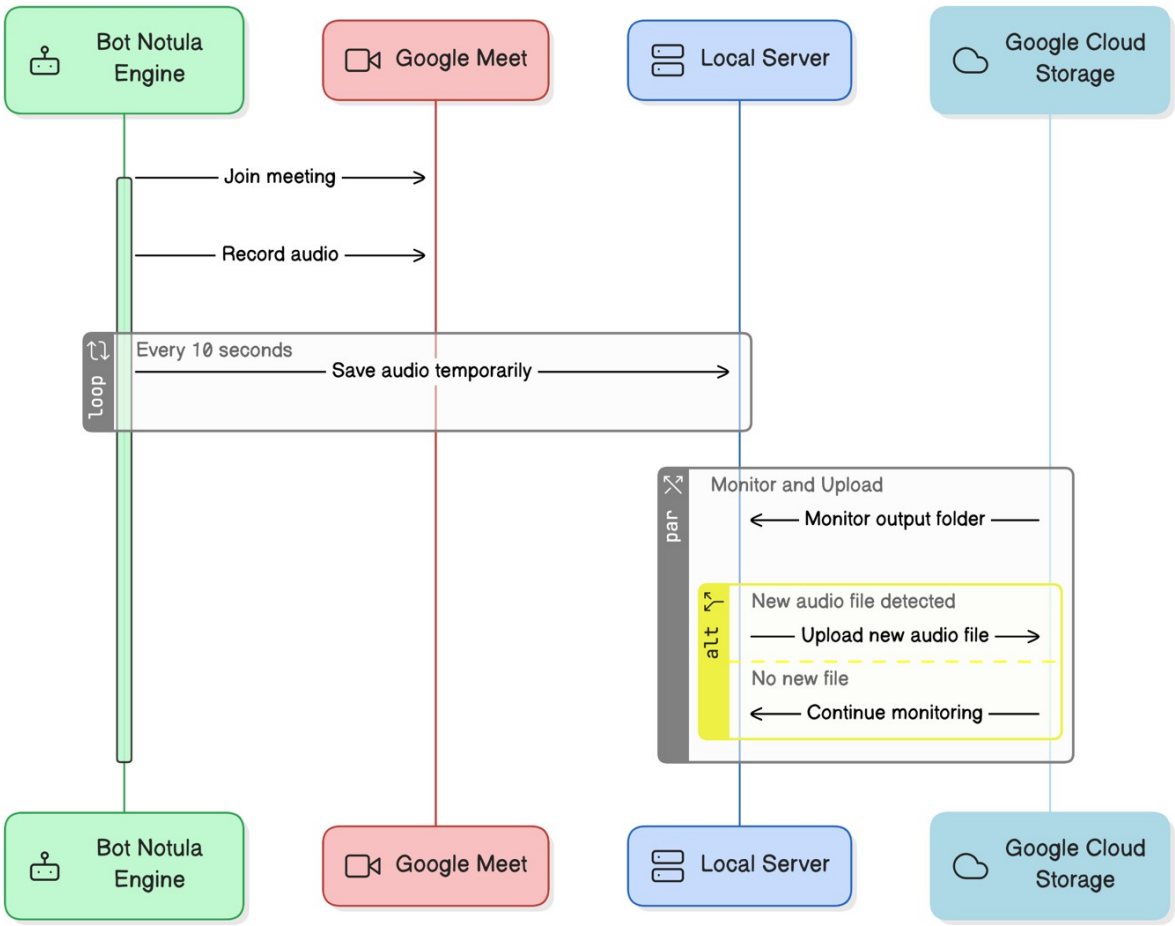
Pada pengaturan env, variabel lingkungan 'LANG' diatur untuk memastikan penggunaan bahasa Inggris. Konfigurasi *headless* menentukan mode tampilan *browser*. Jika 'Env.HEADLESS' diaktifkan, *browser* akan berjalan dalam mode *headless* (tanpa GUI), yang cocok untuk lingkungan server. Mode ini dapat disesuaikan dengan kebutuhan pengembangan. Pengaturan 'handleSIGINT' dan 'handleSIGTERM' memastikan bahwa sinyal penghentian ditangani secara manual, memberikan kontrol lebih lanjut terhadap penutupan *browser*. Kode selanjutnya mengecek apakah 'CHROMIUM_PATH' (lokasi *executable* Chromium) ada di sistem. Jika *path* Chromium ditemukan maka 'puppeteerLaunchOptions.executablePath' diatur ke path ini untuk memastikan Puppeteer menggunakan versi Chromium yang diinginkan. Setelah konfigurasi lengkap, *browser* diluncurkan menggunakan 'puppeteer.launch()' dengan opsi yang telah diatur. Pesan log dicetak untuk menandai bahwa *browser* telah diluncurkan. Setelah *browser* aktif, hak akses untuk mikrofon, kamera, dan notifikasi secara eksplisit diizinkan untuk domain

https://meet.google.com. Ini penting agar bot dapat bergabung ke rapat tanpa ada *prompt* izin yang mengganggu. Berikut adalah pesan log yang muncul di terminal ketika *browser* berhasil diluncurkan pada Gambar 3.13 pada baris terakhir.



Gambar 3.13 Pesan Log *Browser* Berhasil Diluncurkan

3.2.2 Implementasi Google Cloud Storage untuk Penyimpanan *File* Audio



Gambar 3.14 Sequence Diagram Unggah dan *Monitoring File* Audio

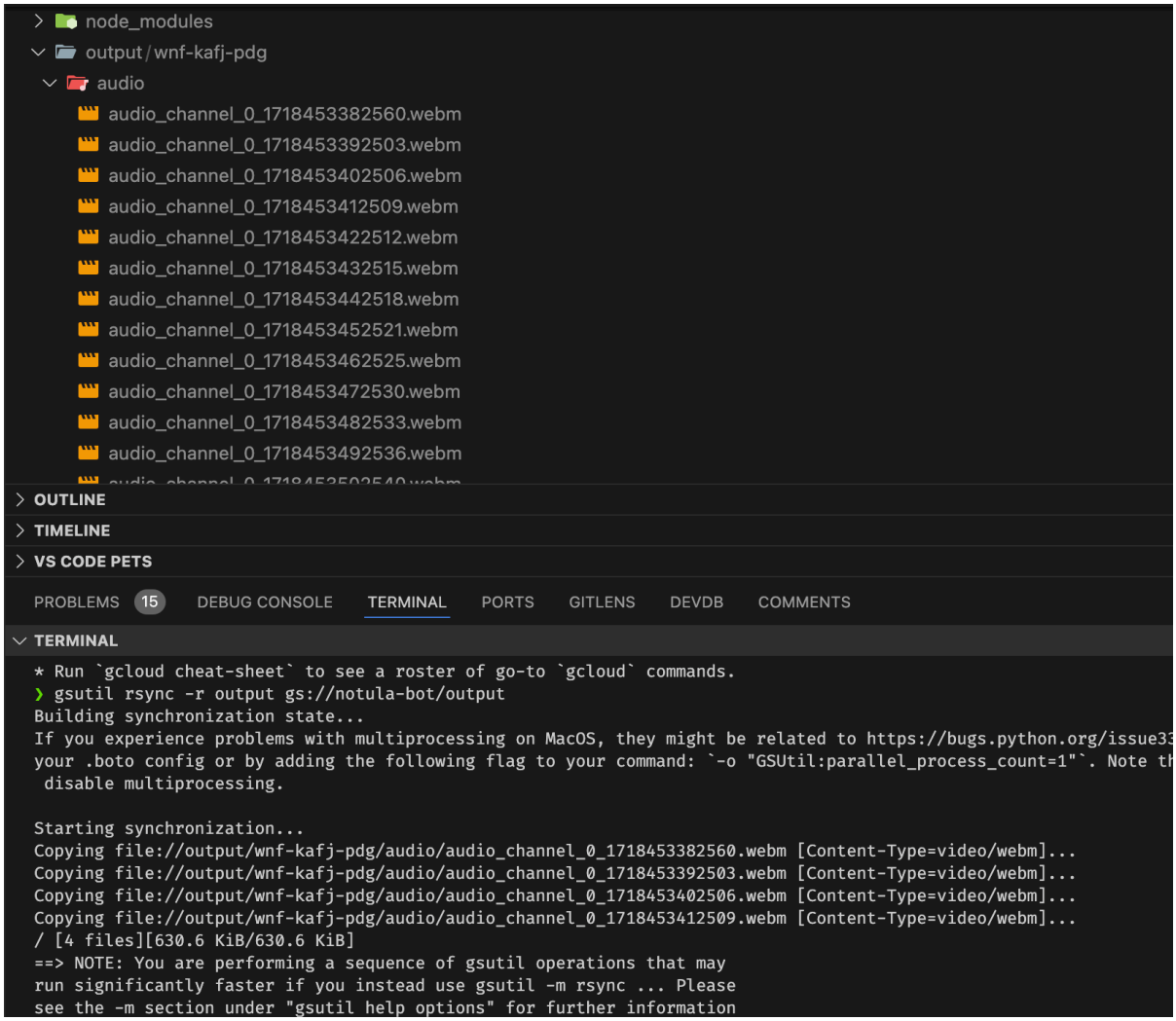
Google Cloud Storage (GCS) merupakan layanan penyimpanan objek yang andal dan *scallable* dari Google Cloud Platform. Pada implementasi bot notula untuk Google Meet, GCS digunakan sebagai media penyimpanan *file* audio yang direkam selama sesi rapat. Gambar 3.14 menunjukan proses unggah dengan cara memantau perubahan folder penyimpanan *file* audio di server lokal. Audio rapat yang direkam oleh Bot Notula Engine disimpan dalam bentuk file

audio secara parsial setiap 10 detik untuk meringankan proses unggah ke GCS. Dengan menggunakan GCS, *file* audio dapat diakses dan dikelola dengan mudah, serta terjamin keamanannya. Proses unggah audio hasil rapat dilakukan secara *realtime* dengan cara menjalankan kode untuk mengawasi perubahan pada folder output, jika ada penambahan *file* audio hasil rapat, maka folder output akan disinkronisasi ke GCS. Berikut kutipan kode untuk melakukan unggah audio secara *realtime*:

```
mkdir output
chmod a+rw ./output

fswatch -o output | while read f; do
    gsutil -m rsync -r output gs://$GCS_BUCKET_NAME /output
done
```

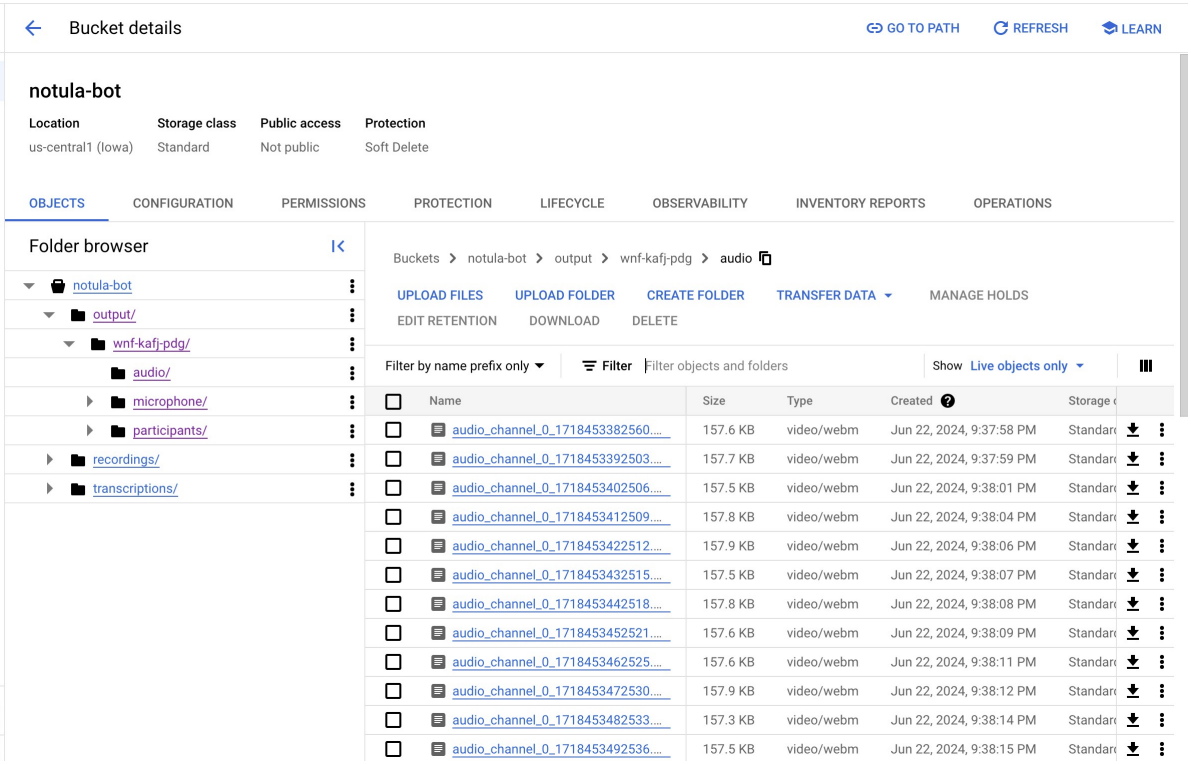
Gambar 3.15 Kode Unggah Audio Secara *Realtime*

Baris pertama dan kedua pada Gambar 3.15 berfungsi untuk membuat direktori khusus yang disebut ‘output’ di dalam struktur direktori aplikasi. Langkah ini penting karena direktori ‘output” akan menjadi tempat penyimpanan sementara untuk semua *file* audio yang direkam selama sesi rapat. Setelah direktori ‘output” dibuat, langkah selanjutnya adalah memastikan bahwa aplikasi memiliki izin akses yang tepat untuk dapat melakukan operasi menulis dan membaca *file* di dalam direktori tersebut. Izin ini penting untuk memastikan bahwa proses pengelolaan *file* audio, seperti perekaman, penyimpanan sementara, dan pengiriman ke penyimpanan permanen, dapat berjalan lancar tanpa kendala izin atau akses. Kode ‘fswatch -o output | while read f; do gsutil -m rsync -r output gs://$GCS_BUCKET_NAME /output; done’ berfungsi untuk melakukan sinkronisasi file secara real-time dari direktori lokal output ke bucket Google Cloud Storage (GCS). Dengan menggunakan perintah fswatch -o output, sistem memantau perubahan yang terjadi pada direktori output. Setiap kali ada perubahan seperti penambahan, modifikasi, atau penghapusan *file*, fswatch akan memberikan sinyal yang diteruskan ke perintah berikutnya. Perintah ‘while read f; do ... done’ membentuk sebuah *loop* yang menangkap setiap sinyal perubahan dari fswatch. Di dalam loop, ‘gsutil -m rsync -r output gs://$GCS_BUCKET_NAME/output’ digunakan untuk menyinkronkan semua isi direktori output lokal ke bucket GCS yang ditentukan ‘gs://$GCS_BUCKET_NAME’. Berikut adalah tangkapan layar ketika kode tersebut dijalankan:



Gambar 3.16 Tangkapan Layar Eksekusi Kode Unggah Audio

Pada Gambar 3.16 menunjukan folder ‘output’ yang berisi kumpulan potongan audio suatu rapat Google Meet yang sedang dilakukan proses unggah secara *realtime* ke Google Cloud Storage. Proses ini dapat dilakukan dengan memanfaatkan Pustaka fswatch yang berfungsi untuk mendeteksi perubahan *file* pada suatu *directory/*folder yang ditargetkan. Hasil audio rapat diletakan pada folder yang diberi nama dengan kode rapat Google Meet untuk memudahkan identifikasi pada tahap selanjutnya. Tangkapan layar audio yang berhasil diunggah pada Google Cloud Storage terdapat pada Gambar 3.17.



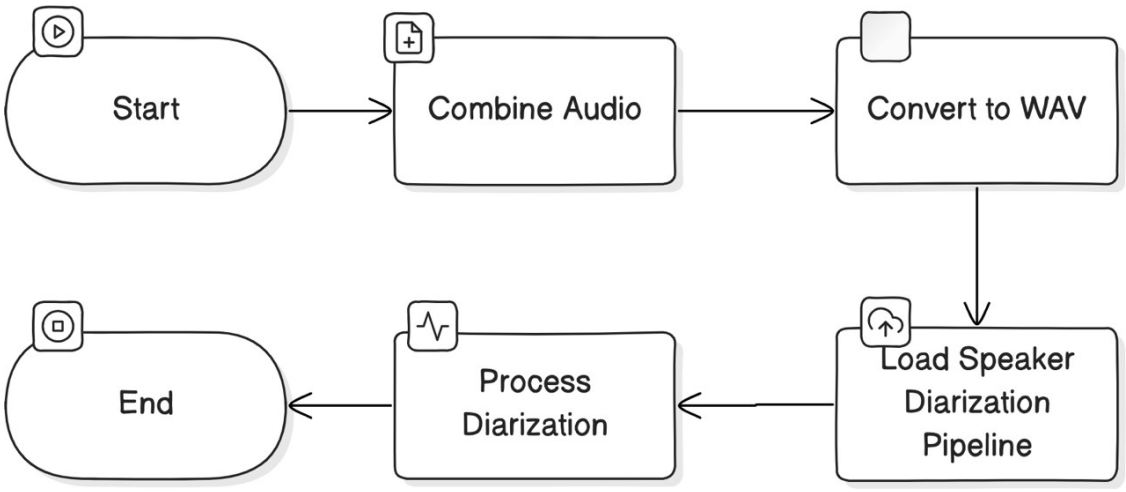
Gambar 3.17 Tangkapan Layar Google Cloud Storage

Implementasi Google Cloud Storage (GCS) untuk menyimpan file audio hasil rekaman rapat di Google Meet memberikan solusi yang andal dan efisien. Namun, ada beberapa tantangan yang perlu diperhatikan dan solusi yang dapat diterapkan untuk mengoptimalkan penggunaan storage:

1. Efisiensi Penyimpanan dan Pengelolaan File:
    - Tantangan: Penggunaan potongan file kecil untuk mempermudah unggahan real-time dapat meningkatkan *overhead* penyimpanan dan menyebabkan kompleksitas dalam manajemen file.
    - Solusi: Mengoptimalkan interval pemotongan file menjadi lebih panjang dapat mengurangi jumlah file kecil yang dihasilkan, sehingga menghemat ruang penyimpanan. Selain itu, kompresi file sebelum unggahan juga dapat mengurangi ukuran file yang disimpan di GCS. Penggunaan sistem penamaan yang konsisten, seperti menggunakan kode rapat Google Meet, membantu dalam manajemen file, dan kebijakan retensi yang otomatis dapat menghapus atau memindahkan file yang sudah tidak diperlukan untuk mengurangi biaya penyimpanan.

2. Biaya Penyimpanan:
    - Tantangan: Penyimpanan file audio dalam jumlah besar di GCS dapat menjadi mahal, terutama jika menggunakan kelas penyimpanan standar.
    - Solusi: Memindahkan file audio yang tidak sering diakses ke kelas penyimpanan yang lebih murah seperti coldline dapat mengurangi biaya. Selain itu, otomatisasi pemindahan file berdasarkan usia atau frekuensi akses dapat lebih mengoptimalkan biaya penyimpanan.

3.2.3 *Speaker Diarization* dengan Pyannote.audio



Gambar 3.18 Diagram Alur Proses *Diarization*

Diarization adalah proses identifikasi dan segmentasi audio berdasarkan pembicara. Dalam konteks pengembangan notula rapat , *diarization* memisahkan segmen audio yang diucapkan oleh pembicara berbeda sehingga setiap segmen berisi ucapan dari hanya satu pembicara. Pada tahap ini pengembangan dilakukan pada *service audio processing.* Gambar 3.18 menunjukan urutan proses *speaker diarization.* Audio yang telah diunggah di Google Cloud Storage pada tahap sebelumnya masih berupa potongan audio per 10 detik, sehingga perlu untuk menyatukan semua audio tersebut menjadi satu agar lalu formatnya dikonversi ke WAV agar lebih mudah diproses untuk *speaker diarization.* Proses tersebut dapat dilakukan dengan memanfaatkan Pydub dan FFmpeg. Pydub adalah *library* Python untuk manipulasi audio, sedangkan FFmpeg adalah alat pemrosesan multimedia. Keduanya dapat digunakan untuk memotong, menggabungkan, dan mengatur properti audio dengan kompatibilitas dan kemudahan

pemrosesan (Sushil, 2024). Berikut kutipan kode untuk melakukan penggabungan audio dan konversi pada Gambar 3.19.

```
import os
from pydub import AudioSegment

def merge_and_convert(folder_path, output_path):
     audio_files = sorted([f for f in os.listdir(folder_path) if
f.endswith('.webm')])
     combined = AudioSegment.empty()
     for file in audio_files:
         audio = AudioSegment.from_file(os.path.join(folder_path, file),
format='webm')
         combined += audio
         combined.export(output_path, format='wav')
```

Gambar 3.19 Kutipan Kode Menggabungkan Audio

Gambar 3.19 mendefinisikan sebuah fungsi bernama ‘merge_and_convert’ yang bertujuan untuk menggabungkan beberapa *file* audio berformat .webm dan mengonversinya menjadi satu *file* audio berformat ‘.wav’. Fungsi ini memanfaatkan pustaka os untuk operasi sistem dan pydub untuk manipulasi audio. Pada awalnya, fungsi ini mengimpor pustaka yang diperlukan dan kemudian mendefinisikan fungsi yang menerima dua argumen: ‘folder_path’, yaitu jalur folder yang berisi *file* audio ‘.webm’, dan ‘output_path’, yaitu jalur untuk *file* audio hasil keluaran. Selanjutnya, fungsi ini mengambil semua *file* berformat .webm dari folder yang ditentukan dan mengurutkannya. Kemudian, fungsi ini membuat objek ‘AudioSegment’ kosong untuk menampung hasil gabungan audio. Dalam *loop*, fungsi ini membaca setiap file audio .webm, mengonversinya menjadi objek AudioSegment, dan menambahkannya ke objek combined. Setelah semua *file* audio digabungkan, fungsi ini mengekspor audio yang telah digabungkan menjadi satu *file* berformat .wav di folder keluaran yang telah ditentukan. Dengan demikian, fungsi ini mengotomatisasi proses penggabungan dan konversi *file* audio secara efisien.

Setelah proses penggabungan dan konversi selesai, proses diarization dapat dilakukan. Dalam penelitian ini, pustaka pyannote.audio digunakan untuk melakukan diarization. Pyannote.audio merupakan alat yang andal dan sering digunakan dalam pengolahan sinyal

suara, terutama untuk tugas-tugas seperti pengenalan pembicara dan segmentasi pembicara. Kutipan kode untuk melakukan *speaker diarization* ditunjukan pada Gambar 3.20.

```
from pyannote.audio import Pipeline

pipeline= Pipeline.from_pretrained(
'pyannote/speaker-diarization', use_auth_token= (access_token) or True )

AUDIO_FILE = {'uri': 'None', 'audio': 'input_prep.wav'}
dz = pipeline(AUDIO_FILE)
with open("diarization.txt", "w") as text_file:
  text_file.write(str(dz))
```

Gambar 3.20 Kutipan Kode Speaker Diarization

Berikut penjelasan kode program pada Gambar 3.20. Pada baris pertama terdapat impor kelas ‘Pipeline’ dari pustaka pyannote.audio. *Pipeline diarization* kemudian diinisialisasi menggunakan model *pre-trained* ‘pyannote/speaker-diarization’, yang diakses melalui token otentikasi yang diperoleh dari platform Hugging Face. *File* audio yang akan dianalisis ditentukan dalam variabel AUDIO_FILE, yang berisi URI dan jalur file audio input_prep.wav. Pipeline diarization diterapkan pada *file* audio ini, menghasilkan objek dz yang berisi hasil *diarization*. Hasil ini kemudian disimpan dalam file teks bernama diarization.txt. Berikut adalah contoh hasil luaran dari proses diatas.

```
(<Segment(2.59597, 4.14847)>, 'A', 'SPEAKER_00')
(<Segment(5.49847, 8.18159)>, 'B', 'SPEAKER_00')
(<Segment(8.95784, 10.291)>, 'C', 'SPEAKER_01')
(<Segment(12.1641, 14.746)>, 'D', 'SPEAKER_01')
(<Segment(15.6066, 16.7203)>, 'E', 'SPEAKER_00')
```

Gambar 3.21 Hasil Speaker Diarization

Luaran pada Gambar 3.21 tersebut adalah hasil dari proses diarization audio yang mencakup informasi tentang durasi segmen audio dan identifikasi pembicara. Setiap baris output menunjukkan rentang waktu segmen audio, label untuk urutan pembicara, dan identifikasi unik untuk setiap pembicara dalam rekaman rapat atau pertemuan. Dengan informasi ini, dapat dipahami siapa yang berbicara pada waktu tertentu dalam rekaman audio, memfasilitasi analisis dan dokumentasi percakapan dengan lebih terstruktur.

Berdasarkan hasil dari *speaker diarization* diatas, dilakukan proses *grouping* untuk mengelompokkan segmen-segmen audio yang teridentifikasi ke dalam kelompok-kelompok

atau kluster berdasarkan pembicara yang sama. Dengan kata lain, setelah setiap segmen audio dikaitkan dengan pembicara tertentu (seperti yang ditunjukkan dalam hasil *diarization*), proses grouping akan mengumpulkan atau mengelompokkan segmen-segmen tersebut yang berasal dari pembicara yang sama ke dalam satu kelompok atau cluster. Tujuan utama dari proses grouping ini adalah untuk menyatukan segmen-segmen yang dikenali sebagai pembicara yang sama secara konsisten dalam satu kesatuan. Hal ini memungkinkan untuk lebih mudah menganalisis percakapan antara pembicara yang berbeda-beda dalam rekaman audio, serta menyediakan landasan untuk langkah-langkah berikutnya seperti analisis lebih lanjut, transkripsi, atau ekstraksi informasi yang lebih detail dari percakapan tersebut. Setelah itu dilakukan proses pemisahan audio berdasarkan *grouping* segmen audio pembicara, berikut adalah kutipan kode untuk melakukan proses tersebut.

```
audio = AudioSegment.from_wav("input_prep.wav")
gidx = -1
for g in groups:
  start = re.findall('[0-9]+:[0-9]+:[0-9]+\.[0-9]+', string=g[0])[0]
  end = re.findall('[0-9]+:[0-9]+:[0-9]+\.[0-9]+', string=g[-1])[1]
  start = millisec(start) end = millisec(end) gidx += 1
  audio[start:end].export(str(gidx) + '.wav', format='wav')
```

Gambar 3.22 Kutipan Kode Pemisahan Audio Berdasarkan Grouping Segmen

Kode pada Gambar 3.22 ini membaca file audio ‘Input_prep.wav’ menggunakan ‘AudioSegment’, lalu mengiterasi melalui setiap grup dalam groups. Setiap grup diekstrak dengan waktu mulai dan akhirnya dari segmen pertama dan terakhir dalam grup, kemudian diekspor ke file audio terpisah dengan nama berdasarkan indeks grup (gidx).

3.2.4 Integrasi Model Whisper untuk Transkripsi Audio Rapat

Integrasi Model Whisper dalam konteks transkripsi audio rapat merupakan langkah penting dalam memperbaiki efisiensi dan akurasi proses dokumentasi rapat menggunakan teknologi terbaru dalam pemrosesan bahasa alami. Model Whisper, yang dikembangkan oleh OpenAI, merupakan salah satu model terdepan dalam bidang ini, mampu menghasilkan transkripsi teks dengan tingkat keakuratan yang tinggi (Lyu et al., 2024). Berikut kutipan kode untuk melakukan transkripsi menggunakan model Whisper:

```
!pip install git+https://github.com/openai/whisper.git
import whisper
import json
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = whisper.load_model('large', device = device)

for i in range(len(groups)):
  audiof = str(i) + '.wav'
```

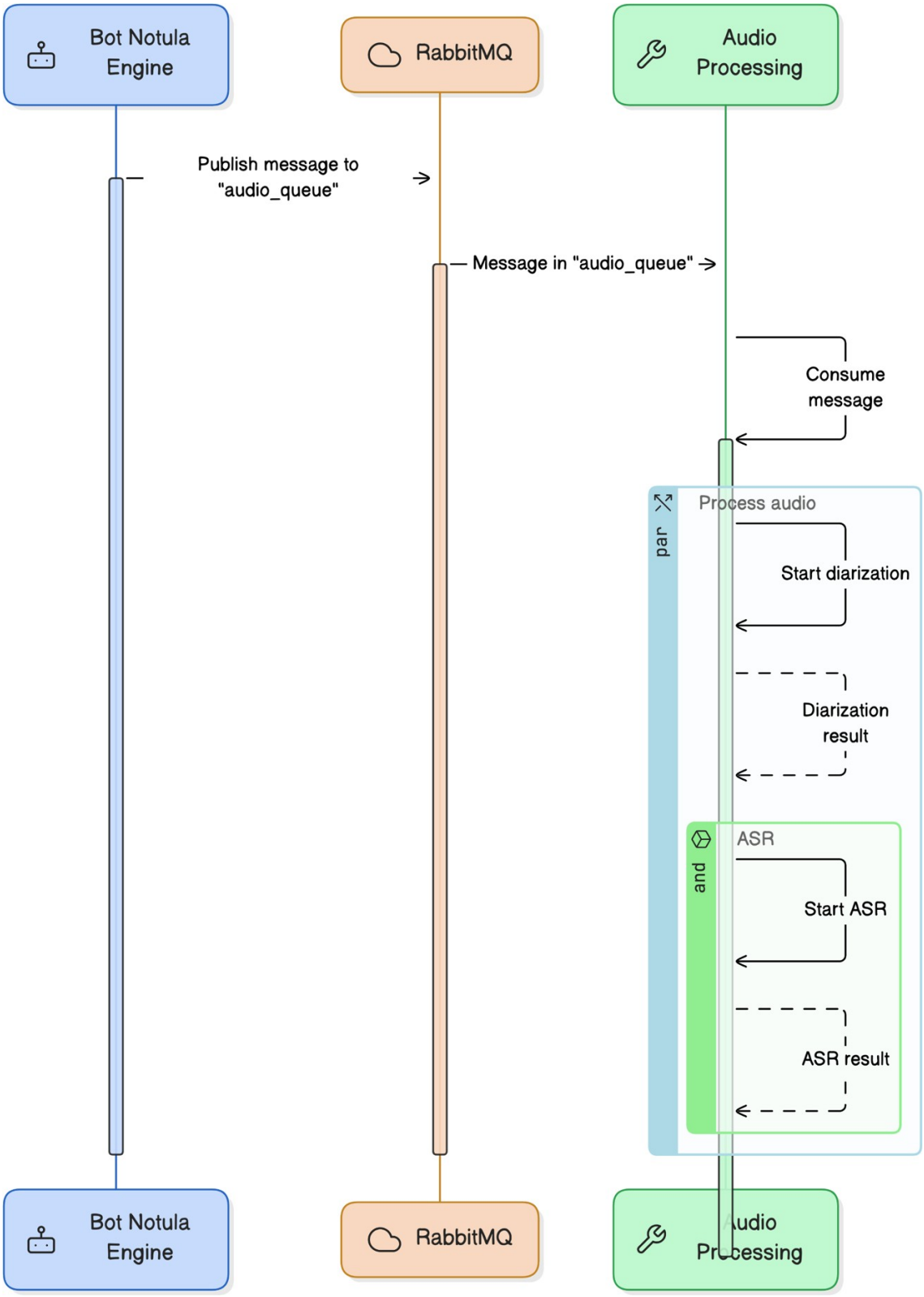
```
  result = model.transcribe(audio=audiof, language='id',
word_timestamps=True)

  with open(str(i)+'.json', "w") as outfile:
    json.dump(result, outfile, indent=4)
```

Gambar 3.23 Kutipan Kode Transkipsi Menggunaan Whisper

Kode pada Gambar 3.23 di atas mengilustrasikan langkah-langkah untuk menggunakan paket Whisper dari OpenAI dalam melakukan transkripsi otomatis terhadap rekaman audio. Pertama, langkah instalasi dilakukan dengan perintah '!pip install git+https://github.com/openai/whisper.git', yang menginstall paket Whisper dari repositori GitHub. Kemudian, kode menginisialisasi penggunaan Whisper dengan memuat model transkripsi besar ('large') ke dalam memori, mengoptimalkan penggunaan perangkat keras GPU jika tersedia, atau CPU sebagai alternatif.

Setelah model dimuat, proses transkripsi dilakukan secara iteratif terhadap setiap grup hasil *diarization* yang telah dipersiapkan sebelumnya. Setiap file audio yang diidentifikasi dengan nama berurutan (misalnya, '0.wav', '1.wav', dan seterusnya) diproses dengan menggunakan model Whisper. Hasil transkripsi untuk setiap file audio disimpan dalam format JSON yang terstruktur, dengan indentasi yang jelas.

3.2.5 Integrasi RabbitMQ untuk Komunikasi Antar *Service*



Gambar 3.24 Sequence Diagram Komunikasi Service Bot Notula Engine dan Audio Processing Menggunakan RabbitMQ

Integrasi RabbitMQ digunakan sebagai solusi komunikasi antara dua *service* utama dalam sistem bot notula untuk Google Meet. RabbitMQ berperan penting dalam menyediakan jalur komunikasi yang andal dan terstruktur antara *service* Bot Notula Engine dan *service* Audio Processing. Selain itu, dengan mengimplementasikan sistem bot notula menggunakan dua *service* terpisah, yaitu Bot Notula Engine dan Audio Processing sistem menjadi lebih *scalable* atau dapat dikembangkan secara lebih fleksibel dan efisien. Skalabilitas ini tercapai karena masing-masing service dapat diatur dan dikelola secara independen, memungkinkan peningkatan kapasitas dan kinerja secara terpisah sesuai kebutuhan.

Gambar 3.24 menggambarkan alur kerja pengolahan audio dalam sistem Bot Notula yang menggunakan RabbitMQ sebagai penghubung antar *service*. Proses ini dimulai dari Bot Notula Engine yang mempublikasikan pesan ke ‘audio_queue’ di RabbitMQ. Setelah pesan berada di ‘audio_queue’, *service* Audio Processing akan mengonsumsi pesan tersebut untuk memulai proses pengolahan audio. Pengolahan audio dimulai dengan memproses pesan yang diterima, di mana tahap pertama adalah melakukan diarization (pemisahan pembicara). Hasil dari proses *diarization* ini kemudian digunakan dalam tahap selanjutnya. Secara paralel, sistem juga melakukan Automatic Speech Recognition (ASR) untuk mengubah audio menjadi teks. Hasil dari kedua proses ini, *diarization* dan ASR, dikombinasikan untuk menghasilkan keluaran akhir yang dapat digunakan oleh Bot Notula Engine. Dengan menggunakan RabbitMQ, sistem ini memastikan bahwa pesan-pesan yang diproses dapat dikelola dengan baik, dan setiap komponen dapat bekerja secara efektif tanpa harus saling tergantung langsung. Berikut adalah kutipan kode implementasi RabbitMQ untuk mengirim pesan ke *queue* tertentu.

```
async sendAudioPathToRabbitMQ(audioPath: string, requestId: string):
Promise<void> {
    const queue = 'audio_queue';



    amqp.connect('amqp://localhost', (error0, connection) => {
      if (error0) {
        throw error0;
      }
      connection.createChannel((error1, channel) => {
        if (error1) {
          throw error1;
        }

        channel.assertQueue(queue, {
          durable: false,
        });

        const message = {
          audioPath: audioPath,
          requestId: requestId,
        };

        channel.sendToQueue(queue,
Buffer.from(JSON.stringify(message)));

        console.log(`[Bot Notula Engine] Sent audio path ${audioPath}
to RabbitMQ`);

        setTimeout(() => {
          connection.close();
        }, 500);
      });
    });
  }
  return go(f, seed, [])
}
```

Gambar 3.25 Kutipan Kode Implementasi RabbitMQ di *Service* Bot Notula Engine

Pada Gambar 3.25 terdapat fungsi ‘sendAudioPathToRabbitMQ’ yang bertanggung jawab untuk mengirimkan informasi mengenai path *file* audio dan request_id ke RabbitMQ setelah proses perekaman audio selesai. Penggunaan AMQP (Advanced Message Queuing Protocol) dalam ‘amqp.connect’ dan ‘connection.createChannel’ memungkinkan pembuatan koneksi dan pengiriman pesan secara efisien ke antrian 'audio_queue'. Pesan yang dikirim dalam format JSON mengandung informasi penting yang akan diproses lebih lanjut oleh *service* berikutnya.

Service Audio Processing dibangun menggunakan Python dan pustaka pika untuk mengonsumsi pesan dari RabbitMQ. *Service* ini bertugas untuk memproses audio yang telah direkam oleh Bot Notula Engine, termasuk pengunduhan audio, *diarization*, dan transkripsi

berdasarkan informasi yang diterima dari RabbitMQ. Berikut implementasi RabbitMQ untuk menerima pesan dari antiran (*queue)* ‘audio_queue’ yang sebelumnya dikirimkan oleh *service* Bot Notula Engine.

```
import pika
import json

class Consumer:
    def__init__(self):
        # Membuat koneksi RabbitMQ
        self.rabbitmq_connection = pika.BlockingConnection(
               pika.ConnectionParameters('localhost')
         )
        self.channel = self.rabbitmq_connection.channel()
        self.channel.queue_declare(queue='audio_queue')

    def callback(self, ch, method, properties, body):
        # Menangani pesan yang diterima dari RabbitMQ
        message = json.loads(body)
        audio_path = message['audioPath']
        request_id = message['requestId']
        # Proses pemrosesan audio berdasarkan audioPath dan requestId
        # ...
        # (acknowledge) bahwa pesan telah diproses
        ch.basic_ack(delivery_tag=method.delivery_tag)

    def start_consuming(self):
        # Memulai konsumsi pesan dari antrian 'audio_queue'
        self.channel.basic_consume(queue='audio_queue',
            on_message_callback=self.callback, auto_ack=False)
        self.channel.start_consuming()

if__name__   == '__main__':
    # Memulai instance dari Consumer
    audio_processing_service = Consumer()
    audio_processing_service.start_consuming()
```

Gambar 3.26 Kutipan Kode Implementasi RabbitMQ di *Service* Audio Processing

Pada Gambar 3.26 terlihat bahwa *service* Audio Processing diimplementasikan dalam Python menggunakan pustaka pika untuk mengonsumsi pesan dari RabbitMQ. Fungsi ‘callback’ digunakan untuk menangani pesan yang diterima dari *queue* 'audio_queue'. Saat sebuah pesan diterima, informasi audioPath dan requestId diekstrak dari pesan JSON menggunakan ‘json.loads(body)’. Informasi ini selanjutnya digunakan untuk menginisiasi dan melaksanakan proses pemrosesan audio lanjutan seperti pengunduhan audio, *diarization*, dan transkripsi. Fungsi ‘ch.basic_ack(delivery_tag=method.delivery_tag)’ memastikan pesan berhasil diproses dan dihapus dari antrian, menunjukkan penyelesaian tugas dan kesiapan untuk menerima pesan berikutnya.

3.2.6 Pengujian Akurasi Transkrip Rapat Google Meet

Evaluasi akurasi transkripsi dilakukan untuk menilai kinerja bot notula yang dibangun dengan model Whisper pada platform Google Meet. Metode pengujian yang digunakan adalah Word Error Rate (WER), yang merupakan metrik standar dalam penilaian sistem pengenalan suara. WER menghitung kesalahan transkripsi dengan mempertimbangkan substitusi, penghapusan, dan penyisipan kata (Morris et al., 2004).

Penelitian yang dilakukan oleh Morris et al. (2004) menjadi referensi utama dalam pengembangan pustaka jiwer, yang bertujuan memudahkan perhitungan WER. Pustaka jiwer ini mencakup proses tokenisasi teks serta perhitungan substitusi, penghapusan, dan penyisipan kata. Rumus yang digunakan untuk menentukan WER juga dijelaskan dalam penelitian tersebut sebagai berikut:

$$WER = \frac{S + D + I}{N}$$

Keterangan:

- S adalah jumlah substitusi (kata yang salah ditranskripsikan).
- D adalah jumlah penghapusan (kata yang hilang dalam transkripsi).
- I adalah jumlah penyisipan (kata tambahan dalam transkripsi).
- N adalah jumlah total kata dalam teks referensi (*ground truth*).

Sedangkan untuk mengetahui nilai presentase akurasi transkripsi dapat dihitung dengan rumus berikut:

$$Presentase\ akurasi = (1 - WER) \times 100\ \%$$

Selama proses pengembangan, pengujian terbatas dilakukan dengan menggunakan bot notula pada empat rapat singkat yang memiliki variasi kualitas audio, yaitu Tinggi, Sedang, dan Rendah. Berikut adalah kriteria nya:

- Kualitas Audio Tinggi: Audio jernih dengan sedikit atau tanpa gangguan latar belakang, serta *volume* yang konsisten.
- Kualitas Audio Sedang: Audio masih dapat dipahami, namun terdapat sedikit ketidakjelasan pada pengucapan kata atau terdapat kebisingan latar belakang yang tidak menghalangi pemahaman.
- Kualitas Audio Rendah: Audio memiliki banyak gangguan, seperti kebisingan latar belakang yang tinggi, distorsi, atau *volume* yang tidak konsisten, sehingga sulit dipahami.

Tabel 3.1 Pengujian Akurasi Bot Notula dengan Metode Ground Truth

No	Ground Truth	Transkripsi Bot	Kualitas Audio	WER	Akurasi (%)
1	[00:00:000.48 - 00:00:002.40] Speaker 1: guys liburan nanti mau kemana nih? [00:00:004.19 - 00:00:005.79] Speaker 2: Gimana kalau ke Bali? Pantainya seru banget. [00:00:008.38 - 00:00:010.23] Speaker 1: Setuju sih, tapi kita harus cari tempat [00:00:010.23- 00:00:010.96] Speaker 1: Penginapan yang murah [00:00:012.37 - 00:00:014.15] Speaker 2: Tenang, aku punya list tempat keren. [00:00:014.15 - 00:00:014.83] Speaker 2: Kapan mau booking?	[00:00:000.48 - 00:00:002.40] Speaker 1: guys liburan nanti mau kemana nih [00:00:004.19 - 00:00:005.79] Speaker 2: Gimana kalau ke Bali? Pantainya seru banget. [00:00:008.38 - 00:00:010.23] Speaker 1: Tujuh sih, tapi kita harus cari tempat [00:00:010.23- 00:00:010.96] Speaker 1: Minat yang murah [00:00:012.37 - 00:00:014.15] Speaker 2: Tenang, aku punya list tempat keren. [00:00:014.15 - 00:00:014.83] Speaker 2: Kapan mau booking?	Tinggi	0.06	93.75%
2	[00:00:000.11 - 00:00:002.23] Speaker 1: acara kampus, tinggal seminggu lagi, siap gak? [00:00:003.67 - 00:00:005.47]	[00:00:000.11 - 00:00:002.23] Speaker 1: acara kampus, tinggal seminggu lagi, siap gak [00:00:003.67 - 00:00:005.47]	Tinggi	0.06	93.75%

No	Ground Truth	Transkripsi Bot	Kualitas Audio	WER	Akurasi (%)
	Speaker 2: Siap dong, siapa yang handle dekorasinya. [00:00:006.74 - 00:00:009.52] Speaker 1: Aku boleh sih bantuin buat konsep yang keren. [00:00:012.74 - 00:00:015.54] Speaker 2: boleh banget aku bawa speaker ya biar makin asik. [00:00:016.25 - 00:00:016.73] Speaker 1: Oke, siap.	Speaker 2: Siap dulu, siapa yang handle dekorasinya. [00:00:006.74 - 00:00:009.52] Speaker 1: Aku boleh sih bantuin buat konsep yang keren. [00:00:012.74 - 00:00:015.54] Speaker 2: boleh banget aku bawa speaker ya biar makin asik [00:00:016.25 - 00:00:016.73] Speaker 1: Oke, siap.			
3	[00:00:000.04 - 00:00:002.18] Speaker 1: Mau belajar bareng buat ujian minggu depan? [00:00:004.14 - 00:00:005.82] Speaker 2: boleh, tapi materinya apa dulu nih [00:00:009.08 - 00:00:011.52] Speaker 1: mulai dari bab terakhir banyak yang susah [00:00:012.98 - 00:00:015.94] Speaker 2: Setuju sih, biar gak pusing.	[00:00:000.04 - 00:00:002.18] Speaker 1: Mau belajar bareng buat ujian minggu depan? [00:00:004.14 - 00:00:005.82] Speaker 2: boleh, tapi materinya apa dulu nih [00:00:009.08 - 00:00:011.52] Speaker 1: mulai dari web terakhir banyak yang susah [00:00:012.98 - 00:00:015.94] Speaker 2: Setuju sih, biar gak pusing.	Sedang	0.13	87.10%

No	Ground Truth	Transkripsi Bot	Kualitas Audio	WER	Akurasi (%)
	Kita sambil makan pizza yuk. [00:00:017.59 - 00:00:017.69] Speaker 1: Oke.	Kita sambil makan pizza yuk. [00:00:017.59 - 00:00:017.69] Speaker 1: Terima kasih.			
4	[00:00:000.00 - 00:00:000.90] Speaker 1: jogging bareng nanti? [00:00:002.28 - 00:00:004.80] Speaker 2: Sabtu pagi boleh sih, gimana? Biar sekalian seger. [00:00:007.28 - 00:00:007.64] Speaker 1: Bisa sih. [00:00:008.31 - 00:00:009.59] Speaker 1: habis jogging kita sarapan bareng ya [00:00:010.97 - 00:00:013.43] Speaker 2: Boleh, sarapan tempat biasa ya, kopinya enak.	[00:00:000.00 - 00:00:000.90] Speaker 1: Kita jadiin baru nanti. [00:00:002.28 - 00:00:004.80] Speaker 2: Sabtu pagi boleh sih, gimana? Biar sekalian seger. [00:00:007.28 - 00:00:007.64] Speaker 1: Biasa sih. [00:00:008.31 - 00:00:009.59] Speaker 1: habis jogging kita sarapan bareng [00:00:010.97 - 00:00:013.43] Speaker 2: Boleh, sarapan tempat biasa ya, kopinya enak.	Rendah	0.23	76.92%

Berdasarkan hasil penelitian yang tercantum pada Tabel 3.1, bot notula menunjukkan performa optimal pada kualitas audio tinggi, dengan tingkat kesalahan kata (WER) yang rendah dan akurasi mencapai 93,75%. Pada kualitas audio sedang, performa bot tetap memadai dengan akurasi sebesar 87,10%. Namun, pada kualitas audio rendah, bot mengalami peningkatan tingkat kesalahan dan penurunan akurasi menjadi 76,92%. Kualitas audio terbukti sangat

mempengaruhi akurasi transkripsi bot, di mana kualitas audio tinggi menghasilkan transkripsi yang lebih akurat dibandingkan dengan kualitas audio rendah.

3.3 Penutupan Proyek

Pada saat penulis menyelesaikan masa magang, proyek bot notula Google Meet telah mencapai tahap yang signifikan dalam pengembangannya. Bot ini telah mampu melakukan pencatatan rapat secara otomatis dengan memanfaatkan model Whisper untuk *diarization* dan transkripsi, serta menggunakan Puppeteer untuk otomatisasi interaksi dengan platform Google Meet. Sistem ini juga sudah terintegrasi dengan RabbitMQ untuk komunikasi antar layanan dan Google Cloud Storage untuk penyimpanan file audio.

Namun, proyek ini masih berada pada tahap pengembangan lebih lanjut. Ketika penulis selesai magang, proyek Bot notula Google Meet tengah dalam proses integrasi dengan aplikasi induk, Widya Notulensi. Aplikasi ini merupakan platform pencatatan rapat otomatis yang mendukung berbagai platform meeting online seperti Zoom, Google Meet, dan Microsoft Teams. Integrasi ini bertujuan untuk memungkinkan akses langsung ke bot notula Google Meet dari aplikasi Widya Notulensi, sehingga memudahkan pengguna dalam mengelola dan mengakses hasil notula rapat dari berbagai platform dalam satu aplikasi terintegrasi.

Selain itu, pengembangan proyek ini juga mencakup penambahan fitur-fitur baru untuk meningkatkan fungsionalitas dan nilai tambah dari bot notula. Salah satu fitur yang sedang dikembangkan adalah fitur rangkuman rapat, yang memungkinkan pengguna mendapatkan ringkasan singkat dari poin-poin penting yang dibahas selama rapat. Fitur ini bertujuan untuk membantu pengguna memahami inti pembicaraan dengan lebih cepat dan efisien.
Fitur lainnya adalah *wordcloud*, yang akan menampilkan visualisasi kata-kata kunci yang sering muncul selama rapat. Fitur ini dirancang untuk memberikan gambaran cepat tentang topik utama yang dibahas, serta membantu pengguna dalam menganalisis tren atau isu yang sering muncul dalam rapat.

Dengan berbagai pengembangan ini, proyek bot notula Google Meet diharapkan dapat memberikan kontribusi yang signifikan dalam meningkatkan efisiensi dan akurasi pencatatan rapat, serta memberikan solusi yang lebih komprehensif dan *user-friendly* melalui aplikasi Widya Notulensi. Penulis merasa bangga telah menjadi bagian dari pengembangan proyek ini dan berharap hasil karyanya dapat memberikan manfaat yang nyata bagi pengguna di masa mendatang.

BAB IV
REFLEKSI PELAKSANAAN MAGANG

4.1 Relevansi Akademik

Selama magang di Widya Robotics, berbagai teori yang diuraikan di Bab 2 terkait pengembangan bot notula diterapkan menggunakan teknologi seperti Puppeteer, Google Meet, Whisper, dan *Speaker Diarization*. Metode System Development Life Cycle (SDLC) prototyping yang telah dipelajari di mata kuliah Rekayasa Perangkat Lunak digunakan dalam pengembangan bot notula ini. Metode ini berfokus pada pembuatan prototipe awal dari produk untuk mendapatkan umpan balik cepat dari pengguna, dengan beberapa iterasi untuk menyempurnakan prototipe berdasarkan masukan yang diterima (Firmansyah et al., 2021). Dalam pelaksanaannya, metode ini diterapkan pada proyek pengembangan bot notula dengan integrasi model Whisper pada platform Google Meet. Proses pengembangan dimulai dengan pembuatan prototipe sederhana yang menunjukkan fungsi dasar bot, seperti simulasi masuk ke dalam meeting, perekaman audio, dan penyimpanan file audio di Google Cloud Storage. Prototipe ini kemudian diuji oleh tim, dan umpan balik yang diperoleh digunakan untuk menyempurnakan fitur dan performa bot. Namun, dalam implementasi prototyping, ditemukan beberapa gap antara teori dan praktik, seperti kebutuhan untuk penyesuaian cepat terhadap perubahan antarmuka pengguna di Google Meet dan batasan waktu serta sumber daya yang sering kali tidak ideal. Hal ini menunjukkan pentingnya fleksibilitas dan adaptasi terhadap kondisi spesifik proyek, serta pengujian berulang dan responsif terhadap umpan balik pengguna dalam pengembangan produk teknologi.

Pada mata kuliah Natural Language Processing (NLP), tahap-tahap penting dalam pengolahan data mentah, termasuk *cleansing*, *pre-processing*, dan transformasi data dipelajari. Selama magang, data audio yang direkam dari Google Meet harus melalui beberapa tahap pre-processing sebelum dapat ditranskripsi menggunakan model Whisper. Proses ini melibatkan pembersihan *noise*, normalisasi *volume*, dan segmentasi audio untuk memudahkan proses transkripsi. Penggunaan Whisper untuk transkripsi audio membuktikan relevansi teorinya dengan praktik lapangan. Whisper, yang memanfaatkan model Convolutional Transformer End-to-End ASR, terbukti efektif dalam mengolah data audio multibahasa dengan akurasi tinggi. Meskipun demikian, penanganan berbagai jenis aksen dan kualitas audio yang bervariasi kadang mempengaruhi hasil transkripsi. Hal ini menunjukkan bahwa meskipun teknologi ASR telah berkembang pesat, faktor-faktor lingkungan tetap mempengaruhi

kinerjanya dan perlu diperhatikan dalam implementasi praktis. Dalam *pre-processing* data, teori sering kali menggambarkan proses yang lebih ideal dan bersih, sementara dalam praktik, data audio yang diterima sering kali memiliki banyak *noise* dan variasi kualitas yang memerlukan teknik *pre-processing* lebih kompleks. Pengalaman ini menunjukkan betapa pentingnya proses *pre-processing* yang efektif dalam meningkatkan akurasi dan performa model transkripsi.

Teknik *Speaker Diarization* menggunakan pyannote.audio juga diterapkan untuk mengidentifikasi siapa yang berbicara dan kapan. Teori ini menjelaskan bahwa *diarization* dapat meningkatkan akurasi transkripsi dengan memisahkan suara dari pembicara yang berbeda. Dalam praktiknya, meskipun Pyannote.audio dapat melakukan *diarization* dengan baik, terdapat tantangan dalam hal integrasi dengan sistem yang ada dan penyesuaian model untuk kondisi spesifik seperti rapat virtual dengan banyak peserta. Pengujian dan penyesuaian berulang sangat penting untuk memastikan keakuratan dan keandalan dalam penggunaan praktis. Selain itu, teori tentang penggunaan DOM, Node, dan Mutation Observer dalam JavaScript diterapkan untuk mendeteksi perubahan dalam elemen HTML saat peserta rapat berbicara. Ini memungkinkan bot untuk secara *real-time* mencatat siapa yang berbicara tanpa memerlukan intervensi manual. Dalam implementasinya, Mutation Observer sangat efektif dalam mendeteksi perubahan, tetapi perlu dikombinasikan dengan strategi penanganan data yang tepat untuk memastikan integritas dan keakuratan informasi yang dicatat. Kombinasi teknologi yang tepat sangat penting untuk memastikan integritas dan keakuratan data dalam implementasi praktis.

Di mata kuliah Manajemen Diri, pentingnya pengaturan waktu dan komunikasi yang efektif dalam menyelesaikan tugas dan bekerja dalam tim dipelajari. Keterampilan ini sangat penting untuk memastikan proyek berjalan lancar dan target tercapai tepat waktu. Selama magang, pengaturan waktu dan komunikasi yang baik sangat diperlukan untuk koordinasi dengan tim pengembang dan *stakeholder* proyek. Setiap iterasi prototyping membutuhkan umpan balik cepat, yang mengharuskan pengaturan waktu yang baik antara pengembangan, testing, dan evaluasi. Selain itu, komunikasi yang jelas dan efektif dengan rekan kerja memastikan setiap perubahan dan pembaruan pada prototipe dapat diterapkan dengan tepat. Dalam manajemen waktu dan komunikasi, teori menggambarkan keterampilan ini sebagai sesuatu yang dapat dipelajari dan diterapkan dengan konsisten, namun dalam situasi nyata, tekanan waktu dan dinamika tim dapat mempengaruhi efektivitas pengaturan waktu dan komunikasi.

Dari pengalaman ini, dapat diambil beberapa rekomendasi dan *insight* yang berharga. Pertama, fleksibilitas dalam mengikuti umpan balik pengguna dan kemampuan untuk cepat beradaptasi dengan perubahan kebutuhan adalah kunci keberhasilan dalam metode *prototyping*. Kedua, pentingnya memahami karakteristik data mentah dan menggunakan teknik *pre-processing* yang sesuai untuk mengatasi berbagai jenis *noise* dan variasi kualitas. Ketiga, mengembangkan strategi manajemen waktu yang lebih fleksibel dan meningkatkan keterampilan komunikasi antar tim untuk mengatasi tantangan dinamis di lapangan. Dengan memahami dan mengatasi gap antara teori dan pelaksanaan di lapangan, pengalaman magang ini memberikan kontribusi akademik yang berharga dan menyediakan rekomendasi praktis untuk pengembangan lebih lanjut di bidang rekayasa perangkat lunak dan NLP.

4.2 Pembelajaran Magang

Magang merupakan tahap penting dalam perjalanan pendidikan seorang mahasiswa, yang bertujuan untuk meningkatkan keterampilan yang dibutuhkan dalam dunia profesional. Pengalaman ini tidak hanya berperan sebagai sarana pembelajaran, tetapi juga sebagai persiapan komprehensif dalam menghadapi berbagai tantangan di masa depan. Melalui magang, mahasiswa dapat menerapkan teori yang telah dipelajari di bangku kuliah dalam konteks nyata, sehingga memperdalam pemahaman dan meningkatkan keterampilan praktis.

Selama pelaksanaan magang, penulis memperoleh banyak manfaat, baik dalam pengembangan keterampilan teknis maupun non-teknis. Keterampilan teknis, seperti penggunaan alat dan teknologi terbaru, analisis data, dan pemrograman, diasah dan ditingkatkan melalui tugas-tugas yang diberikan. Di sisi lain, keterampilan non-teknis seperti manajemen waktu, komunikasi efektif, dan kerjasama tim juga mengalami peningkatan yang signifikan. Pengalaman ini memberikan penulis wawasan berharga tentang dinamika kerja di dunia industri dan memperkuat kemampuan adaptasi dalam lingkungan yang terus berubah. Melalui integrasi antara pembelajaran teoritis dan praktik lapangan, magang membantu mempersiapkan mahasiswa untuk menjadi profesional yang kompeten dan siap menghadapi tantangan karir di masa depan.

4.2.1 Keterampilan Teknis

Dalam pengembangan bot notula di Widya Robotics, keterampilan teknis yang diperoleh mencakup berbagai aspek teknologi yang kompleks. Bot notula dikembangkan dengan dua layanan terpisah: Bot Notula Engine dan Audio Processing, dengan tujuan utama untuk mencapai skalabilitas. Pendekatan ini memungkinkan penyesuaian skala masing-masing layanan secara independen. Sebagai contoh, jika aplikasi pemrosesan audio membutuhkan

lebih banyak sumber daya karena intensitas pemrosesan audio, kapasitas layanan ini dapat disesuaikan tanpa mempengaruhi kinerja bot.

Sebelumnya, fokus penulis lebih kepada aplikasi sederhana tanpa menggunakan teknologi yang lebih kompleks seperti *message broker* atau model *machine learning*. Penggunaan RabbitMQ sebagai *message broker* merupakan pengalaman baru, dimana RabbitMQ membantu menghubungkan layanan-layanan dalam arsitektur *microservice* dan mengelola pesan antara *publisher* dan *consumer* menggunakan protokol AMQP. RabbitMQ dipilih karena instalasi dan penggunaannya yang relatif mudah, yang mengurangi beban dan waktu pengiriman pesan serta meningkatkan efisiensi dan keandalan sistem.

Selain itu, integrasi Puppeteer dan Whisper dari OpenAI pada platform Google Meet juga merupakan penerapan teknologi baru. Puppeteer digunakan untuk mensimulasikan masuk ke dalam meeting sebagai peserta dan merekam audio menggunakan elemen HTML audio pada halaman Google Meet. Setelah perekaman, *file* audio diunggah ke Google Cloud Storage. Model Whisper digunakan untuk melakukan transkripsi audio menjadi teks, yang meningkatkan efisiensi dan akurasi pencatatan rapat serta mengurangi kesalahan manusia.

Pengalaman baru lainnya adalah penggunaan Pyaannote untuk *speaker diarization*, dengan tujuan memisahkan suara pembicara yang berbeda dalam rekaman audio. Teknik ini memungkinkan identifikasi dan pemisahan segmen-segmen yang diucapkan oleh pembicara yang berbeda, sehingga meningkatkan akurasi transkripsi dan analisis lebih lanjut.

Implementasi Google Cloud Storage dalam proyek ini memberikan pengalaman dalam pengaturan bucket, pengunggahan file audio, dan pengelolaan izin akses untuk memastikan keamanan data. Google Cloud Storage memungkinkan penyimpanan data yang skalabel dan mudah diakses dari berbagai lokasi.

Pengalaman dalam proyek ini memperkaya keterampilan teknis dalam pengembangan sistem berbasis *microservices*, integrasi *message broker*, penggunaan model *machine learning* untuk transkripsi audio, serta pemrosesan dan penyimpanan data audio di cloud. Pengalaman memberikan pengalaman dan peningkatan yang signifikan dan memberikan sudut pandang baru dalam mengembangkan aplikasi.

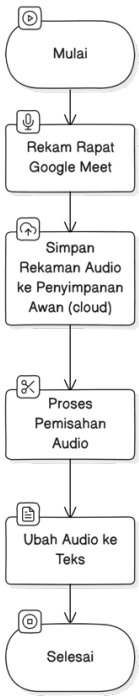
4.2.2 Keterampilan Non Teknis

Selain keterampilan teknis, pengalaman magang di Widya Robotics juga memperkaya keterampilan non-teknis yang sangat penting dalam dunia kerja. Selama magang, terdapat kesempatan untuk belajar berkomunikasi secara efektif dengan tim, *stakeholder*, dan pengguna akhir untuk mengumpulkan kebutuhan, memberikan laporan perkembangan, dan mengelola

umpan balik. Keterampilan manajemen waktu juga meningkat karena harus mengatur prioritas tugas dan mengelola waktu untuk memenuhi tenggat waktu proyek.

Kerja sama tim menjadi keterampilan penting lainnya, di mana terdapat pembelajaran berkolaborasi dalam tim pengembangan perangkat lunak, bekerja sama dalam menyelesaikan masalah, dan berbagi pengetahuan dengan anggota tim lain. Keterampilan pemecahan masalah juga sangat penting, karena analisis terhadap masalah yang muncul selama pengembangan perangkat lunak diperlukan untuk menemukan solusi yang efektif. Selain itu, pengalaman memimpin sub-proyek kecil atau tugas tertentu dalam tim pengembangan perangkat lunak memperkaya keterampilan kepemimpinan. Secara keseluruhan, pengalaman magang di Widya Robotics tidak hanya memperkaya keterampilan teknis dan non-teknis yang relevan dengan bidang studi informatika, tetapi juga memberikan wawasan yang lebih dalam mengenai praktik kerja di industri teknologi.

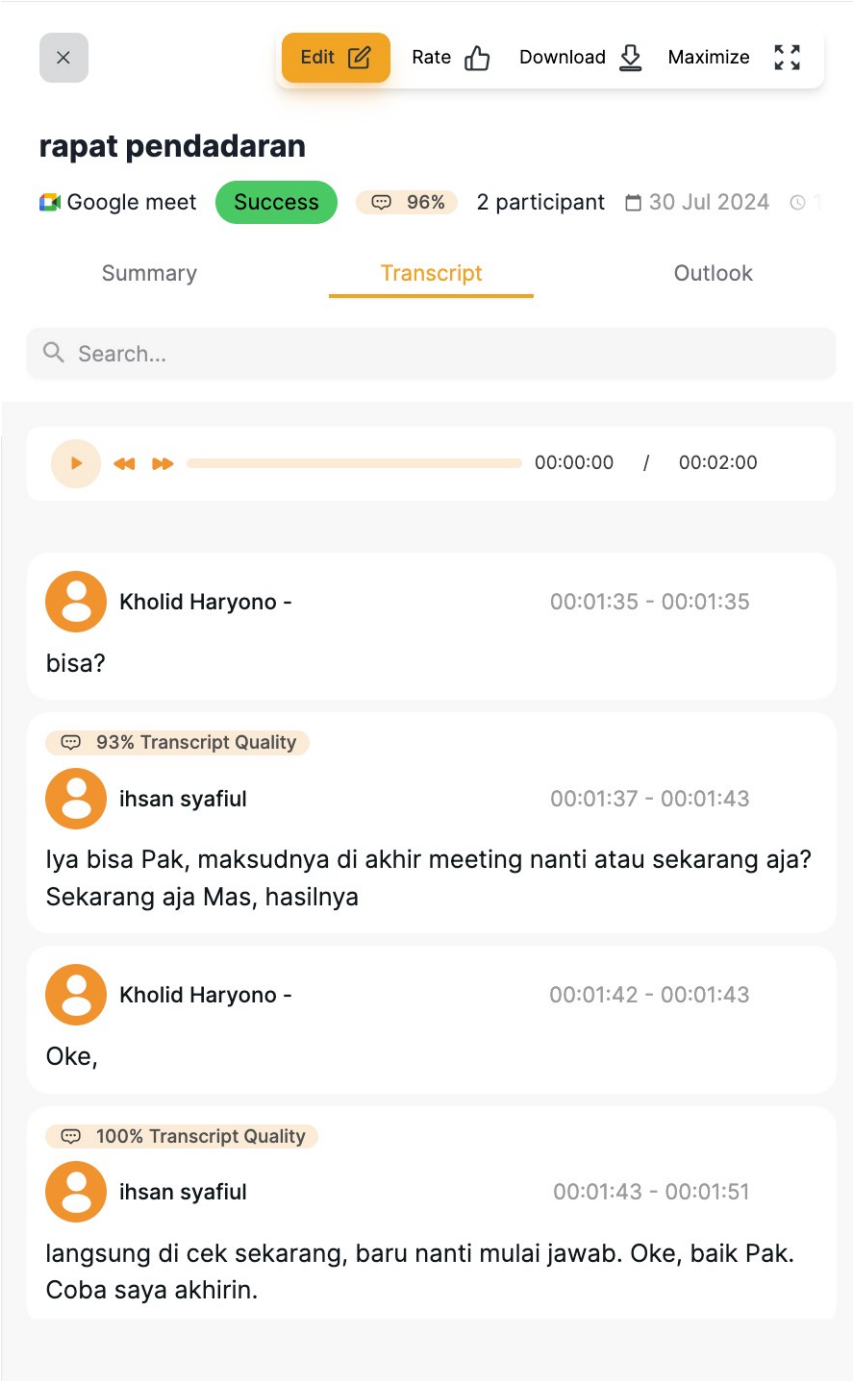
4.3 Ikhtisar Pengembangan Bot Notula Google Meet



Gambar 4.1 Diagram Alir Sederhana Bot Notula Google Meet

Bot Notula Google Meet adalah sebuah aplikasi tambahan yang dapat melakukan pencatatan rapat pada Google Meet secara otomatis. Gambar 4.1 menunjukkan diagram sederhana untuk menjelaskan alur kerja Bot Notula Google Meet secara sederhana. Bot merekam audio dari rapat Google Meet. Setelah itu, rekaman audio disimpan ke dalam penyimpanan *cloud*. Rekaman ini kemudian diproses untuk memisahkan audio berdasarkan

pembicara dan waktu percakapan. Setelah pemisahan, audio diubah menjadi teks. Proses ini memastikan bahwa semua percakapan dalam rapat dicatat secara otomatis dan akurat tanpa memerlukan pencatatan manual. Berikut adalah contoh hasil pencatatan rapat dari Bot Notula Google Meet pada Gambar 4.2.



Gambar 4.2 Contoh Hasil Pencatatan Rapat Google Meet

BAB V
PENUTUP

5.1 Kesimpulan

Tugas akhir ini menunjukkan bahwa pengembangan bot notula pada platform Google Meet dengan integrasi Puppeteer dan Whisper OpenAI berhasil mencapai tujuan utama, yaitu meningkatkan efisiensi pencatatan notula rapat secara otomatis. Sistem yang dikembangkan mampu menangkap dan mentranskrip percakapan selama rapat dengan tingkat akurasi yang cukup baik, terutama pada kualitas audio yang tinggi. Hal ini terbukti dari hasil pengujian yang menunjukkan tingkat kesalahan kata (Word Error Rate/WER) rendah dan akurasi tinggi pada kualitas audio yang baik.

Penggunaan Whisper OpenAI sebagai teknologi transkripsi memberikan hasil transkripsi yang akurat dan dapat diandalkan, sehingga peserta rapat tidak perlu lagi mencatat secara manual dan dapat lebih fokus pada diskusi yang berlangsung. Selain itu, integrasi dengan Google Meet melalui Puppeteer membuat sistem ini mudah digunakan tanpa memerlukan perubahan signifikan pada infrastruktur yang sudah ada, yang turut meningkatkan efisiensi dan produktivitas selama rapat.

Selama proses pengembangan, berbagai kendala teknis seperti gangguan jaringan dan kecepatan proses transkripsi berhasil diatasi melalui optimasi skrip dan konfigurasi sistem. Tantangan dalam integrasi komponen teknologi juga berhasil diatasi melalui proses *debugging* dan *testing* yang intensif. Pengalaman ini memberikan wawasan mendalam tentang tantangan nyata dalam pengembangan sistem berbasis AI, serta relevansi akademik yang signifikan dalam mengaplikasikan teori-teori yang telah dipelajari.

Secara keseluruhan, tugas akhir ini membuktikan bahwa integrasi teknologi AI dengan platform komunikasi seperti Google Meet memiliki potensi besar dalam meningkatkan efisiensi dan produktivitas rapat. Hasil pengujian dan implementasi yang dilakukan menunjukkan bahwa tujuan dari pengembangan bot notula ini berhasil dicapai, memberikan solusi yang efektif dan praktis untuk pencatatan notula rapat secara otomatis. Pengalaman yang diperoleh dari tugas akhir ini memberikan pembelajaran berharga yang dapat dijadikan dasar untuk pengembangan lebih lanjut di masa depan.

5.2 Saran

Format Untuk pengembangan lebih lanjut dari bot notula, ada beberapa aspek yang perlu diperhatikan guna meningkatkan performa dan kegunaannya. Disarankan untuk melakukan penelitian dan pengembangan lebih lanjut pada teknologi Automatic Speech Recognition (ASR) yang digunakan, dengan fokus pada peningkatan akurasi transkripsi dalam kondisi audio yang rendah. Penggunaan algoritma yang lebih canggih dan model pembelajaran mendalam (deep learning) dapat membantu mengurangi kesalahan pengenalan kata (Word Error Rate/WER). Selain itu, penggunaan data pelatihan yang lebih beragam, mencakup berbagai kondisi audio seperti lingkungan yang bising dan rekaman dengan kualitas mikrofon yang berbeda, akan membuat model ASR lebih adaptif terhadap variasi kualitas audio. Untuk meningkatkan kualitas audio input, bot notula juga dapat diintegrasikan dengan teknologi pengurangan kebisingan (*noise reduction*), yang dapat memfilter suara latar belakang yang tidak diinginkan sehingga suara yang relevan lebih jelas dan mudah dikenali oleh sistem ASR. Penggunaan teknik seperti spektral gating atau filter adaptif dapat membantu dalam mencapai kualitas audio yang lebih baik.

Lakukan pengujian bot pada berbagai skenario dan lingkungan berbeda untuk memastikan robustitas dan kehandalan sistem. Pengujian harus mencakup berbagai jenis rapat, mulai dari rapat formal dengan sedikit gangguan hingga diskusi kelompok yang lebih interaktif dan dinamis, serta berbagai lingkungan akustik seperti ruangan dengan echo atau latar belakang bising. Selain itu, penting untuk meminta feedback dari pengguna terkait akurasi dan kegunaan bot notula untuk mengidentifikasi area yang perlu perbaikan dan peningkatan lebih lanjut. Feedback dari pengguna nyata sangat berharga dalam mengidentifikasi kelemahan sistem yang mungkin tidak terdeteksi selama fase pengujian internal. Melalui survei, wawancara, dan analisis penggunaan, pengembang dapat memperoleh wawasan tentang pengalaman pengguna dan kebutuhan spesifik yang mungkin belum terpenuhi. Data ini dapat digunakan untuk membuat perbaikan yang tepat sasaran dan meningkatkan kepuasan pengguna. Secara keseluruhan, langkah-langkah ini diharapkan dapat membantu dalam mengoptimalkan performa bot notula, membuatnya lebih andal, dan meningkatkan kepuasan pengguna dalam berbagai kondisi rapat dan lingkungan.

Selain itu, perusahaan Widya Robotics disarankan untuk melakukan riset terkait model manajemen proyek yang cocok untuk perusahaan. Menerapkan manajemen proyek yang tepat dapat mengoptimalkan proses pengembangan dan memastikan kesuksesan proyek. Beberapa hal yang dapat dipertimbangkan adalah:

1. Implementasi Model Manajemen Proyek yang Sesuai: Perusahaan sebaiknya melakukan riset untuk menemukan model manajemen proyek yang paling cocok dengan kebutuhan dan karakteristiknya. Model seperti Waterfall, Agile, Kanban, atau Hybrid bisa dipertimbangkan. Pemilihan model yang tepat akan membantu dalam mengelola proyek secara lebih efektif.
2. Penetapan Tujuan dan Sasaran yang Jelas: Setiap proyek harus dimulai dengan menetapkan tujuan dan sasaran yang jelas dan terukur. Ini membantu semua anggota tim memahami apa yang diharapkan dan bagaimana kontribusi mereka mendukung tujuan keseluruhan proyek.
3. Komunikasi yang Efektif: Meningkatkan komunikasi antara anggota tim proyek melalui rapat rutin, alat kolaborasi digital, dan laporan status proyek. Komunikasi yang baik membantu mengidentifikasi dan menyelesaikan masalah lebih cepat serta memastikan semua pihak tetap sinkron.
4. Pengelolaan Risiko: Melakukan analisis risiko di awal proyek dan mengembangkan rencana mitigasi risiko yang komprehensif. Ini termasuk mengidentifikasi potensi risiko, menilai dampaknya, dan merencanakan tindakan pencegahan untuk mengurangi atau mengelola risiko tersebut.
5. Peningkatan Kapabilitas Tim: Menginvestasikan dalam pelatihan dan pengembangan profesional untuk anggota tim agar mereka memiliki keterampilan dan pengetahuan terbaru yang relevan dengan proyek. Ini dapat mencakup pelatihan teknis, manajemen proyek, atau soft skills.
6. Dokumentasi Pengembangan: Sangat penting untuk mendokumentasikan seluruh proses pengembangan dengan baik. Dokumentasi yang komprehensif akan memudahkan handover kode atau sistem kepada developer baru yang nantinya akan masuk ke tim. Dengan dokumentasi yang baik, transisi antar anggota tim menjadi lebih mudah dan memastikan bahwa pengetahuan tidak hilang ketika ada perubahan tim.

Dengan menerapkan strategi-strategi ini, diharapkan perusahaan Widya Robotics dapat meningkatkan efisiensi dan efektivitas dalam manajemen proyek, sehingga menghasilkan produk yang lebih berkualitas dan memenuhi kebutuhan pengguna dengan lebih baik.

DAFTAR PUSTAKA

Alturki, U., & Aldraiweesh, A. (2022). Adoption of Google Meet by Postgraduate Students: The Role of Task Technology Fit and the TAM Model. *Sustainability (Switzerland)*, *14*(23). https://doi.org/10.3390/su142315765

Amorese, T., Greco, C., Cuciniello, M., Milo, R., Sheveleva, O., & Glackin, N. (2023). *Automatic speech recognition (ASR) with Whisper: Testing Performances in Different Languages*. http://ceur-ws.org

Arwin Dermawan, D., Chamdan Mashuri, M., Ginanjar Setyo Permadi, Mk., & Duta Alif Gunawan Dini Widiasih, Mk. S. (2022). *Membuat Game Berbasis Website Menggunakan Bahasa Javascript dan PHP*. www.rcipress.rcipublisher.org

Ćatović, A., Buzađija, N., & Lemes, S. (2022). Microservice development using RabbitMQ message broker. *Science, Engineering and Technology*, *2*(1), 30–37. https://doi.org/10.54327/set2022/v2.i1.19

De Cerff, W. S., Van De Vegte, J., Boers, R., Brandsma, T., Haij, M. De, Moosel, W. Van, Noteboom, J. W., Pagani, G. A., & Van Der Schrier, G. (2018). Agile development in meteorological r&d achieving a minimum viable product in a scrum work setting. *Bulletin of the American Meteorological Society*, *99*(12), 2507–2518. https://doi.org/10.1175/BAMS-D-17-0273.1

Firmansyah, Y., Maulana, R., & Maulana, M. S. (2021). Implementasi Metode SDLC Prototype Pada Sistem Informasi Indeks Kepuasan Masyarakat (IKM) Berbasis Website Studi Kasus Dinas Kependudukan Dan Catatan Sipil. *Jurnal Sistem Dan Teknologi Informasi (Justin)*, *9*(3), 315. https://doi.org/10.26418/justin.v9i3.46964

Gupta, V., Rauscher, M., Casqueira, Schmidt, E., Migone, T., & Somo. (2022). *Meetbot*. Balena Cloud. https://github.com/balena-io-experimental/meetbot

KBBI VI Daring. (n.d.). Retrieved May 26, 2024, from https://kbbi.kemdikbud.go.id/entri/notula

Iqbal, J. (2014). Detection and Prevention of Changes in the DOM Tree. In *BZU*. THE UNIVERSITY OF NEW BRUNSWICK.

Kanda, N., Ye, G., Wu, Y., Gaur, Y., Wang, X., Meng, Z., Chen, Z., & Yoshioka, T. (2021). *Large-Scale Pre-Training of End-to-End Multi-Talker ASR for Meeting Transcription with Single Distant Microphone*. http://arxiv.org/abs/2103.16776

Lyu, K. M., Lyu, R. yuan, & Chang, H. T. (2024). Real-time multilingual speech recognition and speaker diarization system based on Whisper segmentation. *PeerJ Computer Science*, *10*, e1973. https://doi.org/10.7717/PEERJ-CS.1973/SUPP-1

Morris, A. C., Maier, V., & Green, P. (2004). From WER and RIL to MER and WIL: Improved evaluation measures for connected speech recognition. *8th International Conference on Spoken Language Processing, ICSLP 2004*, 2765–2768. https://doi.org/10.21437/interspeech.2004-668

Nitin Ghadge, S. (2024). AI-Powered Information Retrieval in Meeting Records and AI-Powered Information Retrieval in Meeting Records and Transcripts Enhancing Efficiency and User Experience. In *Part of the Anthropology Commons*. Communication Commons. https://repository.fit.edu/etd

Nurkholis, I., & Yunial, A. H. (2023). Automation Testing Tool Dalam Pengujian Website Sistem Informasi Pembayaran Iuran Dan LKS Di MTS Al-Ittihad Menggunakan Puppeteer. *LOGIC : Jurnal Ilmu Komputer Dan Pendidikan*, *2*(1), 101–108. https://doi.org/10.4230/OASICS.ICPEC.2022.10

Park, T. J., Kanda, N., Dimitriadis, D., Han, K. J., Watanabe, S., & Narayanan, S. (2022). A review of speaker diarization: Recent advances with deep learning. *Computer Speech & Language*, *72*, 101317. https://doi.org/10.1016/J.CSL.2021.101317

Setiawan, R. (2021). *Apa Itu Bot? Ayo Berkenalan Lebih Dekat Dengannya - Dicoding Blog*. https://www.dicoding.com/blog/apa-itu-bot/

srivastava, pranjal. (2021). *Node.js Puppeteer - GeeksforGeeks*. https://www.geeksforgeeks.org/node-js-puppeteer/

Sushil, N. (2024). Multilingual Video Viewing Subtitle to Audio Translator. *International Journal for Research in Applied Science and Engineering Technology*, *12*(5), 697–703. https://doi.org/10.22214/ijraset.2024.61621

Wang, M. Y., & Purushotam, G. N. (2023). Using Deep Learning and Augmented Reality to Improve Accessibility: Inclusive Conversations Using Diarization, Captions, and Visualization. *2023 ASEE Annual Conference & Exposition*. https://peer.asee.org/44572

pptr.dev. (n.d.). Retrieved May 26, 2024, from https://pptr.dev/guides/what-is-puppeteer

Zima, B., & Barszcz, M. (2024). *Comparative analysis of NodeJs frameworks*.

LAMPIRAN