BAB III

PELAKSANAAN KERJA

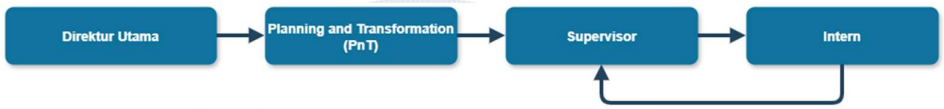
3.1 Kedudukan dan Koordinasi

3.1.1 Kedudukan

Kegiatan magang di PT Telekomunikasi Selular (Telkomsel), peserta magang menempati posisi sebagai *AI Product Development Intern* pada Direktorat Planning and Transformation (PnT). Direktorat ini berperan penting dalam merancang dan mengimplementasikan berbagai inisiatif strategis untuk mendorong transformasi digital serta inovasi berbasis teknologi di lingkungan Telkomsel. Sebagai bagian dari tim di bawah direktorat tersebut, penulis memiliki kedudukan yang berfokus pada pengembangan dan eksplorasi solusi berbasis *Artificial Intelligence* (AI), khususnya yang berkaitan dengan *agentic AI systems*, orkestrasi *multi-agent*, serta integrasi teknologi AI ke dalam proses bisnis internal. Kedudukan ini juga mencakup tanggung jawab untuk melakukan riset, menguji konsep, serta membangun *prototype* proyek-proyek berbasis AI yang relevan dengan kebutuhan strategis perusahaan. Salah satu proyek utama yang dikerjakan adalah *Automation Ads Generation*, yaitu sebuah sistem otomatisasi *end-to-end* yang dirancang untuk menghasilkan konten iklan secara otomatis menggunakan kombinasi *workflow automation* dan AI *agent*. Proyek ini bertujuan untuk menyederhanakan proses pembuatan materi promosi, mulai dari pengumpulan *brief* pengguna, membuat konten teks dan visual, proses *approval*, hingga publikasi ke platform sosial media secara otomatis. Dalam pelaksanaannya, peserta magang bekerja secara kolaboratif dengan pembimbing untuk memastikan bahwa setiap inisiatif yang dikembangkan selaras dengan arah transformasi korporasi yang sedang dijalankan Telkomsel. Posisi ini memberikan kesempatan bagi penulis untuk memahami secara langsung bagaimana pendekatan teknologi cerdas diterapkan dalam konteks industri telekomunikasi yang kompleks dan dinamis. Melalui

kedudukan tersebut, peserta magang tidak hanya memperoleh pengalaman teknis dalam pengembangan produk AI, tetapi juga wawasan strategis mengenai bagaimana inovasi dapat dikelola untuk mendukung pertumbuhan dan efisiensi organisasi di era digital.

3.1.2 Koordinasi



Gambar 3.1 Bagan Alur Koordinasi

Dalam pelaksanaan kegiatan magang di Telkomsel, alur koordinasi kegiatan magang mengikuti struktur hierarkis yang berlaku di perusahaan, sebagaimana digambarkan pada bagan di atas. Secara umum, koordinasi dimulai dari Direktur Utama yang memberikan arahan strategis pada Direktorat Planning and Trasformation (PnT) sebagai pelaksana transformasi dan inovasi digital di tingkat korporasi. Selanjutnya, Direktorat PnT meneruskan arahan tersebut kepada Supervisor yang berperan sebagai pengarah teknis dan pembimbing lapangan bagi peserta magang. Supervisor bertanggung jawab dalam memberikan bimbingan terkait perencanaan proyek, pemberian tugas, hingga evaluasi hasil kerja yang dilakukan oleh peserta magang.

Dalam pelaksanaan kegiatan sehari-hari, penulis berkoordinasi langsung dengan supervisor untuk menyampaikan progres pekerjaan, mendiskusikan kendala teknis, serta mendapatkan umpan balik terhadap hasil pengembangan proyek. Alur koordinasi ini memastikan adanya komunikasi dua arah antara peserta dan supervisor, di mana peserta tidak hanya menerima instruksi, tetapi juga dapat memberikan masukan atau hasil evaluasi yang bersifat konstruktif terhadap pengembangan proyek. Sebagai bagian dari mekanisme koordinasi yang telah ditetapkan, penulis juga wajib untuk menyusun dan mengirimkan tugas atau laporan progres mingguan kepada supervisor secara rutin setiap hari Senin. Pengiriman tugas mingguan ini sebagai sarana *monitoring* perkembangan pekerjaan sekaligus bahan evaluasi

bagi supervisor dalam memberikan arahan lanjutan. Dengan adanya mekanisme koordinasi yang sistematis dan terjadwal tersebut, proses pembelajaran dan pelaksanaan tugas magang dapat berlangsung secara efektif, terarah, dan selaras dengan kebutuhan perusahaan. Struktur koordinasi ini juga memberikan kesempatan bagi peserta untuk memahami dinamika kerja di lingkungan korporasi, termasuk bagaimana komunikasi lintas fungsi dan pengambilan keputusan dilakukan secara profesional dalam konteks organisasi besar seperti Telkomsel.

3.2 Tugas yang Dilakukan

Tabel 3.1 Detail Pekerjaan yang Dilakukan

No.	Tanggal Mulai	Tanggal Selesai	Proyek	Keterangan
Foundations & Orientation				
1.	1 Agustus 2025	8 Agustus 2025	Orientasi (*briefing* program, pengenalan jenis AI, studi kasus)	Kegiatan dimulai dengan *briefing* program, pemahaman *workplan*, dan pengenalan jenis-jenis AI termasuk *Agentic AI*. Minggu ini menjadi fondasi sebelum masuk ke tahap teknis.
2.	11 Agustus 2025	15 Agustus 2025	Dasar-dasar *Agentic* AI (arsitektur agen otonom, memori, *tools*, *reasoning loops*, eksplorasi	Mendalami komponen *Agentic AI* seperti LLMs, *memory*, *tools*, dan *reasoning loops*, lalu mengeksplorasi *framework* seperti LangChain, LlamaIndex, dan AutoGen. Kegiatan ditutup dengan penyusunan peta konsep *Agentic AI.*
3.	18 Agustus 2025	22 Agustus 2025	pengenalan *tools* otomatisasi (n8n, bandingkan dengan Zapier dan Make.com	Membangun *dummy workflow* di n8n untuk memahami otomatisasi *no-code* dan integrasi API. Minggu ini juga menguji dan membandingkan n8n dengan Zapier dan Make.com.
4.	25 Agustus 2025	29 Agustus 2025	kolaborasi & *Workflow* Agen (pengenalan CrewAI, implementasi tugas kolaborasi 2 agen)	Menerapkan konsep *multi-agent* menggunakan CrewAI melalui dua skenario, *Researcher-Writer* dan *Customer Support Simulation*. Fokus kegiatan pada *task coordination* dan alur komunikasi antar agen.

Tools Exploration & Prototyping				
5.	1 September 2025	5 September 2025	*Workflow* lanjutan di n8n (membangun *multi-step workflow*, integrasi API, *error handling* & *trigger*)	Mengembangkan *AI Customer Care System* berbasis *multi-step workflow* dengan integrasi OpenAI dan HuggingFace. *Workflow* pendukung untuk *error handling* otomatis juga dibuat agar sistem lebih stabil.
6.	8 September 2025	12 September 2025	Pendalaman CrewAI (membangun *multi-role agent*, uji *task handoff*)	Membangun arsitektur *multi-role agent* berisi *planner*, *executor*, *verifier*, hingga *optimizer* untuk simulasi kolaborasi agen yang kompleks. minggu ini fokus menguji alur *handoff* antar peran dan stabilitas *multi-agent workflow*.
7.	15 September 2025	19 September 2025	Langchain/AutoGen (penerapan memori, *tools*, dan *reasoning chain*)	Mengimplementasikan memori, pemanggilan *tools*, dan *reasoning chain* berbasis LangChain untuk menyusun alur logika berlapis.
8.	22 September 2025	26 September 2025	Evaluasi & *Benchmarking* (perbandingan n8n, CrewAI, LangChain dari sejumlah aspek)	Melakukan *benchmarking* tiga platform (n8n, CrewAI, LangChain) berdasarkan *ease of use*, *scalability*, dan *flexibility*. Hasil evaluasi disajikan dalam *comparison matrix* yang memperlihatkan kekuatan dan batasan masing-masing *tools*.
Applied Projects				
9.	29 September 2025	3 September 2025	Konsep proyek utama	Melakukan *brainstorming* beberapa ide proyek dan menganalisis kelayakannya berdasarkan kompleksitas dan relevansi. Pada akhir sesi, ditetapkan proyek *Automation Ads Generation* sebagai proyek utama yang akan dikembangkan.
10.	6 Oktober 2025	17 Oktober 2025	Integrasi *Agent*, Data, & API(implementasi alur kerja *end-to-end*)	Mendesain alur *input* berbasis Telegram yang mengumpulkan *brief* iklan secara bertahap menggunakan *AI Agent*. Sistem memproses *input* dan menyiapkan *pipeline* untuk otomatisasi pembuatan konten.

11.	20 Oktober 2025	24 Oktober 2025	Peningkatan otonomi Agen	Mengoptimalkan alur agen agar lebih adaptif melalui *reasoning loops*, validasi otomatis, dan struktur logika yang lebih mandiri. Tahap ini meningkatkan kemampuan agen untuk mengambil keputusan tanpa banyak intervensi manual.
Testing, Deployment & Presentation				
12.	27 Oktober 2025	7 November 2025	Pengujian & Penyempurnaan	Melakukan pengujian *workflow*, penanganan eror otomatis, serta penyempurnaan logika publikasi konten. Perbaikan dilakukan pada stabilitas API, pengelolaan sesi, dan penanganan *payload* agar sistem lebih konsisten.
13.	10 November 2025	14 November 2025	Dokumentasi proyek	Menyusun dokumentasi lengkap mencakup arsitektur sistem, alur kerja, konfigurasi agen, serta panduan penggunaan. Dokumentasi dibuat untuk memastikan proyek dapat dipahami dengan mudah.
14.	17 November 2025	21 November 2025	Presentasi Akhir	Menyiapkan dan mempresentasikan hasil proyek beserta demo sistem.

3.3 Uraian Pelaksanaan Kerja

3.3.1 Proses Pelaksanaan

3.3.1.1 Orientasi (*briefing* program, pengenalan jenis AI, studi kasus)

Sesi awal program magang ini difokuskan pada *briefing* program secara menyeluruh, yang diperlukan untuk menyelaraskan pemahaman dan ekspektasi peserta magang serta menetapkan dasar yang kokoh bagi kegiatan yang akan datang. Tujuan utama dari fase ini adalah memberikan gambaran mendalam mengenai *roadmap*, menguraikan fase-fase penting, dan memperkenalkan proyek akhir yang menjadi luaran utama dari kegiatan magang. Secara spesifik, penetapan objektif program ditekankan untuk memastikan setiap

peserta dapat mengukur capaian personal dan kontribusi tim terhadap tujuan organisasi yang lebih besar. Selanjutnya, diskusi mengenai ekspektasi kinerja dijabarkan, termasuk standar kualitas *output*, etika kerja, dan pentingnya komunikasi yang efektif dalam tim virtual, sehingga memberikan kejelasan operasional yang maksimal. Penekanan juga diberikan pada budaya proaktif dalam pembelajaran dan kemampuan beradaptasi terhadap teknologi baru yang terus berkembang pesat mengingat sifat dinamis dari bidang kecerdasan buatan. Dengan adanya pemahaman yang jelas mengenai kerangka kerja ini, peserta diharapkan dapat segera mengintegrasikan diri ke dalam alur kerja dan mulai memberikan kontribusi yang berarti sejak dini.

Pengantar mendalam mengenai klasifikasi jenis-jenis Kecerdasan Buatan (AI) merupakan inti dari sesi pembelajaran dengan membedah tiga kategori utama, yaitu *Reactive* AI, *Generative* AI, dan *Agentic* AI. *Reactive* AI merupakan bentuk AI paling dasar yang hanya mampu merespons *input* saat ini tanpa memiliki memori masa lalu atau kemampuan pembelajaran, seperti yang dicontohkan oleh sistem *Deep Blue* milik IBM. Selanjutnya, *Generative* AI diperkenalkan sebagai evolusi signifikan yang mampu menciptakan konten baru, seperti teks, gambar, dan kode, berdasarkan pola yang dipelajari dari set data yang masif, sehingga menjadikannya sebagai alat revolusioner di berbagai industri kreatif dan teknis. Namun, fokus utama diberikan pada *Agentic* AI yang ditandai dengan kemampuan untuk menetapkan tujuan, merencanakan serangkaian langkah, berinteraksi dengan lingkungan eksternal (API atau *tools*), dan secara otonom menjalankan tugas kompleks untuk mencapai target yang ditetapkan, sehingga menjadikannya sebagai konsep AI yang paling canggih untuk saat ini. Perbedaan fundamentalnya terletak pada kemampuan *Agentic* AI untuk mengambil inisiatif dan melakukan iterasi berbasis umpan

balik dari aksinya, bukan hanya merespons atau menghasilkan konten statis.

Gambar 3.2 Eksplorasi konsep AI

Untuk mematangkan konsep *Agentic* AI, tugas pertama yang dilakukan adalah melakukan eksplorasi praktis penerapan konsep *Agentic* AI dalam skenario nyata dengan bantuan platform orkestrasi *workflow* n8n. Eksplorasi ini bertujuan mendalami bagaimana agen otonom dapat diorkestrasi dalam alur kerja yang kompleks untuk menjalankan tugas berlapis tanpa memerlukan intervensi langsung dari pengguna. Proses implementasi mencakup perancangan alur kerja modular yang mengombinasikan *trigger, logic node,* dan *action node*, yang memungkinkan AI untuk menyimulasikan proses berpikir (*reasoning*) dan bertindak (*act*) secara terintegrasi dan berurutan. Sebagai studi kasus praktis, peserta mencoba menyimulasikan sebuah agen yang dirancang untuk menerima *input* teks, menganalisis konteksnya, melakukan pencarian informasi tambahan melalui API eksternal sebagai *tool use*, dan kemudian menghasilkan respons yang terstruktur secara mandiri. Melalui percobaan mendalam ini, pentingnya integrasi elemen *Tool Use* dan

Planning & *Reasoning* teridentifikasi sebagai dua pilar kunci dalam membangun sistem *Agentic* AI yang efisien dan fungsional. Selain itu, kegiatan ini memperjelas perbedaan substansial antara sistem berbasis one-*shot prompting* yang pasif dengan sistem yang memanfaatkan *multi-step reasoning*, yang terbukti jauh lebih adaptif dalam menghadapi dinamika dan kompleksitas tugas nyata. Hasil dari kegiatan praktis ini memperkuat pemahaman bahwa *Agentic* AI bukan sekadar perpanjangan dari *Generative* AI, melainkan merupakan paradigma baru yang menuntut kemampuan desain sistem yang modular, reflektif, dan kolaboratif untuk mencapai tingkat otonomi yang sesungguhnya.

3.3.1.2 Agentic AI Fundamentals

Pada minggu kedua pelaksanaan magang, fokus kegiatan diarahkan pada eksplorasi mendalam mengenai arsitektur agen otonom (*autonomous agents*) berbasis kecerdasan buatan. Pembahasan ini meliputi pemahaman komponen utama yang membentuk sistem *Agentic* AI, yaitu *Large Language Models* (LLMs), *memory*, *tools*, serta *reasoning loops*. Setiap komponen memiliki fungsi saling melengkapi dalam membentuk agen yang mampu berpikir, mengambil keputusan, dan bertindak secara mandiri. *Large Language Model* seperti GPT-4, Claude, atau LLaMA-3 berperan sebagai pusat pemrosesan dan penalaran yang berfungsi menafsirkan instruksi, memahami konteks, dan menghasilkan keluaran berbasis bahasa yang relevan. Model ini bekerja melalui mekanisme *prompt-driven reasoning*, yang memungkinkan agen menganalisis situasi dan membuat keputusan berdasarkan data yang diterimanya. Namun, kemampuan LLM akan terbatas tanpa dukungan sistem lain seperti *memory* dan *tools*, yang memperluas kapasitasnya untuk belajar dari pengalaman serta berinteraksi dengan lingkungan eksternal.

Komponen *memory* menjadi aspek krusial dalam menjaga kontinuitas interaksi dan peningkatan kemampuan agen dari waktu ke waktu. Dalam konteks *Agentic* AI, *short-term memory* digunakan untuk menyimpan konteks percakapan aktif dalam satu sesi, sementara *long-term memory* menyimpan pengetahuan dan pengalaman yang diperoleh dari berbagai interaksi sebelumnya. Penyimpanan jangka panjang ini umumnya dikelola melalui *vector database* yang mampu merekam dan mencatat informasi berdasarkan kemiripan semantik. Dengan sistem ini, agen tidak hanya mampu mengingat percakapan sebelumnya, tetapi juga dapat menyesuaikan responsnya secara personal berdasarkan konteks historis pengguna. Implementasi memori semacam ini menciptakan agen yang lebih *context-aware* dan adaptif, karena setiap keputusan yang diambil didasari oleh pengetahuan yang terus diperbarui. Dalam konteks pengembangan, pemahaman terhadap struktur memori ini penting untuk membangun agen yang mampu mempertahankan konsistensi perilaku dan belajar dari pengalaman sebelumnya secara berkelanjutan.

Selain itu, *tools* merupakan komponen yang memperluas kemampuan agen di luar sekadar generasi teks. Melalui integrasi API, eksekusi kode, dan koneksi dengan sistem eksternal, agen dapat melakukan tindakan nyata seperti mengambil data terkini, menjalankan perhitungan, atau bahkan memproses media visual dan audio. Hal ini menjadikan agen lebih dinamis dan mampu menjembatani pemahaman bahasa dengan aksi dunia nyata. Misalnya, sebuah agen dapat menggunakan *retrieval tool* untuk mengakses data perusahaan secara *real-time* atau menjalankan perintah otomatisasi kerja. Dengan menggabungkan LLM dan *tools*, agen dapat berfungsi layaknya asisten digital yang bukan hanya memberikan saran, tetapi juga mengeksekusi tindakan yang diperlukan untuk mencapai tujuan. Dalam konteks magang,

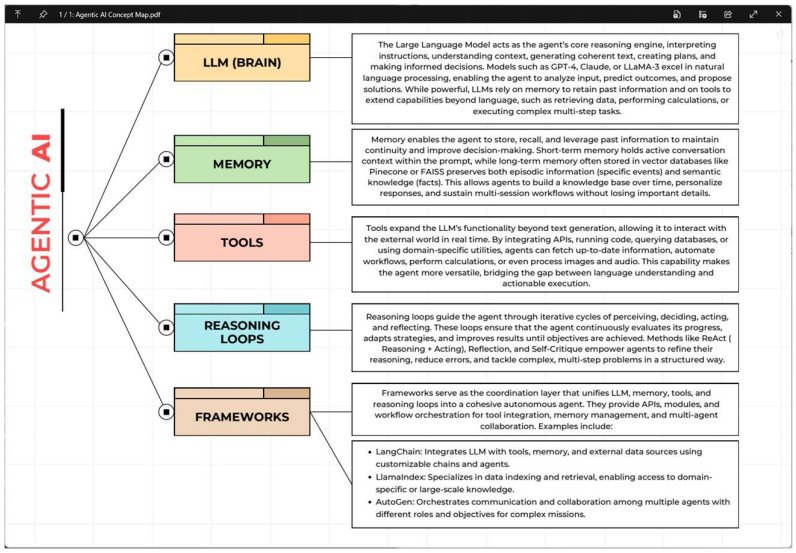
pemahaman mengenai integrasi *tools* menjadi fondasi penting untuk merancang sistem *multi-agent* yang mampu melakukan kolaborasi dan menyelesaikan tugas kompleks secara mandiri.

Komponen berikutnya adalah *reasoning loops*, yaitu mekanisme berpikir berulang yang memungkinkan agen melakukan penalaran, tindakan, evaluasi, dan refleksi secara berkelanjutan. Melalui *reasoning loop*, agen dapat menilai hasil tindakannya, mengidentifikasi kesalahan, dan memperbaiki strategi untuk menghasilkan keluaran yang lebih akurat. Pendekatan seperti ReAct (*Reasoning + Acting*), *Reflection*, dan *Self-Critique* banyak digunakan untuk meningkatkan kemampuan berpikir secara otonom pada agen modern. Proses ini menyerupai cara manusia belajar dari umpan balik, dengan melakukan tindakan, menilai hasil, dan memperbaikinya secara repetitif. Dalam praktiknya, *reasoning loops* membantu agen menangani masalah berskala besar yang memerlukan beberapa langkah penalaran, seperti analisis data kompleks atau *multi step planning*.

Setelah memahami komponen fundamental *Agentic* AI, tahap berikutnya dalam proyek magang adalah melakukan eksplorasi terhadap berbagai *framework* yang digunakan untuk membangun agen otonom, yaitu LangChain, LlamaIndex, dan AutoGen. Ketiga *framework* ini berperan sebagai *coordination layer* yang menghubungkan seluruh komponen ke dalam satu sistem yang terpadu. LangChain merupakan *framework* yang dirancang untuk mengintegrasikan model bahasa dengan data eksternal, memori dan API eksternal melalui konsep *chain* dan *agents*. *Framework* ini memudahkan pengembang untuk membangun alur kerja (*workflow*) berbasis logika yang dapat diatur sesuai kebutuhan aplikasi. Di sisi lain, LlamaIndex berfokus pada pengindeksan dan pengambilan data yang memungkinkan agen mengakses informasi domain spesifik

secara efisien. Pendekatan ini sangat relevan untuk skenario di mana agen perlu menelusuri dan mengelola sumber data besar yang tidak terstruktur.

Sementara itu, AutoGen menawarkan pendekatan yang lebih maju dalam orkestrasi agen, dengan memfasilitasi komunikasi dan kolaborasi antar beberapa agen yang memiliki peran dan tujuan berbeda. *Framework* ini memungkinkan terbentuknya sistem *multi-agent collaboration*, di mana setiap agen dapat saling berinteraksi, berdiskusi, dan berbagi hasil *reasoning* untuk menyelesaikan tugas kompleks secara kolektif. Konsep ini meniru dinamika kerja tim manusia di lingkungan profesional, di mana pembagian peran dan kemampuan analisis kolektif menjadi kunci keberhasilan. Dalam kegiatan magang, eksplorasi *framework* ini memberikan wawasan mendalam tentang bagaimana sistem AI modern tidak hanya bekerja secara individual, tetapi juga secara terkoordinasi dalam lingkungan *multi-agent*.



Gambar 3.3 *Concept Map of Agentic AI Components*

Sebagai hasil dari kegiatan di minggu kedua ini peserta menghasilkan *Concept Map of Agentic AI Components*, yaitu peta

konsep yang menggambarkan hubungan antar elemen pembentuk agen otonom. Peta konsep tersebut menjadi visualisasi dari pemahaman mengenai bagaimana LLM berperan sebagai pusat penalaran, memori sebagai penyimpanan pengetahuan, *tools* sebagai perpanjangan kemampuan agen, serta *reasoning loops* seabgai mekanisme berpikir adaptif. *Framework* seperti LangChain, LlamaIndex, dan AutoGen berfungsi sebagai laporan koordinasi yang menyatukan semua komponen tersebut menjadi sistem yang utuh dan berfungsi secara otonom. *Output* ini tidak hanya merepresentasikan hasil pembelajaran teknis, tetapi juga menunjukkan pemahaman konseptual terhadap cara kerja *Agentic* AI secara menyeluruh. Dengan demikian, kegiatan ini menjadi dasar yang kuat untuk tahap berikutnya dalam proyek magang, yaitu implementasi dan orkestrasi *multi-agent* yang lebih kompleks.

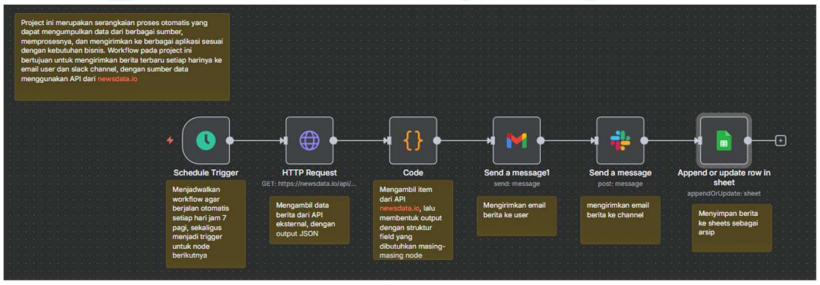
3.3.1.3 Automation Tools Intro

Gambar 3.4 Logo n8n

N8n merupakan platform otomatisasi alur kerja berkonsep *node-based* yang memungkinkan pembuatan *pipeline* integrasi tanpa atau dengan sedikit penulisan kode, sehingga cocok untuk praktisi non-*programmer* sekaligus tim teknis yang membutuhkan fleksibilitas. Antarmuka visualnya mengandalkan *node* sebagai blok fungsi sehingga proses desain *workflow* menjadi intuitif, dengan hanya menghubungkan *node* satu per satu untuk membentuk alur data *end-to-end*. Konsep *no-code/low-code* pada n8n mempermudah *prototyping* dan iterasi karena perubahan *flow* dapat diuji langsung tanpa perlu *deploy* aplikasi besar. Namun, n8n tetap menyediakan *Code node* untuk transformasi data yang kompleks sehingga

developer tidak kehilangan kebebasan teknis. Selain itu, n8n mendukung *scheduling trigger*, *credential* untuk layanan pihak ketiga, dan kemampuan *self-hosting* untuk kebutuhan privasi dan kontrol sumber daya. Dalam praktik magang, pemahaman tentang konsep ini penting karena memungkinkan desain solusi otomatis yang dapat diskalakan dari tugas sederhana hingga *pipeline* data yang kompleks. Platform ini juga memfasilitasi *logging* dan debug sehingga *troubleshooting* menjadi lebih cepat dibandingkan menelusuri kode program linear.

Pada minggu ketiga, fokus pembelajaran diarahkan pada pengenalan konsep *no-code* dan *low-code automation* menggunakan platform n8n. Dalam kegiatan ini proyek yang dikerjakan bersifat *dummy project*, yaitu simulasi *workflow* sederhana yang tidak melibatkan data nyata perusahaan, melainkan digunakan murni untuk *training* dan eksplorasi fitur-fitur n8n. Melalui proyek latihan ini, peserta magang dapat memahami bagaimana otomatisasi digital dapat mempercepat proses kerja dan mengurangi intervensi manual tanpa risiko terhadap sistem produksi sebenarnya. Kegiatan ini juga menjadi fondasi penting untuk memahami bagaimana *agentic system* dapat diintegrasikan ke dalam ekosistem kerja melalui otomatisasi berbasis API dan layanan *cloud*.



Gambar 3.5 *News automation workflow*

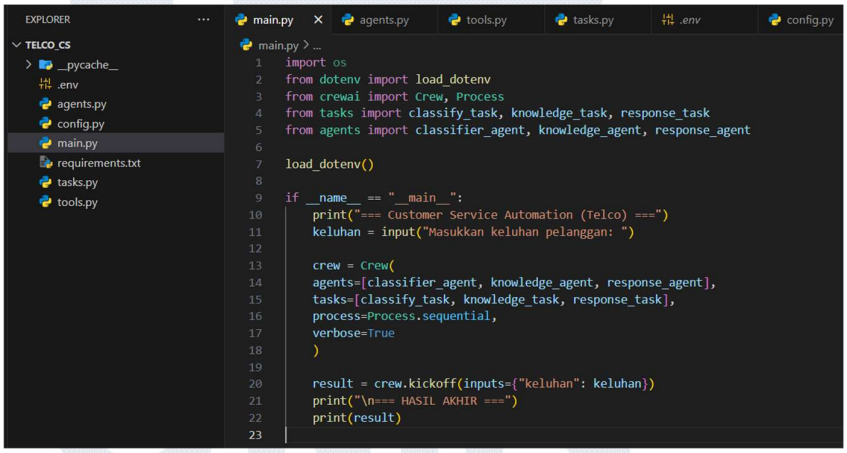
Sebagai bentuk praktik langsung dari pemahaman konsep *low-code automation*, peserta magang membangun *workflow*

simulasi menggunakan n8n yang mengintegrasikan berbagai API populer, seperti *email notification* dan *Slack alert*. *Workflow* ini dirancang untuk menjalankan proses otomatis di mana sistem mengambil data dari API eksternal, kemudian mengirimkan notifikasi ringkas ke platform komunikasi internal seperti Slack, serta mengirimkan versi lengkapnya melalui email. Alur kerja ini dimulai dengan *node Schedule Trigger* yang dijadwalkan berjalan secara berkala, diikuti *node HTTP Request* untuk mengambil data API, lalu *node Code* untuk melakukan transformasi dan format teks agar kompatibel dengan media pengiriman yang berbeda. Setelah proses format selesai, data dikirim ke *node Slack* untuk memberikan pemberitahuan singkat, dan *node* Gmail untuk mengirimkan detail berita secara lengkap kepada pengguna. Karena proyek ini bersifat *dummy project*, seluruh data dan *endpoint* yang digunakan adalah data publik atau simulasi, sehingga aman dari aspek privasi dan keamanan. Tujuan utama dari latihan ini bukan pada nilai fungsional dari data yang dihasilkan, tetapi pada pemahaman tentang bagaimana *workflow automation* bekerja, bagaimana setiap *node* saling berinteraksi, serta bagaimana integrasi API dapat diimplementasikan secara efisien.

Setelah berhasil membangun *workflow* simulasi di n8n, tahap berikutnya adalah melakukan perbandingan terhadap dua platform otomatisasi populer lainnya, yaitu Zapier dan Make.com. Kedua platform ini juga mendukung konsep *no-code integration*, namun memiliki perbedaan mendasar dalam arsitektur, fleksibilitas, dan tingkat penyesuaian. Dalam konteks pelatihan, peserta membuat *dummy workflow* serupa pada kedua platform tersebut untuk memahami kelebihan dan keterbatasan masing-masing. Zapier unggul dari sisi kemudahan penggunaan dan koneksi instan dengan ribuan aplikasi populer, sehingga sangat cocok untuk pengguna non-teknis yang memerlukan otomatisasi cepat tanpa konfigurasi

kompleks. Sementara itu, Make.com menonjol dalam kemampuan manipulasi data secara visual dan mendalam. Misalnya, melakukan iterasi, agregasi, serta pemetaan data antar aplikasi yang lebih kompleks. di sisi lain, n8n menawarkan fleksibilitas tertinggi karena dapat *self-host* dan mendukung *custom code execution*, sehingga menjadikannya cocok untuk keperluan eksperimen dan penelitian. Melalui perbandingan ini, peserta memahami bahwa setiap platform memiliki peran dan segmen pengguna yang berbeda. Pemilihan platform otomatisasi yang tepat sangat bergantung pada kebutuhan organisasi, tingkat kemampuan teknis pengguna, serta skala kompleksitas proyek yang dikembangkan.

3.3.1.4 Collaboration & Workflow Agents



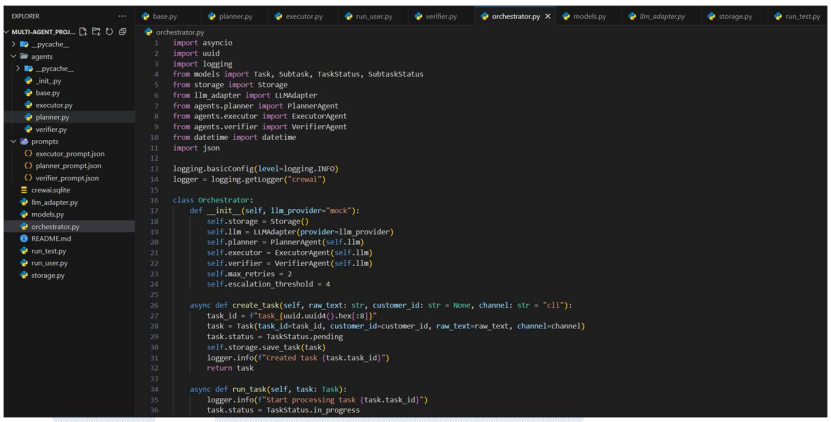
Gambar 3.6 Potongan kode CrewAI

Minggu keempat kegiatan magang, fokus pembelajaran diarahkan pada pengenalan konsep *multi-agent orchestration* menggunakan *framework* CrewAI. *Framework* ini merupakan salah satu inovasi baru dalam pengembangan sistem *agentic AI* yang memungkinkan koordinasi dan kolaborasi antar beberapa agen kecerdasan buatan secara terstruktur. CrewAI memungkinkan setiap agen memiliki peran, tujuan, dan konteks kerja yang berbeda, namun saling berinteraksi dalam satu kesatuan sistem yang dikelola oleh *Crew* sebagai pengatur utama. Dalam proyek ini, peserta magang

mempelajari bagaimana agen dapat dibentuk menggunakan konfigurasi sederhana berbasis Python melalui kelas *Agent* dan *Task* yang disediakan oleh *library* CrewAI. Masing-masing agen memiliki atribut seperti *role*, *goal*, dan *backstory* yang membantu menentukan gaya berpikir dan pola respons yang dihasilkan oleh LLM yang menjadi fondasinya. *Framework* ini juga memanfaatkan API eksternal (model LLaMA 3.1-8B Instant dari Groq API) untuk mendukung proses inferensi teks yang cepat dan efisien. Seluruh kegiatan ini dilakukan dalam bentuk *dummy project*, di mana seluruh data dan skenario hanya digunakan untuk simulasi dan eksplorasi konsep *multi-agent collaboration* tanpa melibatkan data operasional perusahaan.

Sebagai bentuk penerapan praktis, peserta membangun sebuah proyek kolaboratif dua agen menggunakan CrewAI yang melibatkan dua peran utama, yaitu *Researcher Agent* dan *Writer Agent*. *Researcher Agent* bertugas mengumpulkan informasi terkini mengenai perkembangan kecerdasan buatan pada tahun 2025, sedangkan *Writer Agent* bertugas menyusun laporan ringkas berdasarkan hasil riset yang dikumpulkan. Dalam *workflow* yang didefinisikan melalui *file* "main.py", kedua agen tersebut dijalankan di bawah satu *Crew Instance*, di mana *research_task* dieksekusi terlebih dahulu dan hasilnya dikirimkan sebagai konteks bagi *write_task*. Mekanisme ini meniru dinamika kerja manusia dalam tim kolaboratif, di mana satu agen berperan sebagai analis yang menyiapkan data dan wawasan, sementara agen lainnya bertindak sebagai penulis yang merangkai hasil tersebut menjadi laporan terstruktur. Model LLM yang digunakan, yaitu *groq/llama-3.1-8b-instant* memungkinkan komunikasi antar tugas berjalan cepat serta menjaga konsistensi gaya bahasa antar agen. Hasil akhir *workflow* ini ditampilkan dalam bentuk teks laporan yang dihasilkan oleh *Writer Agent* setelah menerima hasil dari *Researcher Agent*. Melalui

eksperimen ini, peserta memahami alur lengkap interaksi antar agen, mulai dari definisi peran, penyusunan tugas, pengaturan konteks, hingga proses *kickoff* yang mengoordinasikan keseluruhan kolaborasi.



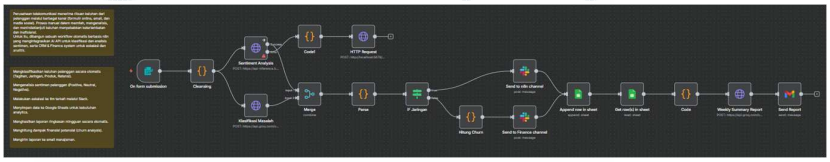
Gambar 3.7 Potongan kode proyek *Customer Support* menggunakan CrewAI

Selain proyek kolaborasi dua agen, pada minggu keempat peserta juga mengerjakan proyek kedua yang berkaitan dengan *Customer Support Simulation* dengan fokus pada penerapan konsep *multi-agent* dalam konteks industri telekomunikasi. Proyek ini disusun masih menggunakan *framework* CrewAI, namun dengan skenario simulasi layanan pelanggan untuk menggambarkan bagaimana sistem kecerdasan buatan dapat digunakan untuk menangani keluhan pelanggan secara otomatis. Dalam proyek ini dua agen utama didefinisikan, yaitu *Customer Support Agent* yang bertugas memahami dan mengklasifikasikan keluhan pelanggan, serta *Solution Analyst Agent* yang berperan memberikan solusi berdasarkan hasil analisis awal yang diterima dari agen pertama. Proses kolaborasi antar agen diatur secara hierarkis di dalam *Crew Instance*, di mana *output* dari *Customer Support Agent* dikirimkan sebagai *contextual input* bagi *Solution Analyst Agent* untuk menghasilkan rekomendasi penyelesaian yang relevan. CrewAI mengelola aliran komunikasi ini secara otomatis melalui mekanisme

context passing dan *task dependency*, sehingga menghasilkan sistem yang menyerupai alur kerja manusia di pusat layanan pelanggan nyata.

Melalui dua proyek tersebut, peserta memperoleh pengalaman yang menyeluruh mengani penerapan *multi-agent system* dalam konteks industri dan riset. *Framework* CrewAI terbukti mampu memfasilitasi *autonomous collaboration* antar agen dengan efisiensi tinggi, sekaligus memberikan fleksibilitas bagi pengembang untuk merancang peran, tujuan, dan interaksi antar agen secara modular. Peserta juga mempelajari praktik pengelolaan kredensial API eksternal melalui *file* ".env", serta mekanisme penjadwalan dan orkestrasi agen yang aman dan terstruktur. Dari sisi konsep, kegiatan ini menekankan pentingnya *reasoning coordination* dan *task delegation* dalam sistem AI, di mana setiap agen tidak hanya menyelesaikan tugasnya sendiri, tetapi juga berkontribusi terhadap pencapaian tujuan bersama melalui koordinasi kontekstual. Selain itu, melalui proyek *Customer Support Simulation*, peserta memahami bagaimana prinsip *Agentic* AI dapat diadaptasi untuk mendukung peningkatan efisiensi proses bisnis, khususnya dalam bidang layanan pelanggan digital.

3.3.1.5 Advanced n8n Workflows



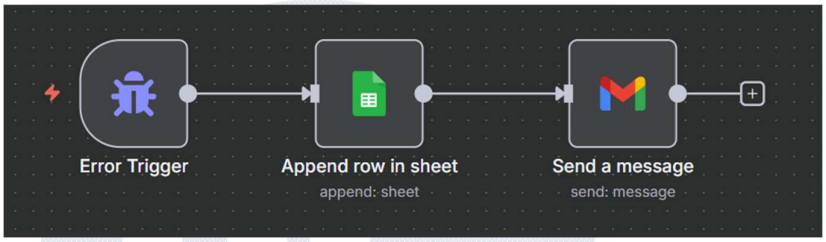
Gambar 3.8 *AI Customer Care System workflow*

Pada minggu kelima, kegiatan magang berfokus pada pembuatan *multi-step workflow* menggunakan platform n8n dengan pendekatan berbasis *node orchestration*. Proyek yang dibuat berkaitan dengan AI *Customer Care System*, yang berfungsi sebagai simulasi sistem otomatisasi layanan pelanggan dengan tahapan

berlapis, mulai dari pengumpulan data keluhan hingga pelaporan hasil analisis mingguan. *Workflow* ini dirancang sebagai *dummy project*, yang berarti seluruh data dan koneksi API yang digunakan hanya untuk kebutuhan simulasi dan tidak terkait dengan sistem produksi perusahaan. Proyek ini dibuat murni untuk keperluan *training* dan eksplorasi teknis. Alur kerja dimulai dengan *Form Trigger* untuk menerima *input* keluhan pelanggan, kemudian dilanjutkan dengan *data cleansing node* yang bertugas menormalkan teks agar siap diproses oleh model AI. Setelah itu, hasil keluhan dikirim ke dua jalur analisis paralel, yaitu *Sentiment Analysis* dan *Issue Classification*. *Node Merge* digunakan untuk menggabungkan hasil analisis tersebut sebelum dilakukan proses *parsing* dan logika bercabang (*conditional routing*) berdasarkan kategori masalah yang terdeteksi, seperti "Jaringan" atau "Tagihan". Setiap hasil kemudian dieksekusi ke *channel* berbeda di Slack dan disimpan dalam Google Sheets sebagai log historis.

Pada tahap selanjutnya, *workflow* ini mengimplementasikan integrasi langsung dengan API eksternal berbasis AI, yakni OpenAI API dan HuggingFace Interface API. *Node* OpenAI digunakan untuk melakukan klasifikasi teks, di mana model LLaMA 33.70B Versatile bertugas mengklasifikasikan teks keluhan pelanggan ke dalam kategori spesifik seperti Tagihan, Jaringan, Produk, dan Retensi, serta menambahkan skor kepercayaan (*confidence score*). Di sisi lain, API HuggingFace digunakan untuk analisis sentimen terhadap keluhan yang sama dengan model *Indonesian Roberta Base Sentiment Classifier*. Hasil dari kedua API tersebut digabungkan untuk menghasilkan analisis yang lebih kaya, mencakup dimensi emosional dan tematik dari setiap keluhan. Proses integrasi ini tidak hanya melibatkan pemanggilan API sederhana, tetapi juga penanganan data hasil (*JSON parsing*), pembersihan format respons, serta penggabungan hasil ke dalam struktur data ter standarisasi

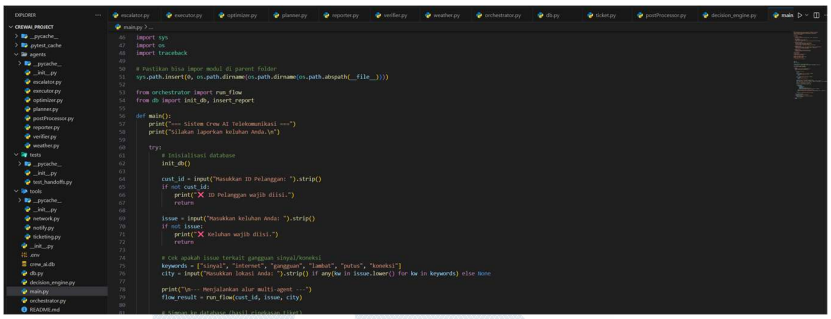
untuk keperluan laporan. Melalui simulasi ini, peserta magang memahami secara langsung bagaimana sistem AI modern beroperasi melalui integrasi RESTful API, serta bagaimana model NLP (*Natural Language Processing*) dapat digunakan untuk mendukung sistem otomatisasi layanan pelanggan.



Gambar 3.9 n8n *error handler workflow*

Selain mengembangkan *workflow* utama, peserta magang juga mempelajari pentingnya implementasi *error handling* dalam proses otomatisasi agar sistem dapat beroperasi dengan stabil. Untuk itu, dibangun *workflow* pendukung terpisah yang berfungsi sebagai sistem pemantau dan penanganan kesalahan otomatis. *Workflow* ini menggunakan *Webhook Trigger* untuk menerima pesan *error* dari dari *workflow* utama yang kemudian secara otomatis mencatat detail kesalahan ke Google Sheets pada tab "*Workflow Error Log*". informasi yang dicatat mencakup *timestamp*, *workflow name*, *node name*, *error message*, dan *error type*, sehingga dapat digunakan untuk analisis insiden di kemudian hari. Selain pencatatan otomatis, *workflow* ini juga mengirimkan notifikasi email ke pengelola sistem menggunakan Gmail API, lengkap dengan detail kesalahan dan langkah-langkah pemulihan dalam format HTML yang rapi. Proses ini memastikan bahwa setiap kegagalan eksekusi, misalnya akibat *timeout* API atau *invalid payload* segera tercatat dan terdokumentasi. Melalui praktik ini, peserta magang memahami pentingnya desain sistem yang kuat dan toleran terhadap kesalahan, di mana setiap kegagalan tidak menyebabkan keseluruhan alur berhenti total.

3.3.1.6 CrewAI *Deep Dive*



Gambar 3.10 Potongan kode *multi-role agent*

Pada minggu keenam kegiatan magang, fokus utama pekerjaan diarahkan pada tahap implementasi arsitektur *multi-role agent* dalam proyek CrewAI. Tahap ini bertujuan untuk menciptakan simulasi sistem kolaboratif yang melibatkan beberapa agen dengan peran berbeda dalam menyelesaikan suatu tugas, mulai dari perencanaan, pelaksanaan, hingga verifikasi hasil. Dalam implementasi awal, proyek ini masih berbentuk *dummy project*, yang berarti belum menggunakan data atau skenario nyata, melainkan hanya menyimulasikan alur kerja antar agen untuk menguji stabilitas sistem dan integrasi logika antar modul. Struktur proyek terdiri dari berbagai komponen seperti *planner.py, executor.py,* dan *verifier.py* yang masing-masing merepresentasikan agen dengan tanggung jawab berbeda. Melalui pendekatan ini, sistem mampu menggambarkan bagaimana agen-agen tersebut dapat berinteraksi secara terstruktur dan saling bertukar data dalam satu ekosistem yang terpusat.

Langkah awal yang dilakukan dalam implementasi ini adalah merancang *multi-role agent setup* yang mencakup definisi berbagai peran dan fungsi spesifik setiap agen. *Planner agent* bertugas menyusun rencana kerja berdasarkan masukan awal, kemudian *executor agent* melaksanakan rencana tersebut sesuai prosedur yang telah ditetapkan. Setelah proses eksekusi selesai, *verifier agent*

melakukan evaluasi terhadap hasil untuk memastikan bahwa *output* sesuai dengan kriteria yang diharapkan. Selain tiga agen utama tersebut, terdapat juga *optimizer agent* yang berfungsi memperbaiki hasil kerja apabila ditemukan ketidaksesuaian, serta *reporter* dan *postProcessor agent* yang menyusun laporan hasil kerja dalam format yang lebih mudah dibaca. Pengaturan koordinasi antar agen dilakukan melalui modul *orchestrator.py*, yang mengatur alur komunikasi, urutan eksekusi, dan pengiriman pesan antar agen. Pendekatan ini tidak hanya membantu menguji Interoperabilitas antar komponen, tetapi juga memastikan bahwa setiap agen memiliki tanggung jawab yang jelas, dan tidak tumpang tindih dengan agen lainnya. Dengan adanya struktur ini, sistem dapat menjalankan simulasi *multi-agent collaboration* yang menyerupai dinamika kerja tim dalam organisasi manusia, di mana setiap individu memiliki keahlian dan tanggung jawab yang terdistribusi secara efisien.

Setelah proses konfigurasi selesai, tahap berikutnya adalah melakukan pengujian terhadap mekanisme *task handoffs* dan *decision-making*. Pengujian ini dilakukan dengan menjalankan serangkaian skenario *dummy* di mana setiap agen menerima, memproses, dan menyerahkan hasil pekerjaannya kepada agen berikutnya. Misalnya, ketika agen *planner* menghasilkan rencana kerja, sistem menguji apakah data tersebut berhasil dikirim ke agen *executor* tanpa kehilangan informasi. Begitu pula ketika *executor* menyelesaikan tugasnya, sistem memastikan bahwa hasil tersebut dapat diverifikasi oleh *verifier* secara otomatis. Dalam proses ini, modul *decision_engine.py* berperan penting sebagai pusat logika yang menentukan kapan suatu agen perlu mengambil keputusan sendiri, menyerahkan keputusan kepada agen lain, atau melakukan eskalasi ke agen *escalator*. Pengujian ini dilakukan untuk memastikan sistem mampu menangani alur kerja kompleks dan

mengimplementasikan pengambilan keputusan berbasis kondisi tertentu secara efisien. Dari hasil uji tersebut dapat disimpulkan bahwa sistem mampu melakukan koordinasi antar agen dengan baik, bahkan dalam skenario sederhana sekalipun.

Hasil dari tahap implementasi dan pengujian ini menunjukkan bahwa sistem *multi-agent* yang dibangun telah berfungsi sesuai rancangan, meskipun dalam skala simulasi. Alur komunikasi antar agen berjalan dengan lancar, dan setiap agen mampu menjalankan peran yang telah ditentukan. Meskipun demikian, karena proyek ini masih merupakan *dummy project*, data yang digunakan bersifat statis dan terbatas pada skenario uji sederhana. Dalam pengembangan berikutnya, sistem ini berpotensi untuk dikembangkan menjadi platform otomatisasi kerja kolaboratif berbasis *intelligent agent* yang dapat beradaptasi dengan konteks nyata, misalnya pada bidang periklanan digital atau manajemen proyek berbasis AI. Dengan demikian, tahapan di minggu keenam ini menjadi fondasi penting untuk memahami dinamika koordinasi antar agen dan menjadi dasar bagi pengembangan sistem *multi-agent* yang lebih kompleks di tahap selanjutnya.

3.3.1.7 LangChain

Minggu ketujuh kegiatan magang, fokus utama diarahkan pada tahap eksplorasi dan implementasi *framework* LangChain serta AutoGen sebagai fondasi dalam pengembangan *agent logic* yang lebih dinamis. Tujuan utama dari tahap ini adalah untuk memahami bagaimana kedua *framework* tersebut dapat digunakan dalam membangun sistem berbasis agen yang memiliki kemampuan memori, *tools*, dan rantai penalaran (*reasoning chains*). Dalam proyek ini, penerapan dilakukan pada konteks simulasi percakapan layanan pelanggan (*customer support simulation*) yang masih berbentuk *dummy project*. Artinya, seluruh data konteks percakapan

dan basis pengetahuan yang digunakan bersifat fiktif hanya untuk menggambarkan alur kerja sistem agen dalam skenario yang realistis. Pendekatan ini memungkinkan eksplorasi konsep teknis seperti penyimpanan memori percakapan, pengambilan informasi dari basis data semu, serta integrasi logika pengambilan keputusan berbasis kondisi. Tahap ini menjadi penting karena LangChain merupakan teknologi inti dalam pengembangan agen AI otonom modern, yang memungkinkan sistem untuk berpikir, berinteraksi, dan menyesuaikan respons secara kontekstual. Implementasi yang dilakukan dalam proyek *dummy* ini menjadi fondasi awal dalam memahami bagaimana arsitektur agen multifungsi dapat dikembangkan ke dalam sistem nyata di masa depan.

```
# Initialize memory
memory = ConversationBufferWindowMemory(
    k=5,  # Keep last 5 exchanges
    memory_key="chat_history",
    return_messages=True
)
```

Gambar 3.11 Potongan kode inisialisasi memori

Langkah awal dari implementasi dilakukan dengan membangun struktur memori percakapan menggunakan *ConversationBufferWindowMemory* dari pustaka LangChain. Komponen ini berfungsi menyimpan konteks dari interaksi sebelumnya antara pengguna dan agen agar sistem mampu memberikan jawaban yang konsisten dan relevan terhadap percakapan yang sedang berlangsung. Dalam proyek ini, memori digunakan untuk melacak pertanyaan dan jawaban seputar tagihan pelanggan, promo layanan, dan detail paket internet dari perusahaan fiktif yang digunakan dalam simulasi. Dengan adanya mekanisme memori ini, agen dapat memahami konteks seperti “pelanggan ini sebelumnya menanyakan tagihan bulan lalu” dan memberikan

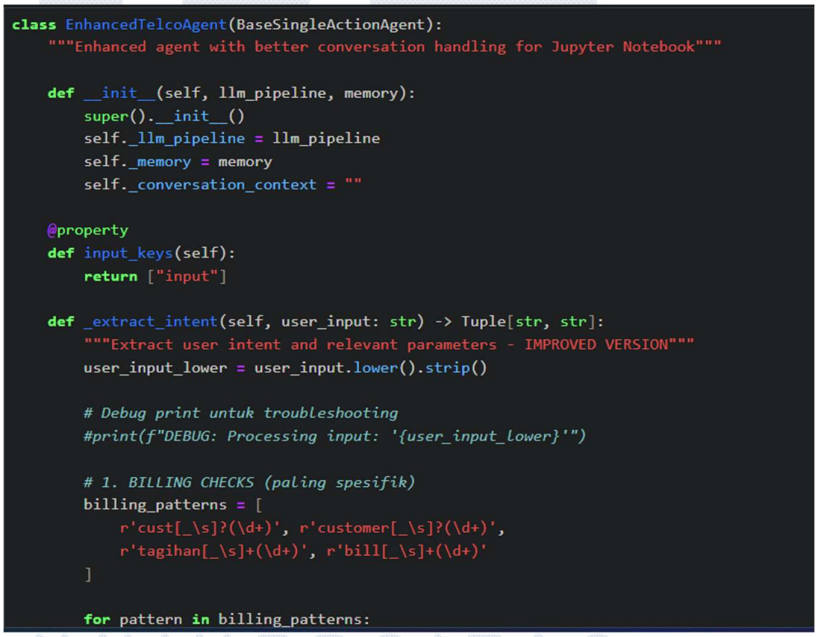
tanggapan yang sesuai tanpa mengulang informasi. Walaupun sistem belum tersambung ke sumber data eksternal yang sesungguhnya, pengujian dengan *mock* data seperti *BILLING_DB* menunjukkan bagaimana konsep memori dapat bekerja secara efektif dalam menjaga kontinuitas percakapan. Selain itu, implementasi in juga menunjukkan bagaimana LangChain memfasilitasi arsitektur agen modular yang dapat dikembangkan lebih lanjut, misalnya dengan menambahkan *vector memory* atau *long-term context retriever* di masa mendatang.

```
def check_billing(customer_id: str) -> str:
    """Enhanced billing check with more details"""
    customer_id = customer_id.lower().replace(" ", "_").strip()

    # Handle different input formats
    if not customer_id.startswith("cust"):
        if customer_id.isdigit():
            customer_id = f"cust_{customer_id.zfill(3)}"
        else:
            customer_id = f"cust_{customer_id}"

    data = BILLING_DB.get(customer_id)
    if not data:
        return "❌ Data billing tidak ditemukan. Pastikan ID customer benar (format: cust_001, cust_002, dll)."

    status_emoji = {"paid": "✅", "pending": "⏳", "overdue": "🚨"}
    status_text = {"paid": "Lunas", "pending": "Menunggu Pembayaran", "overdue": "Terlambat"}

    result = f"""💳 **Data Billing - {data['name']}**
📊 Paket: {data['package']}
💰 Tagihan: Rp {data['last_bill']:,}
📅 Jatuh Tempo: {data['due_date']}
{status_emoji.get(data['status'], '❓')} Status: {status_text.get(data['status'], 'Unknown')}"""

    if data['status'] == 'overdue':
        result += "\n⚠️ Segera lakukan pembayaran untuk menghindari pemutusan layanan."
    elif data['status'] == 'pending':
        result += "\n💡 Pembayaran dapat dilakukan melalui berbagai channel yang tersedia."

    return result
```

Gambar 3.12 Potongan kode integrasi *tools*

Setelah memori berhasil diimplementasikan, tahap berikutnya adalah integrasi *tools* atau fungsi bantu yang memungkinkan agen melakukan aksi tertentu sesuai kebutuhan percakapan. Dalam proyek ini, beberapa *tools* utama didefinisikan kebutuhan percakapan. Dalam proyek ini, beberapa *tools* utama didefinisikan seperti *check_billing()* untuk mengecek informasi tagihan pelanggan, serta *get_knowledge()* untuk mengambil informasi harga dan promo dari basis pengetahuan semu proyek. Masing-masing fungsi dirancang dengan logika yang realistis untuk menunjukkan bagaimana agen dapat menggunakan alat bantu untuk

menyelesaikan permintaan pengguna. Misalnya, ketika pengguna bertanya "berapa harga internet 30 Mbps?", sistem akan mengeksekusi *get_knowledge()* untuk mencari kecocokan kata kunci dan mengembalikan jawaban yang tepat dari basis pengetahuan *dummy*. Konsep ini meniru bagaimana agen Ai di dunia nyata mengakses API eksternal, membaca *database*, atau memanggil fungsi sistem tertentu sesuai konteks percakapan. Dengan demikian, integrasi *tools* menjadi komponen penting dalam membangun agen yang tidak hanya bersifat reaktif, tetapi juga mampu melakukan tindakan berbasis instruksi.



```python
class EnhancedTelcoAgent(BaseSingleActionAgent):
    """Enhanced agent with better conversation handling for Jupyter Notebook"""

    def __init__(self, llm_pipeline, memory):
        super().__init__()
        self._llm_pipeline = llm_pipeline
        self._memory = memory
        self._conversation_context = ""

    @property
    def input_keys(self):
        return ["input"]

    def _extract_intent(self, user_input: str) -> Tuple[str, str]:
        """Extract user intent and relevant parameters - IMPROVED VERSION"""
        user_input_lower = user_input.lower().strip()

        # Debug print untuk troubleshooting
        #print(f"DEBUG: Processing input: '{user_input_lower}'")

        # 1. BILLING CHECKS (paling spesifik)
        billing_patterns = [
            r'cust[_\s]?(\d+)', r'customer[_\s]?(\d+)',
            r'tagihan[_\s]+(\d+)', r'bill[_\s]+(\d+)'
        ]

        for pattern in billing_patterns:
```

Gambar 3.13 Potongan kode *reasoning chains*

Tahap berikutnya adalah penerapan *reasoning chains*, yang menjadi logika berantai yang memungkinkan agen untuk menganalisis konteks, menentukan langkah yang tepat, dan menyusun jawaban yang logis berdasarkan urutan proses berpikir. Dalam proyek *dummy* ini, *reasoning chains* diterapkan melalui kombinasi struktur *if-else* dan pemanggilan berurutan terhadap *tools* dan relevan. Sebagai contoh, ketika sistem menerima pertanyaan

pelanggan tentang status pembayaran, agen akan memulai dari langkah identifikasi konteks pertanyaan, lalu memanggil fungsi *check_billing()*, dan akhirnya menghasilkan tanggapan yang disesuaikan dengan kondisi yang ditemukan (misalnya "status pembayaran Anda masih tertunda"). Meskipun alur ini sederhana dan belum melibatkan model pembelajaran mendalam, struktur yang digunakan telah mencerminkan prinsip dasar *chain-of-thought reasoning* yang banyak digunakan dalam pengembangan agen AI cerdas. LangChain memfasilitasi hal ini dengan mendefinisikan *pipeline* logika yang sistematis, sehingga agen dapat meniru proses berpikir manusia dalam menyelesaikan permasalahan bertahap.

Selain itu, proyek di minggu ketujuh ini juga memperkenalkan konsep AutoGen, sebuah *framework* yang memungkinkan orkestrasi antar agen AI dengan peran berbeda untuk berinteraksi dalam satu lingkungan kerja. Dalam konteks simulasi ini, AutoGen digunakan secara terbatas untuk menunjukkan bagaimana satu agen dapat berkomunikasi dengan agen lain melalui skenario percakapan yang diatur. Walaupun implementasinya masih *dummy*, pendekatan ini membantu memperkenalkan konsep kolaborasi *multi-agent* dalam konteks percakapan, di mana setiap agen memiliki memori dan logika penalaran sendiri. Dalam tahap berikutnya, pendekatan AutoGen dapat dikembangkan untuk menciptakan sistem layanan pelanggan otomatis yang lebih kompleks, di mana agen dapat bernegosiasi, berbagi konteks, dan mengoptimalkan keputusan berdasarkan hasil percakapan sebelumnya. Dengan demikian, tahap ini memperkuat pemahaman mengenai interaksi antar agen serta bagaimana komunikasi tersebut dapat diatur menggunakan *framework* modern seperti AutoGen.

Hasil dari kegiatan implementasi dan eksplorasi pada minggu ketujuh menunjukkan bahwa sistem berbasis LangChain dan

AutoGen mampu menyimulasikan logika percakapan dan pengambilan keputusan yang cukup realistis meskipun masih dalam skala *dummy*. Agen dapat mengingat konteks percakapan, memanggil fungsi tertentu sesuai kebutuhan, dan menyusun respons yang relevan terhadap *input* pengguna. Dari sisi teknis, integrasi memori dan *tools* berjalan dengan stabil tanpa *error* besar, menunjukkan bahwa arsitektur modular yang digunakan telah sesuai dengan prinsip desain sistem AI modern. Namun, karena proyek ini belum menggunakan data asli atau model pembelajaran adaptif, performa sistem masih bergantung pada logika statis dan basis pengetahuan yang terbatas. Walau demikian, hasil ini memberikan fondasi kuat untuk pengembangan sistem agen yang lebih canggih dan kontekstual di tahap selanjutnya. Dengan pemahaman yang diperoleh dari minggu ketujuh ini, peserta magang telah memiliki gambaran menyeluruh mengenai bagaimana agen AI dapat dikembangkan, diatur, dan diperluas menggunakan *framework* generatif modern untuk berbagai skenario bisnis dan operasional.

3.3.1.8 Evaluation & Benchmarking

Tabel 3.2 Matriks Perbandingan *Tools*

Kriteria	N8n	CrewAI	LangChain
Ease of Use	Sangat mudah (*No-code/Low-code*), dapat digunakan oleh tim non-teknis. Antarmuka visual *drag-and-drop* sangat ideal untuk alur kerja operasional.	Cukup mudah (Python), tetapi memerlukan pemahaman tentang konsep *agent*. Dokumentasinya masih terbatas dan memerlukan eksperimen lebih lanjut.	Kompleks (pemrograman), memerlukan pemahaman konsep yang mendalam serta penggunaan LLM (*Large Language Models*).
Scalability	Moderat. Cocok untuk menangani ribuan keluhan harian, namun performa mungkin menurun di bawah beban kerja AI yang berat.	Rendah hingga moderat. Mendukung orkestrasi *agent*, namun belum cukup matang untuk integrasi produksi skala besar (*enterprise*).	Sangat tinggi. Mampu menangani jutaan keluhan dengan integrasi LLM API dan *vector database*. Cocok untuk beban

			kerja NLP (*Natural Language Processing*) skala besar.
Deployment	Mudah diterapkan melalui *cloud* atau Docker. Konsumsi sumber daya rendah."	Memerlukan lingkungan Python dan pustaka (*library*) tambahan. Cocok untuk *cloud* dan laboratorium litbang (R&D).	Memerlukan infrastruktur yang kuat (LLM, API, vector DB). Cocok untuk AI skala *cloud*.
Flexibility	Bagus untuk orkestrasi operasional. Berfungsi sebagai penghubung antar sistem (CRM, basis data, notifikasi). Namun, masih terbatas untuk logika AI yang kompleks.	Sangat baik untuk *multi-agent setups*. Cocok untuk mengoordinasikan agen-agen AI dengan peran yang berbeda-beda. Namun, ekosistemnya masih relatif kecil, sehingga memerlukan upaya lebih untuk integrasi eksternal.	Sangat fleksibel. Cocok untuk berbagai skenario AI seperti analisis sentimen, klasifikasi, RAG, dan lainnya. Dapat berintegrasi dengan ekosistem AI yang luas
Kelebihan	- Mudah dipelajari dan diimplementasikan. - Dilengkapi dengan banyak penghubung (*built-in connectors*) bawaan. - Cocok sebagai perekat antar sistem.	- Natural untuk orkestrasi tim agen. - Lebih ringan daripada LangChain untuk pembuatan prototipe. - Modular dan fleksibel.	- Ekosistem yang paling matang. - Cocok untuk sistem AI yang kompleks. - Skalabilitas tinggi.
Kekurangan	- Dukungan terbatas untuk penalaran (*reasoning*) AI secara langsung. - Skalabilitas terbatas untuk tugas-tugas AI dengan beban tinggi. - Memerlukan integrasi eksternal untuk analisis AI.	- Komunitas kecil dan dokumentasi yang terbatas. - Sulit untuk diintegrasikan ke dalam sistem operasional. - Belum stabil untuk produksi skala besar (*enterprise*).	- Kurva pembelajaran yang curam (memerlukan kemampuan pemrograman). - Pengaturan (setup) yang lebih kompleks. - Memerlukan integrasi tambahan untuk alur kerja (*workflow*) operasional.

Pada minggu kedelapan kegiatan magang berfokus pada tahap *evaluation and benchmarking*, yang bertujuan untuk menilai dan membandingkan tiga alat pengembangan kecerdasan buatan populer, yaitu n8n, CrewAI, dan LangCHain. Ketiga platform ini dipilih karena mewakili tiga pendekatan berbeda dalam membangun sistem otomatisasi dan orkestrasi berbasis AI, mulai dari *no-code orchestration* hingga *agentic AI programming*. Evaluasi dilakukan berdasarkan tiga kriteria utama, yaitu *ease of use, scalability, dan flexibility*, yang dianggap paling relevan dengan konteks pengembangan sistem penanganan keluhan pelanggan di industri telekomunikasi. Dalam proses *benchmarking*, setiap alat diuji dari sisi kemudahan implementasi, efisiensi dalam menangani beban kerja besar, serta kemampuan adaptasi terhadap berbagai skenario integrasi AI. Berdasarkan analisis yang dilakukan, ketiga alat memiliki karakteristik dan keunggulan yang berbeda sesuai dengan tujuan penggunaannya. N8n unggul dalam kemudahan penggunaan dengan antarmuka visual yang intuitif, CrewAI menonjol dalam orkestrasi *multi-role agent* dengan pendekatan modular, sedangkan LangChain menunjukkan keunggulan dalam fleksibilitas dan skalabilitas untuk sistem AI berskala besar. Hasil evaluasi ini disajikan dalam bentuk *comparison matrix* dan *visual chart* untuk memberikan gambaran kuantitatif mengenai performa relatif masing-masing alat terhadap kriteria yang diuji.

Aspek kemudahan penggunaan (*ease of use*) menjadi kriteria pertama dalam proses evaluasi, karena faktor ini menentukan seberapa cepat suatu alat dapat diadopsi oleh tim dengan tingkat keahlian teknis yang berbeda. Berdasarkan hasil perbandingan, n8n memperoleh skor tertinggi dalam aspek ini karena dirancang sebagai platform *no-code/low-code* yang dapat digunakan oleh pengguna

non-teknis. Dengan antarmuka *drag-and-drop visual*, pengguna dapat membangun alur kerja (*workflow*) yang kompleks tanpa perlu menulis kode secara langsung, menjadikannya sangat ideal untuk tim operasional yang ingin mengotomatisasi proses bisnis dengan cepat. Di sisi lain CrewAI memiliki tingkat kemudahan yang moderat karena membutuhkan pemahaman tentang konsep *agent orchestration* dan pemrograman dasar Python. Walaupun dokumentasi CrewAI masih terbatas, *framework* ini memberikan fleksibilitas yang lebih besar bagi tim riset yang ingin bereksperimen dengan sistem *multi-agent*. Sementara itu, LangChain dikategorikan sebagai alat yang paling kompleks untuk digunakan karena memerlukan kemampuan pemrograman lanjutan serta pemahaman mendalam tentang LLM, API, dan arsitektur *pipeline AI*. Walaupun memiliki *learning curve* yang tinggi, LangChain menawarkan kontrol penuh terhadap desain sistem dan integrasi logika. Berdasarkan temuan ini, dapat disimpulkan bahwa n8n lebih cocok untuk *non-technical operators*, CrewAI lebih ideal bagi *AI researchers* atau *R&D engineers*, dan LangChain paling sesuai untuk *software developers* yang membangun sistem AI canggih secara mendalam.

Dari sisi skalabilitas (*scalability*), ketiga *tools* menunjukkan performa yang berbeda sesuai dengan pendekatan teknis yang mereka gunakan. LangChain menempati posisi teratas karena dirancang untuk mendukung beban kerja skala besar melalui integrasi dengan *vector databases, retrieval-augmented generation (RAG),* dan API LLM yang dapat memproses jutaan entri secara paralel. *Framework* ini banyak digunakan dalam sistem NLP korporasi, seperti analisis sentimen berskala besar dan klasifikasi teks. Meskipun menjadi *tools* yang sangat efisien untuk otomatisasi operasional, n8n memiliki skalabilitas yang moderat karena dirancang lebih sebagai *workflow orchestrator* daripada *AI*

computation engine. Dalam pengujian, n8n mampu menangani ribuan keluhan pelanggan per hari dengan stabilitas tinggi, namun performanya menurun signifikan saat diintegrasikan dengan model AI berukuran besar karena keterbatasan dalam pengelolaan beban pemrosesan paralel. Selain itu, CrewAI masih berada dalam tahap pengembangan awal dalam hal skalabilitas. Meskipun *framework* ini mendukung orkestrasi beberapa agen sekaligus, stabilitasnya untuk implementasi berskala *enterprise* masih belum optimal, terutama karena belum memiliki mekanisme *load balancing* dan *distributed memory management* yang matang.

Kriteria berikutnya adalah fleksibilitas (*flexibility*) yang menilai seberapa luas suatu alat dapat digunakan dalam berbagai konteks dan seberapa mudah alat tersebut diintegrasikan dengan sistem eksternal. Berdasarkan hasil evaluasi, LangChain menonjol dalam aspek ini karena memiliki ekosistem yang luas dan mendukung berbagai jenis skenario AI, mulai dari analisis sentimen, klasifikasi teks, hingga RAG dan *contextual chatbot development*. *Framework* ini juga kompatibel dengan berbagai *vector databases* seperti Pinecone, FAISS, serta API besar seperti OpenAI dan HuggingFace, menjadikannya alat yang sangat fleksibel untuk integrasi lintas platform. CrewAI menempati posisi kedua dalam aspek fleksibilitas CrewAI terletak pada kemampuannya untuk mengatur peran agen dengan cara yang modular, seperti menggabungkan *Planner*, *Executor*, dan *Verifier* dalam satu *pipeline*. Namun, keterbatasan dokumentasi dan ukuran komunitas yang masih kecil membuat integrasi eksternal memerlukan upaya tambahan. Sementara itu, n8n tetap unggul dalam fleksibilitas operasional, terutama karena kemampuannya untuk berfungsi sebagai penghubung antar sistem (*system connector*) seperti CRM, email, *database*, dan platform notifikasi. Namun, fleksibiltasnya

dalam konteks logika AI masih terbatas, karena n8n tidak memiliki fungsi *AI reasoning* internal yang setara dengan dua *tools* lainnya.

Secara keseluruhan, hasil *benchmarking* menunjukkan bahwa ketiga *tools* tersebut memiliki keunggulan yang saling melengkapi dan dapat dipilih sesuai dengan kebutuhan serta tingkat kematangan organisasi. N8n menjadi pilihan terbaik untuk tim operasional yang memerlukan otomatisasi cepat tanpa pengetahuan teknis mendalam, berkat antarmuka visual dan ketersediaan ratusan konektor bawaan. Kelemahan utama n8n adalah keterbatasannya dalam menangani tugas-tugas AI yang berat, karena tidak memiliki *native reasoning engine* dan harus bergantung pada integrasi eksternal seperti OpenAI API. Selain itu, CrewAI memberikan nilai tambah bagi tim penelitian dan pengembang yang ingin mengeksplorasi sistem kolaborasi antar agen AI. *Framework* ini ringan dan modular, tetapi masih menghadapi tantangan dalam dokumentasi, stabilitas, dan adopsi komunitas. LangChain menempati posisi teratas dalam hal kemampuan teknis dan ekosistem pengembang, menjadikannya pilihan ideal bagi perusahaan atau tim AI yang membangun sistem berskala besar dan kompleks. namun, tingkat kesulitannya yang tingi dan kebutuhan infrastruktur yang besar menjadi hambatan utama bagi organisasi yang baru memulai. Berdasarkan hasil analisis dari dokumen *Tools Comparison Matrix*, disimpulkan bahwa n8n cocok untuk tim operasional, CrewAI untuk tim riset dan pengembangan, sedangkan LangChain untuk tim teknis tingkat lanjut yang berfokus pada arsitektur AI *enterprise*. Dengan demikian, hasil evaluasi ini memberikan gambaran komprehensif tentang posisi strategis masing-masing alat dalam ekosistem pengembangan AI modern dan menjadi dasar penting untuk pengambilan keputusan teknologi di proyek-proyek selanjutnya.

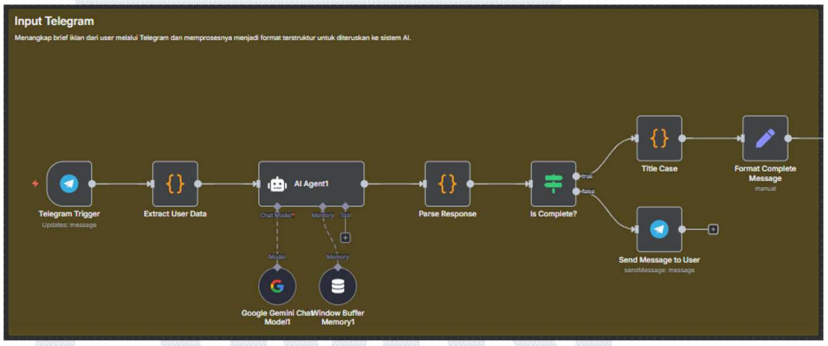
3.3.1.9 Project Ideation

Fokus utama pada minggu kesembilan kegiatan magang diarahkan pada *project ideation*, yaitu proses perumusan ide proyek yang akan menjadi dasar kegiatan pengembangan selanjutnya. Tahap ini bertujuan untuk mengidentifikasi peluang penerapan konsep *Agentic* AI dan *automation workflow* dalam konteks yang relevan dengan transformasi digital. Proses *brainstorming* dilakukan secara kolaboratif dengan pembimbing lapangan (mentor) melalui sesi diskusi daring yang dilakukan menggunakan platform komunikasi WhatsApp. Selama proses *brainstorming*, peserta diminta untuk mengajukan beberapa gagasan proyek yang inovatif, dengan mempertimbangkan aspek teknis, relevansi terhadap kebutuhan industri, serta potensi dampaknya terhadap kebutuhan industri, serta potensi dampaknya terhadap efisiensi kerja dan pengalaman pengguna. Dari hasil diskusi awal muncul beberapa ide proyek, antara lain *AI-based document analyzer, agentic report assistant, customer support automation,* hingga *AI-driven ads generation system*. Setiap ide kemudian dianalisis berdasarkan tingkat kompleksitas, ketersediaan sumber daya, serta kesesuaian dengan tujuan pembelajaran magang yang berfokus pada eksplorasi teknologi *Agentic AI* dan *workflow automation*.

Setelah melalui proses evaluasi bersama mentor, disepakati bahwa ide proyek yang akan dikembangkan lebih lanjut adalah *Automation Ads Generation*, yaitu sistem yang dirancang untuk secara otomatis menghasilkan konten iklan digital dan mempublikasikannya ke platform media sosial. Proyek ini dinilai paling relevan dengan tujuan eksplorasi karena menggabungkan dua bidang utama yang menjadi fokus kegiatan magang. Dalam konsep awalnya, sistem ini diharapkan mampu menghasilkan teks promosi dan visual poster menggunakan model AI kreatif hingga melakukan publikasi otomatis ke platform media sosial seperti Instagram dan

Facebook menggunakan *workflow automation tool* seperti n8n. Tahap *ideation* ini juga mencakup eksplorasi masalah, identifikasi kebutuhan data, serta pemetaan *tool* dan *framework* yang akan digunakan, seperti n8n untuk otomatisasi, OpenAI API untuk melakukan *brand context research* Gemini API untuk melakukan *copywriting* yang menghasilkan konten visual. Mentor juga memberikan arahan agar proyek ini tetap dijalankan sebagai *dummy project* untuk eksplorasi, yang artinya seluruh data, konten, dan akun media sosial yang digunakan bersifat palsu dan tidak terhubung dengan aset resmi perusahaan. Dengan demikian proses pengembangan dapat dilakukan secara bebas namun tetap aman dari risiko pelanggaran privasi atau kebijakan perusahaan.

3.3.1.10 *Agents*, Data, & API *Integration*



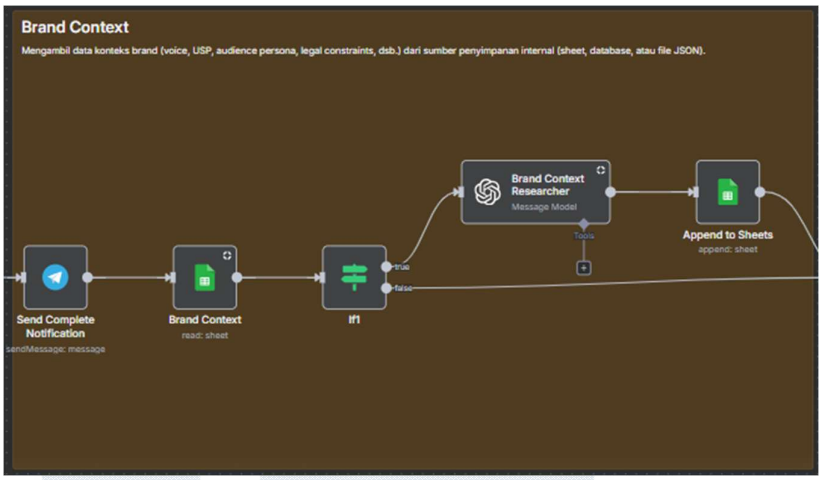
Gambar 3.14 Modul *Input* Telegram

Pada minggu kesepuluh, kegiatan magang difokuskan pada tahap implementasi alur kerja *end-to-end* dasar untuk sistem *Automation Ads Generation*, dengan titik awal berupa perancangan *user input flow* menggunakan Telegram sebagai *trigger* utama. Pemilihan Telegram dilakukan karena platform ini mendukung integrasi melalui *webhook* secara mudah, memiliki antarmuka percakapan yang sederhana, serta mendukung pengelolaan sesi per pengguna yang memudahkan proses pengumpulan *brief* iklan secara bertahap. Alur *input* dirancang untuk mengumpulkan data struktural penting, seperti nama merek, produk, target audiens, nada

komunikasi (*tone*), tujuan kampanye, platform publikasi, serta *creative brief* dari pengguna. Semua data tersebut dikumpulkan melalui serangkaian pertanyaan bertahap yang dilakukan oleh AI *Agent* sehingga interaksi terasa alami dan terarah. *Agent* ini memanfaatkan *session key* berbasis *Chat ID* agar status pengisian *brief* tersimpan di *buffer memory*, sehingga percakapan dilanjutkan tanpa kehilangan konteks apabila pengguna berhenti di tengah proses. Setelah seluruh data terisi lengkap, respons pengguna diproses oleh *node Parse Response*, yang menormalkan keluaran model bahasa (*LLM output*) menjadi format JSON terstruktur sehingga dapat diteruskan ke tahap berikutnya secara otomatis.

Implementasi *agent* yang menangani *input* pengguna bersifat *hybrid*, menggabungkan antara kemampuan percakapan berbasis LLM (Gemini) dengan logika pemrosesan data menggunakan *function node*. Agen percakapan bertugas mengekstraksi niat dan isi *brief* pengguna, sedangkan *function node* berfungsi untuk melakukan pembersihan dan validasi struktur data JSON yang dihasilkan. Secara teknis, akur kerja (*pipeline*) dimulai dari *Telegram Trigger, Extract User Data, AI Agent, Parse Response, Format Complete Message,* hingga notifikasi akhir dikirimkan ke pengguna bahwa data berhasil dikumpulkan dan siap diproses. *Prompt* pada *AI Agent* dirancang untuk memastikan bahwa *output* berbentuk JSON valid ketika seluruh data sudah terkumpul, sehingga dapat diurai secara pasti oleh sistem. Jika penguraian gagal, *node if* dan *error handler* otomatis akan meminta klarifikasi tambahan atau mengirim notifikasi kesalahan. Untuk menjaga kesinambungan percakapan, digunakan *Window Buffer Memory* (*context window*) agar *agent* dapat mengingat konteks interaksi beberapa langkah terakhir, sehingga proses revisi atau penambahan data tidak memerlukan pengulangan *input* dasar. Desain ini menyeimbangkan antara kemudahan penggunaan bagi pengguna

dan ketepatan struktur data bagi sistem, menjadikan alur kerja lebih efisien dan adaptif terhadap dinamika percakapan.



Gambar 3.15 Modul *Brand Context*

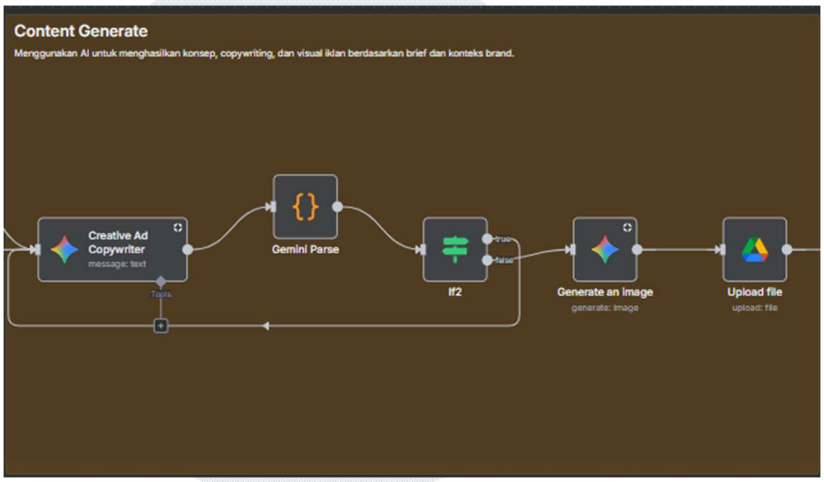
Setelah *input* data tervalidasi, alur kerja berlanjut menuju modul *Brand Context Retrieval*. Proses ini merupakan komponen penting dalam keseluruhan proses otomatisasi yang bertujuan untuk menjamin kelengkapan informasi identitas merek sebelum sistem menghasilkan konten. Setelah seluruh data awal yang diperlukan telah dipastikan lengkap, *workflow* memulai eksekusi melalui *node Send Complete Notification* sebagai penanda bahwa tahap *input* telah selesai. Selanjutnya, *node Brand Context* melakukan proses pembacaan terhadap basis data internal, yaitu Google Sheets untuk memeriksa apakah data konteks merek seperti *brand voice*, *Unique Selling Proposition* (USP), *audience persona*, karakter gaya bahasa, hingga batasan komunikasi *brand* telah tersedia sebelumnya. Proses pembacaan ini dilakukan dengan mencocokkan nama merek yang diberikan pengguna sebagai parameter kunci. Informasi hasil pembacaan kemudian diteruskan menuju *node IF* sebagai titik pengambilan keputusan utama dalam alur kerja.

Node IF memainkan fungsi strategis karena menentukan apakah alur kerja akan langsung menuju ke *content generate* atau

melakukan riset terlebih dahulu. Jika hasil evaluasi logis pada *node* tersebut bernilai *false*, maka hal ini menunjukkan bahwa data konteks merek telah tersedia di dalam *database*, sehingga sistem dapat langsung melanjutkan proses ke tahap pembuatan konten tanpa harus melakukan analisis atau pencarian tambahan. Kondisi ini memberikan efisiensi signifikan dalam hal waktu dan pemanfaatan sumber daya komputasi, terutama bagi merek yang sudah sering dipakai dalam proses kreatif. Sebaliknya, apabila hasil evaluasi bernilai *true*, maka konteks merek dinyatakan belum tersedia dan alur kerja secara otomatis dialihkan menuju *node Brand Context Researcher*. *Node* ini menggunakan model bahasa berbasis kecerdasan buatan yang mampu melakukan riset otonom dengan mengidentifikasi ciri khas, nilai inti, posisi pasar, dan karakter komunikasi merek berdasarkan referensi yang relevan.

Hasil dari proses riset tidak langsung digunakan, melainkan disimpan kembali ke dalam *database* Google Sheets melalui *node Append to sheets*. Tahap penyimpanan balik seperti ini memiliki peranan yang sangat krusial karena menciptakan siklus pembelajaran adaptif yang memungkinkan sistem menjadi semakin lengkap dan efisien seiring waktu. Dengan bertambahnya data referensi yang tersimpan, sistem mampu mengenali merek yang pernah diproses sebelumnya dan melakukan *retrieval* secara instan pada permintaan berikutnya. Dengan demikian, *Brand Context Retrieval* tidak hanya berfungsi sebagai langkah persiapan teknis sebelum pembuatan konten, tetapi juga sebagai mekanisme peningkatan pengetahuan sistem secara bertahap yang berkelanjutan. Ketika konteks merek telah dipastikan tersedia baik dari pembacaan *database* maupun hasil riset yang baru dilakukan, alur kerja kemudian melanjutkan ke tahap *Content Generation*. Pada fase ini, konteks merek yang telah disusun atau diperbaharui akan digabungkan dengan *brief* kreatif dari pengguna untuk membentuk

rich prompt yang berkaitan. *Prompt* tersebut kemudian diproses oleh modul kreatif berbasis LLM untuk menghasilkan *copywriting*, pesan pemasaran, serta visual yang sesuai identitas merek. Dengan struktur alur yang ternormalisasi dan adaptif ini, sistem tidak hanya mampu mempertahankan konsistensi *brand*, tetapi juga mendukung skala produksi konten yang besar secara efisien dan berkelanjutan.



Gambar 3.16 Modul *Content Generation*

Tahap *Content Generation* menjadi inti dari sistem *Automation Ads Generation*, di mana proses kreatif dilakukan secara otomatis untuk menghasilkan konsep iklan, naskah (*copywriting*), dan visual yang sesuai dengan konteks merek serta *brief* pengguna. Proses ini diawali oleh *node Creative Ad Copywriter*, yang menggunakan model LLM (Gemini) untuk menghasilkan keluaran teks dalam format terstruktur. *Node* ini menerima *input* yang telah digabungkan antara konteks merek dan *brief* pengguna, kemudian membentuk *prompt* yang kaya (*rich prompt*) berisi instruksi kreatif untuk menghasilkan tiga komponen utama, yaitu *headline*, *caption*, dan *visualIdea*. Model bahasa dirancang agar menghasilkan *output* dalam format JSON, yang memungkinkan hasil teks diolah lebih lanjut oleh sistem tanpa kehilangan struktur. Setiap komponen memiliki fungsi spesifik. *Headline* berfungsi sebagai judul utama

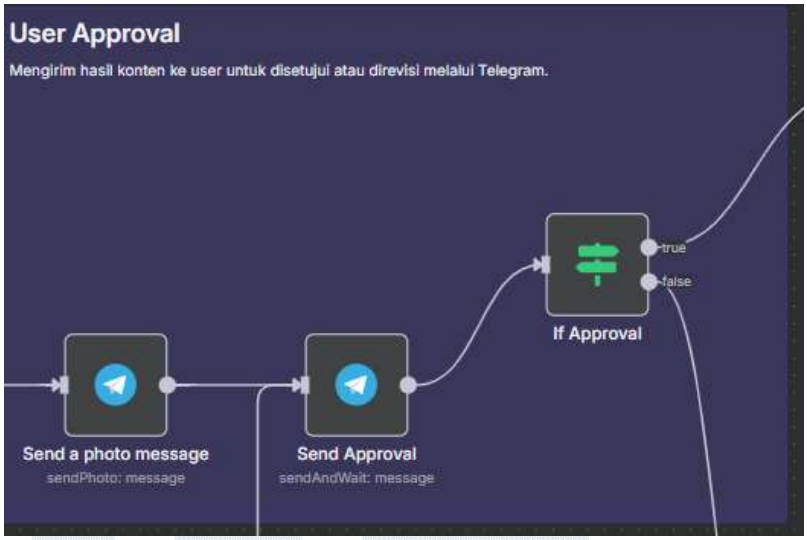
iklan, *caption* sebagai narasi pendukung dengan gaya bahasa yang sesuai dengan nada (*tone*) *brand*, dan *visualIdea* sebagai deskripsi konsep visual yang akan digunakan untuk pembuatan gambar. Dengan pendekatan ini, sistem mampu meniru alur tim kreatif manusia, di mana ide dan pesan utama diturunkan dari data strategis yang telah dikumpulkan sebelumnya.

Hasil *output* dari LLM kemudian diteruskan ke *node* Gemini Parse, yang berfungsi untuk menormalkan dan memvalidasi format JSON dari hasil teks yang dihasilkan oleh *Creative Ad Copywriter*. *Node* ini memastikan setiap atribut seperti *headline*, *caption*, dan *visualIdea* terbaca dalam format yang benar sehingga dapat diteruskan dengan aman ke tahap berikutnya. Apabila format JSON tidak valid atau model gagal mematuhi struktur *output* yang diharapkan, *node if* digunakan untuk melakukan percabangan logika (*conditional branching*). Jika kondisi valid terpenuhi, alur dilanjutkan ke tahap berikutnya. Namun, apabila *parsing* gagal, sistem akan melakukan *retry* otomatis pada *node Creative Ad Copywriter* hingga *output* yang valid didapatkan. Mekanisme ini memberikan ketahanan sistem terhadap potensi kesalahan model bahasa, sekaligus menjamin bahwa setiap hasil teks memiliki konsistensi sebelum digunakan sebagai dasar pembuatan visual.

Setelah *output* valid berhasil didapatkan, alur berlanjut ke *node Generate an Image*, yang memanfaatkan hasil *visualIdea* dari LLM sebagai bahan utama untuk menghasilkan gambar iklan. *Node* ini menggunakan API *image generation* berbasis AI *(Gemini Image* atau *Imagen*) yang menerima *prompt* deskriptif berisi arahan visual. Misalnya gaya desain, palet warna, suasana emosional, hingga tata letak teks (*call-to-action*). Proses ini meniru tahap *art direction* pada produksi iklan konvensional, tetapi dilakukan sepenuhnya secara otomatis dan terintegrasi. *Output* dari tahap ini berupa *file* gambar

berformat JPEG/PNG dengan resolusi tinggi yang siap digunakan untuk publikasi. *Node* ini juga memiliki pengaturan *retry policy* agar proses tetap berjalan walaupun terjadi gangguan koneksi atau respons lambat dari API eksternal. Dengan demikian, tahap ini tidak hanya menghasilkan visual yang konsisten dengan identitas merek, tetapi juga memastikan efisiensi proses kreatif yang sebelumnya memerlukan kolaborasi manual antara desainer dan *copywriter*.

Langkah berikutnya adalah mengunggah *output* dari *node Generate an Image* ke repositori *cloud* seperti Google Drive untuk menyimpan hasil visual. Setiap *file* yang dihasilkan diberi nama berdasarkan tanggal visual tersebut dibuat, agar mudah dilacak dan diintegrasikan pada tahap *approval* maupun *publishing*. Node ini juga menghasilkan URL publik sementara yang dapat disisipkan dalam pesan *preview* untuk pengguna melihat hasil desain secara langsung melalui Telegram sebelum memutuskan untuk menyetujui atau merevisi konten tersebut. Dengan adanya sistem penyimpanan ini, seluruh data kampanye tersimpan dengan rapi dan dapat digunakan kembali untuk keperluan analisis maupun dokumentasi performa. Proses penyimpanan ini juga memastikan hasil *project* tetap terkelola secara sistematis, meskipun tidak terhubung ke lingkungan produksi perusahaan.



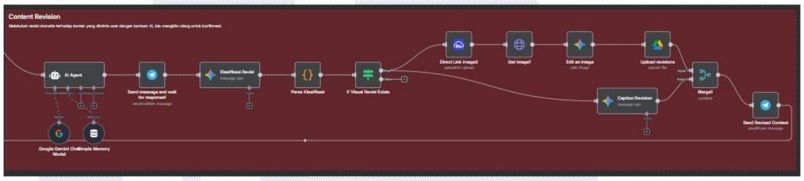
Gambar 3.17 Modul *Approval*

Bagian *User Approval* menjadi salah satu tahap krusial dalam sistem *Automation Ads Generation*, karena berfungsi sebagai mekanisme validasi hasil konten sebelum dipublikasikan atau direvisi. Tahapan ini dirancang agar pengguna tetap memiliki kendali penuh terhadap hasil akhir kampanye yang dihasilkan oleh sistem otomatis. Setelah proses *content generation* selesai dan menghasilkan aset berupa teks (*headline, caption, visual idea*) serta visual (poster iklan), sistem secara otomatis mengirimkan hasil tersebut kepada pengguna melalui *node Send a Photo Message*. *Node* ini memanfaatkan integrasi API Telegram untuk mengirimkan pesan berisi gambar iklan yang sudah diunggah sebelumnya beserta keterangan singkat dari hasil AI, seperti nama kampanye dan platform target. Tampilan visual ini dibuat agar menyerupai *preview* konten yang akan ditayangkan di media sosial sehingga pengguna dapat dengan mudah menilai kualitas hasil dan kesesuaian dengan *brief* yang diberikan. Pendekatan ini memberikan pengalaman interaktif dan efisien bagi pengguna karena tidak perlu mengakses sistem tambahan untuk melakukan peninjauan.

Setelah *preview* konten dikirim, sistem melanjutkan ke *node Send Approval*, yang berfungsi untuk mengirimkan pesan tindak lanjut berupa opsi persetujuan (*approval*) atau menolak (*decline*). *Node* ini menggunakan metode *sendAndWait* pada Telegram, yang memungkinkan sistem menunggu tanggapan langsung dari pengguna dalam sesi percakapan yang sama. Pengguna cukup membalas dengan menekan *button* yang tersedia, dan sistem akan mengekstrak respons tersebut untuk ditindaklanjuti. Hal ini menciptakan pola kerja menyerupai *human-in-loop automation*, di mana keputusan akhir tetap berada di tangan manusia namun seluruh komunikasi dan alur persetujuan berlangsung secara otomatis. Keuntungan lain dari metode ini adalah kemampuan menjaga konteks percakapan. Setiap sesi *approval* diidentifikasi berdasarkan *chat ID* dan *session key*, sehingga interaksi antar pengguna tidak saling tumpang tindih meskipun terdapat banyak kampanye yang berjalan bersamaan. Desain seperti ini juga meningkatkan pengalaman pengguna karena sistem terasa responsif dan mudah diajak berinteraksi tanpa memerlukan antarmuka web tambahan.

Proses validasi hasil kemudian dikendalikan oleh *node If Approval*, yang berfungsi sebagai percabangan logika untuk menentukan jalur eksekusi berikutnya berdasarkan tanggapan pengguna. Jika pengguna memberikan respons positif seperti "*Approve*", maka sistem akan meneruskan alur ke bagian *Publishing Flow* untuk mengunggah konten ke platform sosial media atau mencatat data kampanye ke dalam Google Sheets. Sebaliknya, apabila pengguna memilih untuk melakukan revisi, sistem akan mengarahkan alur ke percabangan lain untuk memicu *Revision Loop*, di mana *agent AI* akan memperbarui konten berdasarkan *feedback* yang diterima. Logika percabangan ini tidak hanya mempermudah pengambilan keputusan, tetapi juga menjadi dasar fleksibilitas sistem dalam menangani berbagai kemungkinan

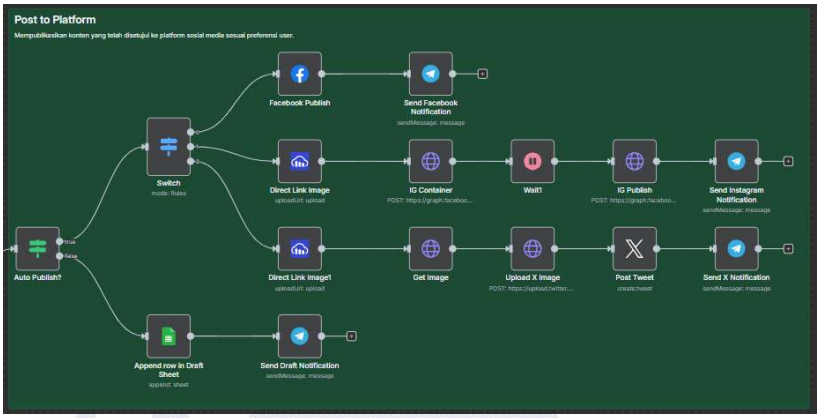
interaksi pengguna tanpa intervensi manual. Di sisi lain, seluruh keputusan dan waktu dicatat secara otomatis dalam Google Sheets sebagai bagian dari sistem internal yang memastikan seluruh aktivitas dapat ditelusuri. Dengan demikian, tahap *User Approval* tidak hanya berperan sebagai proses konfirmasi, tetapi juga sebagai titik kontrol penting yang menjamin bahwa hasil kampanye tetap relevan, akurat, dan sesuai dengan tujuan komunikasi.



Gambar 3.18 Modul Revisi

Tahap *Content Revision* dimulai setelah pengguna memilih opsi "*decline*" pada tahap *user approval*. Mekanisme ini dirancang untuk menerima masukan pengguna secara natural dan mengubah masukan tersebut menjadi instruksi yang terstruktur untuk proses perbaikan otomatis. Secara teknis, pertama-tama catatan *feedback* ditangkap oleh *node Send message and wait* yang menunggu respons teks dari pengguna untuk dikirimkan ke *AI Revision Parser* yang berfungsi mengekstrak poin revisi secara semantik. Misalnya, perubahan *tone*, penekanan USP, penggantian CTA, atau perbaikan elemen visual. Hasil ekstraksi dipetakan menjadi objek revisi yang terstruktur sehingga langkah *downstream* dapat menanganinya secara deterministik. Jika parser mendeteksi ambiguitas atau instruksi yang saling bertentangan, sistem akan memicu dialog klarifikasi otomatis untuk meminta detail tambahan kepada pengguna sehingga revisi dapat dilakukan dengan presisi. Pendekatan ini menggabungkan kekuatan LLM untuk memahami bahasa natural dengan kontrol berstruktur agar hasil revisi konsisten dan dapat diproses otomatis.

Setelah instruksi revisi terstruktur tersedia, alur melanjutkan ke modul *Creative Rewriter* untuk revisi teks dan *Image Editor* untuk pembaruan aset gambar bila diperlukan. Untuk revisi teks *node Creative Rewriter* menerima konteks gabungan, seperti *brief awal*, *brand context*, hasil *content generate* sebelumnya, serta objek revisi. *Prompt* dirancang sedemikian rupa agar model menghasilkan variasi yang mempertahankan identitas merek namun memenuhi permintaan revisi. Untuk elemen visual, *Visual Re-Generator* menerima *visual idea* yang telah dimodifikasi berdasarkan instruksi. Misalnya ubah palet warna, tambahkan ruang untuk CTA, atau ganti objek visual), lalu memanggil API *image generation* untuk membuat versi baru. *Output* tekstual dan visual selanjutnya digabungkan oleh *node Merge Revised Content* menjadi satu paket terintegrasi. *Node* ini juga menerapkan validasi akhir dengan memastikan ukuran *file*, format, dan panjang teks sesuai batasan platform sebelum mengirimkan *preview* revisi kembali ke pengguna. Siklus revisi dibuat untuk menjadi interaktif namun terkontrol sehingga pengguna dapat melakukan beberapa iterasi tanpa kehilangan konteks atau menimbulkan duplikasi kerja. Sistem menggunakan *session key* dan *shared context memory* agar semua revisi terhubung ke kampanye yang sama, sekaligus menyimpan riwayat versi sehingga tim pengembang dapat menganalisis perubahan dan performa model dari waktu ke waktu.



Gambar 3.19 Modul Publikasi

Tahap *Post to Platform* menjadi komponen akhir dalam sistem *Automation Ads Generation*, di mana konten iklan yang telah disetujui pengguna diproses menjadi konten yang siap tayang di platform media sosial. Tahap ini dirancang sangat modular agar dapat menangani berbagai skenario publikasi, baik secara otomatis maupun sebagai draf untuk diproses manual. *Workflow* diawali dengan evaluasi terhadap pilihan pengguna melalui *node "Auto Publish?"*. Jika pengguna memilih untuk tidak mempublikasikan otomatis, sistem akan langsung mengalihkan alur ke *node Append Row in Draft Sheet*, yang menyimpan hasil konten ke *sheet* draf lengkap dengan tanggal, *brand, headline, caption*, dan tautan visual yang sudah disimpan ke dalam Google Drive. Namun apabila pengguna memilih untuk langsung mempublikasikannya, sistem mengaktifkan jalur publikasi penuh melalui *node Switch* yang berfungsi menyeleksi platform berdasarkan *input*. *Switch* ini menjadi pusat logika kontrol yang memungkinkan *workflow* bercabang tanpa harus membangun alur terpisah untuk Facebook, Instagram, maupun X.

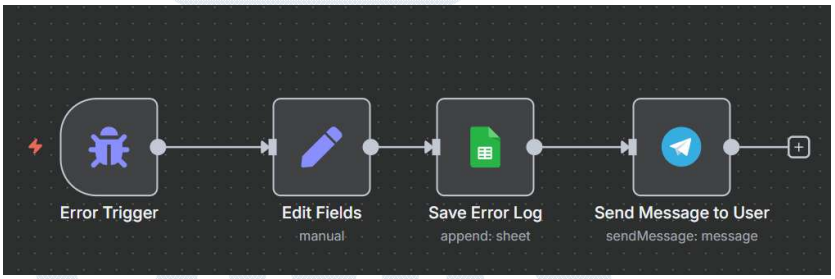
Pada alur publikasi Facebook, *workflow* memanfaatkan *node Facebook Publish* yang berhubungan langsung dengan Graph API. Platform ini memerlukan tautan gambar sebagai *input*, karena

upload media tidak dilakukan melalui *file* langsung tetapi melalui tautan yang telah diunggah sebelumnya ke *cloud storage*. *Caption* yang dihasilkan oleh *node Creative Copywriter* juga dimuat sebagai parameter pesan, sehingga unggahan iklan dapat muncul di halaman Facebook pengguna. Kemudian sistem mengirimkan konfirmasi otomatis kembali ke pengguna melalui *node Send Facebook Notification* untuk memberitahukan bahwa iklan telah berhasil tayang. Notifikasi ini bukan hanya berfungsi sebagai *user feedback*, tetapi juga sebagai bentuk *monitoring* internal *pipeline* agar developer dapat memastikan bahwa API Facebook berhasil merespons tanpa *error*. Selain publikasi Facebook, Instagram menjadi salah satu alur yang paling kompleks karena API Instagram Business memiliki arsitektur yang berbeda dari Facebook, meskipun menggunakan Graph API. *Workflow* harus mengikuti proses dua tahap, pertama membuat *media container* melalui *node IG Container* kemudian menunggu proses *backend* API selesai *memproses container* tersebut. Pada tahap ini, *node Wait* digunakan untuk memberi jeda beberapa detik agar *container* dapat diproses sebelum dilanjutkan ke tahap publikasi. Setelah itu, *node IG Publish* memanggil *endpoint* "*media_publish*" untuk mengirimkan konten tersebut ke *feed* Instagram. Penggunaan dua *node* terpisah menjadi penting karena jika proses *container* tidak selesai atau melewati batas waktu, API akan mengembalikan *error*. Setelah publikasi berhasil, node *Send Instagram Notification* mengirimkan notifikasi kepada pengguna melalui Telegram.

Sementara itu, jalur publikasi X (Twitter) memerlukan serangkaian langkah berbeda karena API platform tersebut mengharuskan media diunggah menggunakan *endpoint media/upload*. *Workflow* menangani ini melalui *node Direct Link Image1* dan *Get Image*, yang bertujuan mengunduh dan mempersiapkan data gambar dari *cloud* sebelum mengirimkannya ke

server X. Setelah media diunggah, X mengembalikan "*media_id*" yang kemudian dikirimkan ke *node Post Tweet*. Proses ini mencerminkan *pipeline* standar Twitter API generasi terbaru dan menunjukkan integrasi penuh antara *workflow aumation* dan REST API. Namun, dalam implementasi proyek ini terdapat keterbatasan signifikan karena akun yang digunakan masih berada pada *free tier*, yang tidak mengizinkan mengunggah gambar bersamaan dengan *tweet*. Akibatnya, *workflow* yang sudah mendukung integrasi penuh tidak dapat mengeksekusi publikasi *tweet* dengan gambar dan hanya dapat mengirim *tweet* berbasis teks. Meskipun demikian, desain *pipeline* tetap dipertahankan agar kompatibel dengan *tier* API yang lebih tinggi, sehingga ketika akses API premium tersedia, sistem dapat langsung menjalankan publikasi gambar tanpa perlu perubahan arsitektur besar.

3.3.1.11 Agent Autonomy Enhancement



Gambar 3.20 *Error handler workflow* baru

Sistem *Automation Ads Generation* tidak hanya dirancang untuk menghasilkan dan mempublikasikan konten secara otomatis, tetapi juga dilengkapi dengan mekanisme *Error Handling Workflow* yang berfungsi untuk mendeteksi, mencatat, dan melaporkan kesalahan secara *real-time*. *Workflow* ini dibangun menggunakan modul *Error Trigger* di n8n yang akan aktif setiap kali terjadi kegagalan eksekusi pada *workflow* utama. Ketika *error* terjadi, *Error Trigger* secara otomatis menangkap data penting seperti nama *workflow*, ID eksekusi, *node* yang gagal dieksekusi, serta pesan

error yang diberikan oleh sistem. Data tersebut kemudian diproses oleh *node Edit Fields* yang bertugas mengekstrak informasi dan menyusunnya ke dalam struktur data yang valid sehingga dapat digunakan oleh *node* berikutnya. Pendekatan ini memastikan bahwa setiap *error* terdokumentasi secara konsisten dan dapat ditelusuri dengan mudah, serta mendukung proses *debugging*, dan peningkatan kualitas sistem di masa mendatang.

Setelah informasi *error* berhasil disusun, *workflow* meneruskannya ke *node Save Error Log*, yang akan menyimpan catatan kesalahan tersebut ke dalam Google Sheets yang berfungsi sebagai *database*. Catatan ini mencakup kolom seperti "*ExecutionID*", "*WorkflowName*", "*Node*", dan "*Description*". Penyimpanan eror di Google Sheets memungkinkan pelacakan dan analisis jangka panjang. Misalnya, mendeteksi pola *error* berulang pada *node* tertentu atau identifikasi potensi *bottleneck* pada alur publikasi ke platform tertentu. Dengan demikian, fitur ini tidak hanya membantu memperbaiki masalah secara langsung tetapi juga berfungsi sebagai sumber *insight* untuk optimasi sistem di fase pengembangan selanjutnya. Selain itu, pendekatan ini juga memudahkan integrasi lebih lanjut apabila perusahaan ingin menghubungkan log tersebut ke *dashboard monitoring* seperti Looker Studio atau Grafana untuk visualisasi performa *workflow*. Terakhir, *workflow* ini akan mengirimkan notifikasi ke developer atau admin melalui Telegram. Notifikasi ini berisi informasi lengkap mengenai *workflow* yang gagal, *node* penyebab *error*, dan deskripsi kesalahannya, sehingga tim yang bertanggung jawab dapat segera melakukan tindakan. Mekanisme ini penting untuk mengurangi waktu respons terhadap insiden dan memastikan sistem berjalan dengan baik, terutama ketika banyak *workflow* bekerja secara bersamaan. Selain itu, desain notifikasi berbasis Telegram juga sangat praktis karena memungkinkan admin menerima laporan *error*

kapan saja tanpa harus membuka *dashboard* secara manual. Dengan begitu, sistem tidak hanya mampu mengotomatisasi proses pembuatan dan publikasi konten, tetapi juga mampu mempertahankan stabilitas operasional melalui pemantauan otomatis yang terintegrasi.

Gambar 3.21 *Dashboard monitoring*

Integrasi antara *workflow* otomatisasi konten di n8n dengan *dashboard monitoring* dirancang untuk memberikan visibilitas penuh terhadap kondisi operasional sistem secara *real-time*. Sistem *monitoring* ini memanfaatkan metrik yang diekspor melalui *Prometheus endpoint* milik n8n, yang kemudian diekstrak dan divisualisasikan dalam berbagai panel Grafana. Dengan pendekatan ini, setiap komponen internal seperti pemakaian CPU, memori *heap*, *event loop lag*, jumlah *workflow* aktif, hingga status kesehatan *instance* dapat dipantau secara kontinu. *Dashboard* yang digunakan dalam proyek ini bukan hanya menampilkan visualisasi dasar, tetapi juga memberikan lapisan analitik mendalam yang memudahkan identifikasi performa abnormal atau *bottleneck* yang mulai muncul. Tampilan seperti "*System Health Overview*", "*Memory & Resource Usage*", dan "*CPU Usage Breakdown*" memberikan gambaran menyeluruh tentang *runtime* n8n. Selain itu, banyak panel dilengkapi *threshold-based coloring* untuk memberikan indikasi visual yang

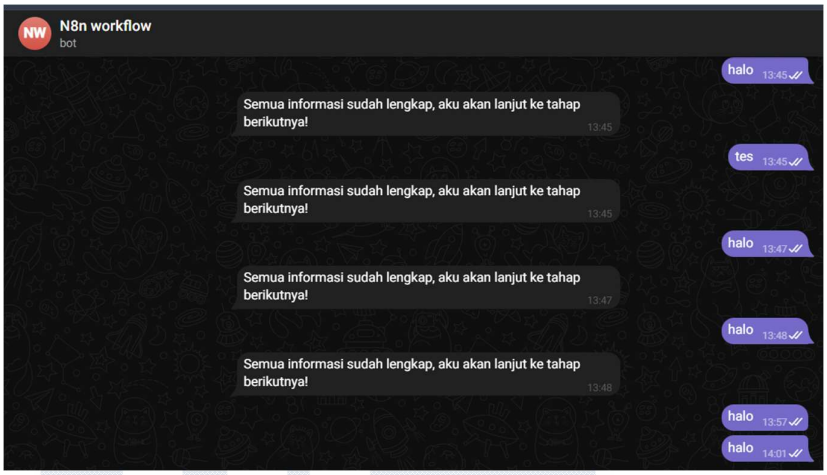
cepat ketika suatu metrik berada di luar batas aman. *Dashboard monitoring* ini juga mencakup panel-panel yang secara spesifik menyoroti aspek stabilitas Node.js yang menjadi fondasi utama *runtime* n8n. Melalui panel seperti *Heap Space Usage Breakdown*, administrator dapat melihat penggunaan memori pada segmen *heap* seperti *old space, new space, code space*, dan *large object space*. Informasi ini krusial untuk mendeteksi potensi *memory leak* yang sering terjadi pada alur otomatisasi berskala besar, terutama ketika banyak operasi asinkron dilakukan secara paralel.

Panel *Event Loop Lag* (*All Percentiles*) juga memberikan informasi penting terkait seberapa cepat *environment* Node.js memproses antrean *event*. Jika nilai *lag* meningkat, menandakan bahwa *workflow* sedang berada di kondisi yang berat atau ada *node* yang menyebabkan proses tersendat. Grafana memungkinkan tampilan dalam bentuk persentase, sehingga pengguna dapat memahami respons sistem di berbagai kondisi beban. Visualisasi histogram *Garbace Collection* (GC) juga membantu mengidentifikasi frekuensi dan durasi operasi GC yang sering menjadi penyebab lonjakan signifikan pada latensi sistem. Selain *monitoring* performa server, *dashboard* ini juga menampilkan informasi terkait status kesehatan server virtual secara langsung. Panel "*n8n Instance Status*" menggunakan metrik *up* dari Prometheus untuk menampilkan apakah server dalam keadaan *UP* atau *DOWN*, lengkap dengan indikator warna yang sangat jelas. Hal ini penting terutama untuk *workflow* otomatis yang berjalan sepanjang waktu tanpa intervensi manual. Selain itu, panel "*Process Uptime*" memastikan bahwa sistem tidak mengalami *restart* tidak terduga yang bisa mengindikasikan *crash* atau instalasi *plugin* yang bermasalah. Informasi seperti *Active Workflows* dan *Instance Role* juga membantu memvalidasi apakah server sedang bekerja sesuai

ekspektasi, terutama jika arsitektur menjalankan fitur skala besar dengan *worker instance*.

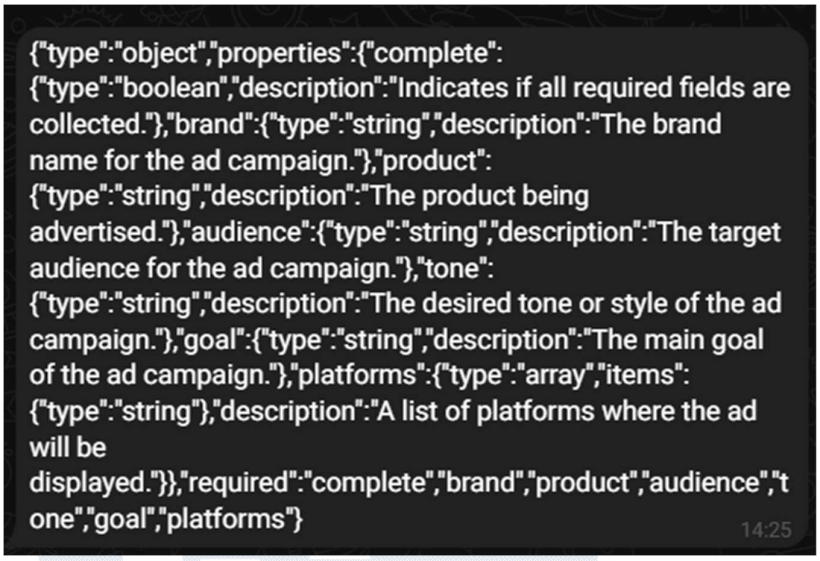
Dashboard monitoring ini juga memiliki bagian khusus yang menampilkan data versi n8n dan Node.js dalam bentuk tabel yang telah ditransformasikan agar mudah dibaca. Panel seperti "*n8n Version Info*" dan "*Node.js Version Info*" secara otomatis menampilkan nomor versi, untuk membantu administrator memastikan bahwa sistem menggunakan versi stabil dan aman. Mengetahui versi perangkat lunak sangat penting ketika terjadi masalah kompabilitas, misalnya ketika *node* tertentu tidak berjalan dengan benar akibat perubahan API internal pada pembaruan n8n. Fitur ini juga mendukung proses audit dan dokumentasi teknis dalam jangka panjang, terutama pada proyek yang terus berkembang. Salah satu aspek yang paling penting dalam *dashboard* ini adalah kemampuannya mendeteksi anomali secara dini sebelum dampaknya dirasakan pengguna. Misalnya, jika *event loop log* meningkat secara tiba-tiba, Grafana dapat menunjukkan tren tersebut dalam grafik yang mudah dianalisis. Ketika memori *heap* mulai mendekati ambang batas tertentu, panel *gauge* akan berubah warna dari hijau menjadi kuning hingga merah menjadikannya peringatan visual yang kuat bahwa sistem membutuhkan perhatian. Demikian juga jika jumlah *GC major* meningkat, menandakan sistem telah kehabisan ruang dan mulai melakukan pembersihan besar yang akan berdampak pada performa. Mekanisme *threshold* pada Grafana sangat membantu karena mempermudah tim teknis memahami tingkat keseriusan setiap kondisi tanpa harus melihat log mentah.

3.3.1.12 Testing & Refinement



Gambar 3.22 Potongan percakapan dengan *agent*

Pada tahap awal pengujian, tantangan muncul ketika sistem mulai diuji dengan *input* aktual dari Telegram tetapi *workflow* belum mampu menafsirkan atau menangani percakapan secara konsisten. Dari riwayat percakapan, terlihat bahwa setiap kali pengguna mencoba berinteraksi melalui Telegram, agen sering memberikan jawaban yang tidak sesuai konteks atau langsung melompat ke kesimpulan "data lengkap" meskipun pengguna belum mengisi seluruh data yang diperlukan. Masalah ini terjadi karena struktur *prompt* awal belum memaksa model untuk mendeteksi kelengkapan data dengan benar. Selain itu, *node parsing* juga masih belum stabil sehingga sering mengalami eror akibat model mengembalikan *output* selain JSON. Masalah lainnya adalah, sebagian besar ekspresi n8n masih mengandalkan variabel dari *node* yang belum dijalankan, sehingga menyebabkan pesan eror seperti "*An expression references this node, but the node is unexecuted*". Tahap ini menunjukkan bahwa fondasi awal *workflow* belum siap berjalan secara *end-to-end*.



Gambar 3.23 Contoh eror yang terjadi

Selama pengujian awal, salah satu sumber eror yang paling signifikan adalah ketidakmampuan agen untuk menghasilkan format JSON yang konsisten ketika diminta memberikan *summary input* atau *output* akhir. Dari riwayat percakapan, terlihat beberapa kasus di mana model memberikan jawaban dalam bentuk paragraf biasa, *bullet list*, atau terkadang JSON yang tidak valid sehingga *node Gemini Parse* gagal memprosesnya. Ketika JSON tidak valid, seluruh *workflow* terhenti karena *node* setelahnya tidak menemukan *field* yang dibutuhkan seperti *brand*, *product*, atau platform. Selain itu, ada momen ketika agen menjawab pertanyaan tetapi menyisipkan informasi tambahan yang tidak diminta, sehingga menyebabkan *parsing* menjadi kacau dan *If node* selalu masuk ke *False Branch*. Kondisi ini membuat agen menjadi bingung dan tidak mampu mengikuti alur percakapan terstruktur sebagaimana yang diharapkan dari *chatbot* interaktif. Permasalahan parsing ini juga menyebabkan eror berantai, misalnya *node* berikutnya mencoba membaca "*$json.complete*", tetapi variabel tersebut tidak pernah dihasilkan. Masalah lainnya juga terjadi pada *routing* dan *logical branching* yang menyebabkan *workflow* tidak bekerja sebagaimana

mestinya. Banyak percobaan menunjukkan bahwa meskipun sudah ada *node If* untuk memeriksa kelengkapan data, agen tetap mengarah ke *false branch* akibat kesalahan evaluasi. Sebagai contoh, ketika pengguna menjawab "ya" atau "data lengkap", sistem tetap menganggap nilainya *false* karena parser tidak berhasil membuat *flag boolean* yang benar. Selain itu, Telegram *Trigger* sesekali tidak mengirim *payload* lengkap sehingga *node Extract User Data* tidak menerima *chat_id*. ada juga masalah ketika *node* tertentu tidak dieksekusi tetapi nilainya tetap direferensikan oleh ekspresi dalam *node* lain, sehingga memunculkan eror "*node unexecuted reference*". Kesalahan hubungan antar *node* juga menjadi penyebab utama *workflow* gagal berjalan, khususnya pada *flow* yang memiliki struktur percabangan kompleks.

Masalah lainnya terdapat pada *agent* yang mengalami kesulitan dalam memahami percakapan bertahap, terutama ketika pengguna tidak menjawab dengan format yang sesuai atau memberikan respons yang ambigu. Di riwayat percakapan terlihat beberapa momen ketika *agent* seharusnya bertanya ulang, namun justru berpindah ke topik lain atau menyimpulkan bahwa *brief* sudah lengkap meskipun banyak *field* yang belum diisi. Hal ini dipicu oleh belum optimalnya konfigurasi *window memory* dan belum adanya mekanisme *strict validation* pada setiap pertanyaan. Selain itu, *session chat_id* pada Telegram juga belum terikat dengan benar ke *memory store* sehingga *agent* sering kehilangan konteks setelah satu atau dua pertanyaan. Pada beberapa percobaan, *agent* mengulang pertanyaan yang sudah dijawab, dan ini membuat *user flow* terasa tidak stabil serta tidak siap untuk pengguna nyata. Masalah seperti ini sangat terlihat ketika *agent* melakukan pengecekan data, namun pesan "tolong lengkapi data Anda" muncul padahal pengguna sebelumnya sudah memberikan informasi lengkap. Kondisi tersebut

menjadi indikasi kuat bahwa *workflow* belum memiliki mekanisme pembatas dan verifikasi yang memadai.

Untuk meningkatkan performa dari *workflow*, perlu dilakukan beberapa perbaikan. Tahap perbaikan dimulai dengan memperkuat struktur *prompt agent* agar model menghasilkan *output* yang jauh lebih stabil dan dapat diprediksi. Pada fase awal testing, terdapat masalah karena *output* LLM tidak dalam format JSON yang valid. Untuk mengatasi masalah tersebut, *prompt agent* direvisi menjadi lebih eksplisit dengan mencantumkan contoh format, aturan ketat, dan instruksi "*always respond in valid JSON*". Tidak hanya itu, mekanisme *fallback* juga ditambahkan melalui *node If* yang memaksa *workflow* mengulang instruksi jika format tidak sesuai. *Node Gemini Parse* kemudian diperbaiki dengan validasi otomatis, jika *parsing* gagal, *workflow* tidak lagi berhenti melainkan meminta klarifikasi dari pengguna sehingga percakapan tetap berlanjut. Dengan revisi tersebut, model dapat menghasilkan struktur data yang lebih konsisten seperti "brand", "product", "goal", "tone", dan sebagainya. Hal ini membuat *node* berikutnya dapat beroperasi tanpa eror referensi yang sebelumnya sering muncul.

Setelah memastikan *output* JSON konsisten, perbaikan berikutnya dilakukan pada sisi logika percabangan terutama di *node if* yang sebelumnya selalu menghasilkan nilai *false*. Masalah ini diatasi dengan mengubah ekspresi pengecekan menjadi lebih fleksibel, seperti memeriksa keberadaan nilai, tipe data *boolean*, dan *fallback* lain jika variabel belum terbentuk. Selain itu, *node if* yang sebelumnya tidak sesuai aliran kerja diperbaiki agar selalu dijalankan setelah *node* parse selesai. Dengan penyusunan ulang jalur *logic*, *agent* tidak lagi melompat ke *branch* yang salah, dan alur dapat berjalan mulus tanpa interupsi.

Perbaikan berikutnya berfokus pada penyempurnaan mekanisme *session handling*, yang sebelumnya menjadi penyebab *agent* kehilangan konteks atau mengulang pertanyaan yang sudah dijawab. *Window Buffer Memory* diperbaiki dengan menetapkan *session key* berbasis "chat_id", sehingga setiap percakapan Telegram dipetakan ke memori yang benar. Selain itu, jumlah pesan disimpan dalam *context window* diperbesar untuk memastikan *agent* tetap ingat percakapan sebelumnya, terutama jika pengguna berhenti di tengah percakapan atau menjawab dengan format acak. Perbaikan ini memastikan alur percakapan bertahap tetap konsisten dari awal hingga akhir. Bersamaan dengan itu, alur *fallback* juga ditambahkan. Jika pengguna memberikan jawaban yang ambigu atau tidak sesuai pertanyaan, *agent* secara otomatis meminta klarifikasi dan tidak langsung memprosesnya sebagai valid *input*. Setelah fondasi percakapan stabil, perbaikan selanjutnya dilakukan pada tahap *mapping* dan konsolidasi data yang menjadi *input* bagi proses *content generation*. Pada versi awal, beberapa *field* seperti *platforms*, *audience*, atau *tone* sering hilang akibat *parsing* yang gagal. Untuk mengatasinya, node *Parse Response* diperbarui dengan *metadata cleaning* yang memaksa semua *field* wajib muncul setidaknya sebagai *string* kosong atau *array* kosong. Setelah itu, *node Format Complete Message* dibuat ulang untuk mengubah seluruh data menjadi JSON standar yang siap digunakan oleh modul kreatif. *Field* yang sebelumnya harus diperiksa manual kini diisi otomatis menggunakan *fallback default*, misalnya "*tone*": *userTone* || "*default friendly marketing tone*". *Node* ini juga menjadi *checkpoint* terakhir untuk memastikan bahwa sebelum masuk ke tahap *content generate*, data sudah bersih, lengkap, dan terstruktur. Selain itu, seluruh ekspresi yang mengacu pada *node* yang belum dieksekusi dihapus atau dipindahkan agar tidak menimbulkan eror referensi lagi.

Setelah berbagai perbaikan penting diterapkan ke dalam *workflow Ads Automation*, proses pengujian kembali dilakukan untuk memastikan seluruh komponen dapat bekerja secara konsisten tanpa eror. Pada tahap ini, penulis kembali menguji alur percakapan mulai dari *trigger* Telegram hingga seluruh *input* terkumpul dengan benar. Hasil pengujian menunjukkan bahwa *agent* kini mampu mempertahankan konteks percakapan dan tidak kehilangan *state* seperti yang terjadi pada tahap awal. Setiap respons pengguna diproses secara konsisten dan diubah menjadi JSON yang valid tanpa ada *field* kosong yang menyebabkan eror pada *node* berikutnya. Histori percakapan juga menunjukkan bahwa alur pertanyaan menjadi jauh lebih stabil dan tidak melompat-lompat seperti sebelumnya. Selain itu, proses validasi format JSON di *node parse Response* tidak lagi menimbulkan eror karena perbaikan pada struktur *prompt* dan pola ekstraksi.

Pengujian kemudian diteruskan pada bagian *content generation*, terutama memastikan bahwa *prompt* yang dikirimkan ke *agent* benar-benar memuat *brand context*, *creative brief*, dan seluruh parameter *input* pengguna. Pada tahap ini terlihat bahwa *wokflow* berhasil mengirimkan *prompt* lengkap tanpa mengalami ketidakcocokan antara variabel *input* dengan data yang diambil dari Google Sheets. *Caption*, *headline*, dan *visual idea* yang dihasilkan oleh model kini menjadi lebih relevan karena isi *prompt* sudah terstruktur dengan benar dan tidak ada lagi bagian yang hilang. *Agent* juga mampu menyelesaikan respons tanpa eror *parsing* sehingga JSON dapat digunakan langsung oleh *node* berikutnya. Dalam riwayat percakapan terlihat beberapa percobaan dilakukan untuk menguji variasi platform dan *tone*, dan semuanya berhasil menghasilkan *output* yang sesuai harapan. Hal ini menunjukkan bahwa *dependency* antar *node* sudah terhubung dengan baik setelah diperbaiki. Tidak ada lagi kasus di mana *agent* menghasilkan format

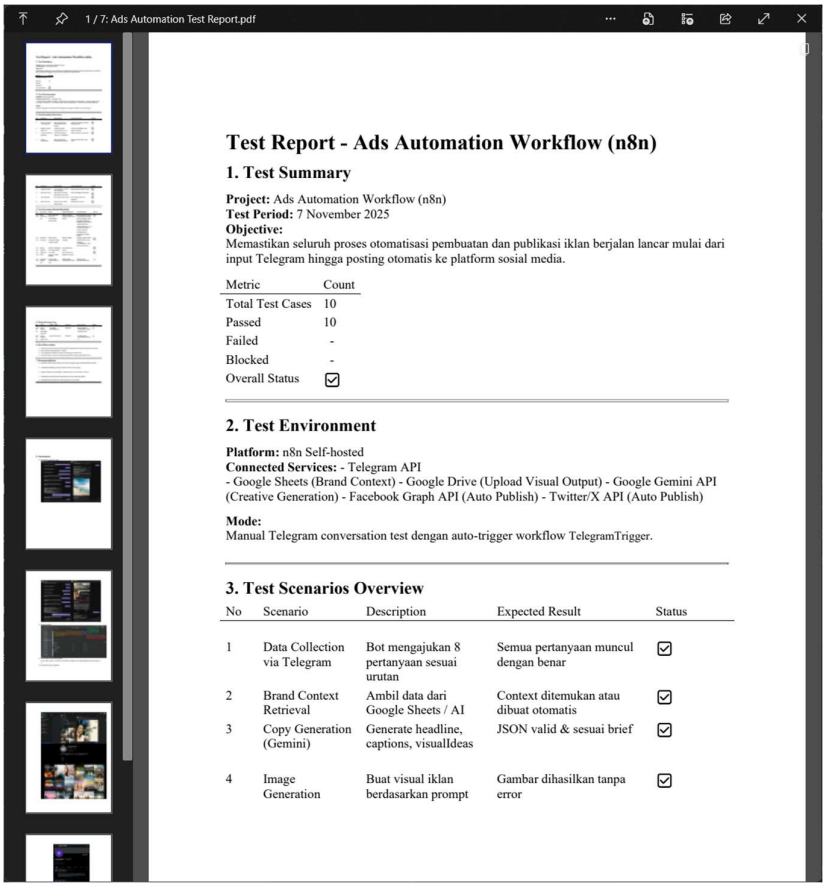
bebas yang tidak bisa diuraikan, yang sebelumnya menjadi salah satu isu terbesar.

Tahap pengujian selanjutnya berfokus pada alur *approval*, khususnya memastikan bahwa sistem mampu mengirimkan hasil *creative output* ke pengguna serta menerima persetujuan atau revisi. Dalam pengujian, pengguna menguji berbagai kondisi seperti memberikan persetujuan langsung, meminta revisi, dan mencoba memberikan masukan tambahan. *Node Wait for Response* kini dapat bekerja stabil, menangkap *input* pengguna secara *real-time* tanpa kehilangan sesi ataupun memicu *branch* yang salah. Sistem berhasil mengirimkan paket hasil yang berisi *caption* dan *preview image* kembali ke Telegram dengan struktur data yang rapi. Ketika pengguna menekan *button approve* untuk *approval*, alur publikasi langsung dijalankan tanpa eror *logic* seperti pada tahap awal. Sementara itu, jika pengguna meminta revisi, alur revisi bekerja dengan baik dan *agent* mampu menghasilkan konten baru berdasarkan *feedback* pengguna.

Bagian selanjutnya dari pengujian mencakup publikasi konten ke platform sosial media yang sebelumnya merupakan salah satu bagian paling rentan eror. Setelah perbaikan, proses publikasi untuk Facebook dan Instagram melalui Meta API dapat berjalan mulus karena *file* gambar yang dikirimkan sudah memuat URL yang benar. Begitu juga pada alur publikasi ke X (Twitter), meskipun API *free tier* membatasi penggunaan media *upload*, sistem tetap dapat mengirimkan *tweet* berbasis teks secara konsisten. Riwayat percakapan Telegram juga menunjukkan beberapa percobaan yang berhasil memicu jalur publikasi tanpa memunculkan eror "*media_id missing*" seperti yang terjadi pada iterasi awal. *Node* pengambilan gambar dari *cloud* kini berfungsi stabil karena format URL dari *image generation* sudah diperbaiki. Selain itu, *output* notifikasi yang

dikirim ke Telegram setelah *posting* sukses memberikan indikasi bahwa seluruh rangkaian publikasi berjalan sesuai rencana.

Pengujian terakhir berhubungan dengan observasi, *error handling*, dan dependensi antar *node* untuk memastikan bahwa *workflow* dapat berjalan tanpa perlu intervensi manual. Setelah perbaikan, setiap eror yang terjadi pada salah satu node langsung diarahkan ke *error handling workflow*. *Chat history* memperlihatkan bahwa penulis melakukan beberapa *stess test* seperti memberikan *input* yang tidak lengkap, mengakhiri percakapan di tengah jalan, atau mengirim pesan yang ambigu. Semua kondisi ini berhasil ditangani dengan *fallback message* yang informatif dan tidak merusak alur keseluruhan. Selain itu, *logging* ke sistem *monitoring* juga telah terhubung sehingga setiap *event* tercatat. Pengiriman *metadata* ke *dashboard monitoring* berhasil diverifikasi melalui uji coba berulang selama sesi percakapan. Tahap pengujian ini memastikan bahwa *workflow* tidak hanya berfungsi saat kondisi ideal, tetapi juga memiliki ketahanan terhadap berbagai skenario eror. Dengan demikian setelah seluruh perbaikan diterapkan, *workflow Ads Automation* dapat dikatakan siap memasuki tahap finalisasi.



Gambar 3.24 Dokumen laporan pengujian proyek

Setelah melalui serangkaian pengujian, terakhir penulis membuat *Test Report* yang disusun untuk memberikan gambaran menyeluruh mengenai keberhasilan *Ads Automation workflow*. Laporan ini memiliki tujuan utama untuk memvalidasi bahwa seluruh fungsi inti mulai dari *input* pengguna, pemrosesan konteks, pembuatan konten, hingga publikasi otomatis dapat berjalan stabil tanpa mengalami kegagalan signifikan. Lingkup pengujian mencakup pengujian fungsional, pengujian alur percakapan, integrasi API dengan layanan eksternal, serta pengujian *output* pada setiap tahap *workflow*. Selain itu, *test report* ini menyajikan sepuluh *test case* utama yang mewakili seluruh komponen kritis dalam alur sistem, termasuk *input collection*, *parsing* JSON, *content*

generation, *approval flow*, dan publikasi otomatis. Pengujian dilakukan secara manual dengan menggunakan Telegram sebagai media interaksi utama untuk memastikan seluruh percakapan berjalan sesuai rencana desain UX. Setiap *test case* dicatat dengan atribut *expected result*, *actual result*, dan status akhir untuk memudahkan evaluasi objektif.

Test Report mencatat bahwa seluruh *test case* yang dijalankan mendapatkan status "*passed*", menandakan tidak ada satu pun skenario yang mengalami kegagalan mayor. Salah satu keberhasilan penting adalah stabilitas alur pengumpulan data melalui Telegram, di mana bot berhasil menyampaikan seluruh pertanyaan secara berurutan dan menyimpan jawaban pengguna dengan benar. Selain itu, proses *parsing* menggunakan *Gemini Parse* juga berhasil diuji berulang kali tanpa menghasilkan eror format yang sebelumnya menjadi salah satu hambatan terbesar pada tahap awal pengembangan. *Workflow* juga berhasil memproses *brand context* dari Google Sheets dan menggabungkannya dengan *user brief* secara konsisten dalam format JSON yang lengkap. Pada tahap *content generation, caption, headline, dan visual description* dapat diproduksi oleh model AI tanpa *output* selain JSON atau format yang tidak bisa diproses. Pengujian juga mencatat bahwa *file* visual yang dihasilkan telah berhasil diunggah ke Google Drive tanpa itu *path* atau *permission*.

Test report juga memberikan fokus khusus pada tahap *approval* yang menjadi jembatan antara proses *generate content* dan fase publikasi. Dalam pengujian, sistem berhasil mengirimkan *caption* dan gambar *preview* ke Telegram dengan struktur pesan yang rapi, sehingga pengguna dapat menilai hasil secara visual sebelum melanjutkan. *Node SendAndWait* diuji berulang kali untuk memastikan sistem dapat membaca respons pengguna, baik berupa

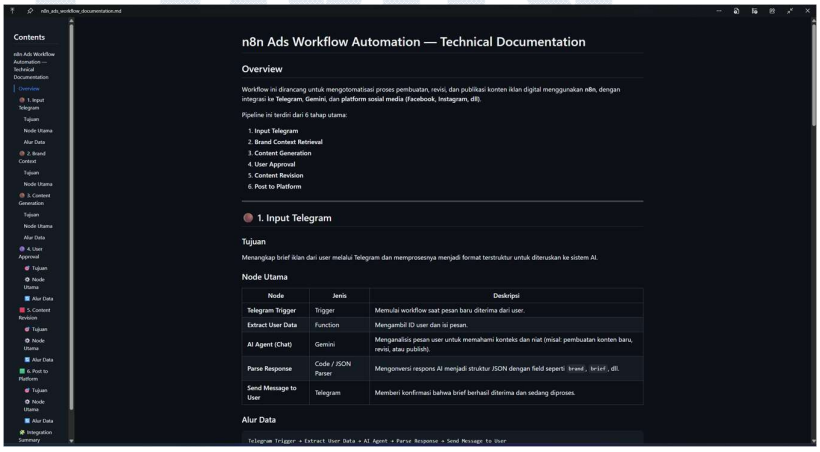
"*Approve*" maupun "*Revise*". Hasil pengujian menunjukkan bahwa jalur persetujuan dapat memicu proses publikasi secara langsung, sedangkan jalur revisi mampu mengembalikan alur ke modul AI untuk memperbaiki konten sesuai *feedback*. Selain itu, *test report* menegaskan bahwa sesi percakapan tidak terputus meskipun pengguna memberikan respons setelah jeda tertentu karena memori sistem dan *session binding* telah berfungsi dengan baik. Tahap ini juga menginformasi bahwa *edge case* seperti respons ambigu dapat ditangani melalui *fallback logic*. Dengan demikian, modul *approval* dan *revision* telah tervalidasi sebagai komponen penting yang memastikan *workflow* bersifat *user friendly*.

Tahap publikasi konten dinyatakan berjalan dengan baik berdasarkan laporan pengujian meskipun terdapat catatan khusus pada platform tertentu. Untuk Facebook dan Instagram, integrasi dengan Meta Graph API diuji dan konten dapat dipublikasikan ke akun *dummy* tanpa eror ataupun kesalahan validasi. Pengujian juga memastikan bahwa URL *file* yang dihasilkan dari Google Drive dapat dibaca dengan benar oleh *node* publikasi karena struktur *prompt* visual telah diperbaiki. Sementara itu, pengujian terhadap platform X (Twitter) menunjukkan bahwa *posting* teks berhasil dilakukan, tetapi publikasi gambar tidak dapat diuji sepenuhnya karena keterbatasan *free tier* API yang tidak mendukung media *upload* secara penuh. *Test Report* mencatat hal ini sebagai *known limitation* dan bukan sebagai *defect*, karena batasan tersebut berasal dari platform bukan dari *workflow*. Selain itu, *node* notifikasi yang mengirim pesan konfirmasi *posting* ke Telegram diuji dan berhasil memunculkan pesan sukses pada seluruh platform yang diuji.

Bagian akhir *Test Report* memberikan rangkuman menyeluruh mengenai performa *workflow* setelah diuji dalam beberapa tahap. Laporan mencatat bahwa tidak ada *test case* yang

menghasilkan status "*failed*", sehingga *workflow* dapat dikatakan stabil dan layak untuk digunakan dalam tahap demonstrasi atau pengembangan lebih lanjut., meskipun demikian, *test report* tetap mencatat beberapa rekomendasi peningkatan seperti migrasi *inline button* Telegram ke *callback_data* agar interaksi tidak membuat tab baru, serta optimasi kecepatan *parsing* ketika model menghasilkan *output* kompleks. selain itu, laporan juga menyarankan peningkatan struktur folder Google Drive agar manajemen *file* menjadi lebih rapi terutama jika *workflow* diperluas untuk *multi-brand* atau *multi-user*. Dokumentasi memperlihatkan bahwa setiap isu minor telah diidentifikasi dan siap diperbaiki dalam iterasi berikutnya. Kesimpulannya, *test report* menyatakan bahwa *workflow Ads Automation* telah mencapai tingkat performa yang memadai untuk dijadikan *prototype* fungsional. Hasil *test report* ini memperkuat bukti bahwa seluruh perbaikan dan iterasi sebelumnya memberikan dampak langsung pada stabilitas sistem. Dengan demikian, sistem dinyatakan berhasil lolos seluruh pengujian fungsional penting dan siap digunakan dalam skenario uji dan pengembangan yang lebih luas.

3.3.1.13 Documentation



Gambar 3.25 Dokumentasi proyek

Dokumentasi *workflow Ads Automation* disusun untuk memberikan pemahaman menyeluruh mengenai arsitektur, alur proses, integrasi, serta mekanisme operasional sistem yang dibangun menggunakan n8n. Setiap bagian dalam dokumentasi dirancang untuk menjelaskan tujuan fungsional, komponen inti, serta keterkaitan antar *node* di dalam *workflow*. Tujuan utama dari dokumentasi ini adalah memastikan bahwa *workflow* tidak hanya dapat digunakan tetapi juga dapat dipahami, diperbaiki, dan dikembangkan oleh siapa pun yang mengelola proyek ini di masa mendatang. Dokumen ini juga akan menjadi referensi utama ketika dilakukan *debugging*, testing, ataupun penambahan fitur baru. Semua langkah mulai dari *input user* via Telegram, pemrosesan oleh AI, revisi konten, hingga publikasi ke platform media sosial media dijelaskan secara terstruktur dan sistematis. Setiap bagian *workflow* diuraikan dengan detail, termasuk logika *decision making* dan format data yang dibutuhkan. Hal ini membuat dokumentasi memiliki nilai penting dalam menjaga konsistensi *pipeline*.

Bagian dokumentasi terkait *input* Telegram menjelaskan bagaimana sistem menangkap dan memproses pesan dari *user* sebagai titik awal seluruh *automation*. Dalam tahap ini, dokumentasi menegaskan peran *Telegram Trigger* sebagai komponen yang mendeteksi pesan baru dan memicu *workflow* secara otomatis. *Node* seperti *Extract User Data* dan *AI Agent* dijelaskan sebagai elemen pengolah pesan untuk memastikan maksud pengguna dipahami dengan benar, misalnya apakah pengguna ingin membuat konten baru, meminta revisi, dan sebagainya. Dokumentasi juga menuliskan struktur JSON yang dibangun oleh parser, sehingga developer lain dapat memahami format internal yang digunakan sebagai *standart input*. Informasi mengenai bagaimana AI mengubah pesan natural menjadi *brief* yang dapat dieksekusi juga disampaikan secara jelas. Selain itu, dokumentasi memuat detail bagaimana sistem

mengirimkan *acknowledgment* melalui Telegram untuk memastikan pengguna mengetahui bahwa permintaan sedang diproses. Bagian ini menjadi penting karena merupakan fondasi komunikasi antara pengguna dengan sistem.

Pada bagian dokumentasi yang membahas *Brand Context*, dijelaskan bahwa *workflow* melakukan pengambilan data spesifik *brand* dari Google Sheets yang digunakan sebagai *database*. Setiap elemen *brand* diuraikan dan dijelaskan bagaimana *node-node* tertentu mengubah data ini menjadi *prompt* yang siap dikirim ke AI. Dokumentasi selanjutnya masuk ke tahap *Content Generation*, menjelaskan bagaimana data kontekstual tersebut dipadukan dengan *brief* pengguna untuk menghasilkan konten kreatif. Detail mengenai *node* seperti *Creative AI Generator*, *Content Parser*, hingga *Visual Generator* ditulis secara komprehensif, termasuk format *input output* masing-masing *node*. Penjelasan ini memastikan alur konten dapat dipahami oleh developer lain tanpa bergantung pada pengetahuan eksternal. Dokumentasi juga menjelaskan bahwa semua hasil sementara disimpan secara terstruktur di Google Drive agar mudah diakses *node* berikutnya.

Bagian selanjutnya menjelaskan proses penting di mana pengguna diberikan kontrol penuh terhadap konten sebelum dipublikasikan. Tahap *User Approval* dibuat dalam dokumentasi sebagai proses pengiriman konten ke Telegram dan menunggu balasan berupa persetujuan atau permintaan revisi. *Node AI Approval Handler* dijelaskan sebagai komponen yang memahami bahasa natural pengguna dan mengonversinya menjadi kondisi *boolean* yang menentukan arah alur. Jika pengguna memilih revisi, dokumentasi menjelaskan bagaimana sistem masuk ke tahap *Content Revision* yang melibatkan AI Parser, *Creative Rewriter*, dan *Visual Regenerator*. Setiap langkah revisi ditulis dengan detail,

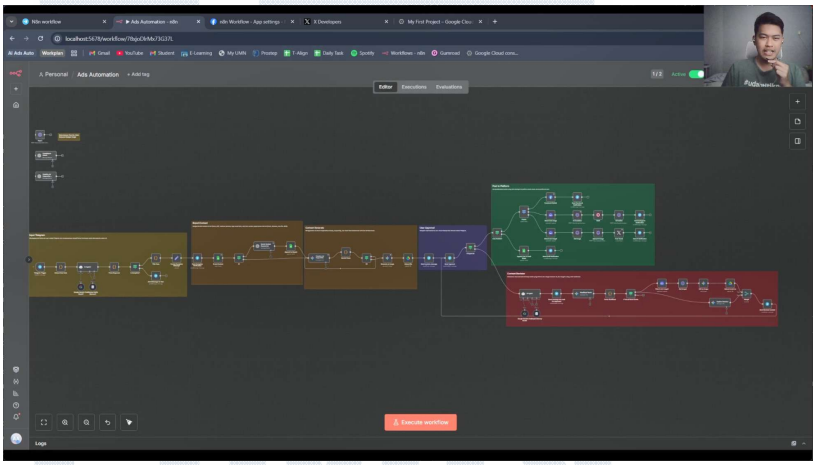
mulai dari membaca *feedback*, mengidentifikasi poin penting, hingga merevisi elemen konten tertentu berdasarkan *feedback* pengguna. Dokumentasi juga menekankan bagaimana sistem menjaga format konsisten meskipun terjadi beberapa kali revisi berulang. Setelah konten direvisi, sistem mengirimkan ulang hasilnya untuk disetujui kembali oleh pengguna, sehingga tercipta siklus umpan balik yang jelas dan terstruktur.

Pada bagian *Post to Platform*, dokumentasi memberikan gambaran lengkap mengenai bagaimana konten yang telah disetujui dipublikasikan ke berbagai platform seperti Facebook, Instagram, dan X (Twitter). Penjelasan dimulai dari *node Switch* yang bertugas mendeteksi platform target berdasarkan *input user*. Dokumentasi menjelaskan setiap perbedaan tenis antar platform, terutama cara Meta Graph API menangani *upload* gambar dan *caption* melalui *endpoint container-based posting*. Terdapat juga penjelasan rinci mengenai proses *upload* media ke X, termasuk keterbatasan *free tier* yang mengharuskan *upload* media dilakukan sebagai langkah terpisah saat membuat *tweet*. Dokumentasi menggambarkan bagaimana *node Get Image* dan *Upload X Image* saling berhubungan untuk menyesuaikan gambar ke format yang kompatibel dengan API X. Penambahan *node* notifikasi seperti *Send Facebook Notification* ditulis sebagai bagian dari proses transparansi agar pengguna mengetahui setiap keberhasilan publikasi.

Secara keseluruhan dokumentasi *workflow Ads Automation* ini secara keseluruhan memberikan gambaran menyeluruh mengenai arsitektur, alur eksekusi, dan integrasi lintas sistem yang membentuk proses otomatisasi pembuatan hingga publikasi konten kreatif, proses *approval* dan revisi, hingga publikasi ke berbagai platform sosial media dijelaskan secara sistematis sehingga memudahkan proses pemahaman, pengujian, serta pengembangan lanjutan.

Dokumentasi juga menyoroti mekanisme *error handling*, *logging*, serta integrasi eksternal, sehingga memberikan kejelasan mengenai bagaimana sistem menjaga kestabilannya. Selain itu, struktur dokumentasi dibuat modular dan ter standarisasi untuk memastikan bahwa setiap komponen *workflow* memiliki acuan operasional yang jelas apabila diperlukan *debugging* atau modifikasi. Dengan cakupan penjelasan yang komprehensif dan berorientasi, dokumentasi ini berfungsi sebagai landasan penting untuk mempertahankan stabilitas *workflow* sekaligus mendukung pengembangan fitur di masa mendatang.

3.3.1.14 Final Presentation



Gambar 3.26 Video presentasi

Pada tahap presentasi akhir, penulis membuat sebuah video demonstrasi yang berfungsi sebagai rangkuman menyeluruh dari keseluruhan proses pengembangan sistem *Ads Automation*, mulai dari desain arsitektur hingga operasional teknis di dalam *workflow* n8n. Presentasi ini diawali dengan pemaparan mengenai gambaran umum proyek tujuan pengembangan, dan manfaat dari sistem otomatisasi pembuatan serta publikasi konten iklan berbasis AI. Penulis menampilkan struktur utama *workflow* secara utuh, sehingga dapat dipahami bagaimana setiap fase terhubung menjadi satu alur

otomatis yang melekat satu sama lain. Bagian awal presentasi ini memberikan konteks kuat mengenai motivasi proyek dan bagaimana teknologi AI serta *automation workflow* membantu efisiensi proses *marketing* digital.

Pada bagian inti presentasi, penulis menjelaskan fungsi masing-masing *node* di dalam *workflow* secara rinci dan sistematis. Presentasi mencakup penguraian setiap modul seperti *Telegram Trigger* yang memulai alur percakapan, *AI Chat Agent* yang mengekstraksi *brief*, dan lain sebagainya. Selain itu, penulis menampilkan cara kerja *image generation*, mekanisme *approval*, dan *revision loop*, di mana masing-masing *node* diperlihatkan secara jelas dilengkapi dengan penjelasan mengenai setiap *input* dan *output* data. Bagian ini juga membahas kredensial yang digunakan untuk menghubungkan berbagai layanan eksternal sambil menekankan bahwa seluruh kredensial tersebut dikonfigurasi melalui *environment* yang aman. Bagian penutup dari presentasi difokuskan pada demonstrasi secara langsung yang memperlihatkan bagaimana *workflow* bekerja secara nyata dari awal hingga akhir. Penulis memperagakan proses interaksi dengan Telegram, mulai dari pengguna mengirimkan *brief* iklan, *agent* menanyakan pertanyaan, hingga sistem menghasilkan konten dan gambar secara otomatis. Setelah itu, ditunjukkan juga bagaimana *workflow* menangani *approval user*, melakukan revisi ketika diminta, dan mempublikasikan konten ke platform sosial media sesuai preferensi. Dengan menampilkan seluruh proses secara langsung, presentasi memberikan bukti konkret mengenai fungsionalitas dan stabilitas sistem yang dibangun, sekaligus menunjukkan bagaimana komponen AI dan API eksternal saling terintegrasi dalam satu ekosistem operasional yang utuh.

3.3.2 Kendala yang Ditemukan

Selama menjalani kegiatan praktik kerja dan mengembangkan proyek *Automation Ads generation*, penulis menghadapi sejumlah tantangan yang muncul dari sisi teknis maupun dari sisi proses pembelajaran. Kendala-kendala ini merupakan bagian natural dari proses pengembangan sistem berbasis AI dan *automation workflow*, terutama mengingat proyek ini merupakan pengalaman pertama penulis dalam membangun integrasi *multi-platform* dan *multi-agent* secara *end-to-end*. Meskipun demikian, setiap hambatan yang ditemui justru memberikan kesempatan bagi penulis untuk lebih memahami cara kerja API, tata kelola *resource* AI, serta arsitektur otomatisasi berbasis n8n. Dalam konteks ini, seluruh kesulitan dipandang sebagai bagian dari proses pembelajaran yang konstruktif dan bukan sebagai kekurangan dari pihak mana pun, melainkan kondisi yang wajar terjadi dalam lingkungan teknologi yang berkembang cepat dan penuh dinamika.

1. Keterbatasan Kuota API pada model AI (Terutama Gemini)

   Kendala paling signifikan yang ditemui penulis berkaitan dengan keterbatasan kuota API pada berbagai model AI yang digunakan, khususnya model Gemini untuk kebutuhan *parsing, content generation*, dan *image generation*. Karena *workflow* yang dibangun mengandalkan pemanggilan LLM berulang kali, batas kuota yang cepat habis sering menyebabkan interupsi pada proses testing maupun eksekusi *workflow*. Beberapa fungsi tidak dapat dijalankan secara terus-menerus sehingga penulis perlu melakukan penjadwalan ulan atau menghemat frekuensi pemanggilan API. Hal ini membuat proses iterasi dan *debugging* menjadi lebih lambat dibandingkan skenario ideal dengan kuota tak terbatas. Selain itu sistem perlu dilengkapi *fallback mechanism* agar *workflow* tetap berjalan ketika API limit tercapai, yang menambah kompleksitas pengembangan. Kendala ini akhirnya memberikan pengetahuan penting bagi penulis mengenai manajemen kuota serta pentingnya

optimasi jumlah pemanggilan LLM dalam sebuah sistem *automation*.

2. Keterbatasan Akses terhadap n8n *Cloud* yang Bersifat Berbayar

   Selain kendala API, penulis juga menghadapi pembatasan yang berkaitan dengan akses platform n8n *Cloud*, yang fitur-fitur lanjutannya hanya tersedia dalam paket berbayar. Beberapa kemampuan seperti *execution history* yang panjang, kapasitas penyimpanan, atau integrasi *node* tertentu tidak tersedia di versi gratis. Keterbatasan ini bukan bersumber dari pihak perusahaan, melainkan merupakan batasan umum layanan *SaaS* yang wajar terjadi pada *free tier*. Dengan kondisi ini, penulis memperoleh wawasan baru mengenai bagaimana menyeimbangkan kebutuhan fitur dengan kemampuan platform, sekaligus memahami pentingnya pemilihan lingkungan *deployment* yang sesuai dengan skala proyek.

3. Kurangnya Pengalaman Awal dalam *Automation Workflow* dan *Agentic* AI

   Karena proyek ini merupakan pengalaman pertama penulis dalam membangun sistem *automation* dan *agentic* AI secara *full-stack*, proses awa pembelajaran memerlukan usaha ekstra. Penulis membutuhkan waktu untuk memahami konsep dasar seperti *state management* dalam *conversational flow*, *callback logic*, serta integrasi *multi* platform melalui API. Selain itu, memahami dependensi antar *node* di n8n menjadi tantangan tersendiri yang memperlambat pekerjaan pada tahap awal. Proses iterasi juga tidak selalu berjalan mulus karena *output* LLM terkadang tidak sesuai format, yang mengharuskan perbaikan *prompt engineering* dan pembuatan mekanisme validasi tambahan. Namun, melalui proses *trial and error* tersebut penulis menjadi lebih terampil dalam merancang alur kerja otomatis yang stabil dan modular.

3.3.3 Solusi atas Kendala yang Ditemukan

Dalam menghadapi berbagai kendala selama pelaksanaan praktik kerja dan proses pengembangan proyek *Automation Ads Generation*, penulis berusaha mencari pendekatan penyelesaian yang efektif tanpa mengurangi kualitas hasil kerja. Setiap hambatan dianalisis dan ditangani dengan strategi yang disesuaikan dengan kapasitas teknis dan lingkungan kerja yang tersedia, sehingga proses pengembangan tetap berjalan meskipun dihadapkan pada keterbatasan tertentu. Penyusunan solusi ini bukan hanya bertujuan menyelesaikan masalah jangka pendek, tetapi juga memberikan kerangka berpikir yang dapat diterapkan untuk proyek-proyek serupa di masa mendatang. Melalui proses ini, penulis memperoleh pengalaman signifikan dalam menangani isu teknis secara sistematis, meningkatkan kemampuan *troubleshooting*, serta belajar merancang *workflow* yang lebih efisien.

1. Optimasi dan Pemanfaatan Kuota API melalui *Free Trial* Google Cloud Console

   Untuk mengatasi keterbatasan kuota API pada model AI seperti Gemini, penulis menerapkan strategi kombinasi antara optimasi teknis dan pemanfaatan *free triel* dari Google Cloud Console. Penulis terlebih dahulu mendaftarkan akun Google Cloud menggunakan bank digital yang mendukung metode verifikasi otomatis, sehingga penulis mendapatkan kredit atau kuota gratis senilai kurang lebih lima juta rupiah yang berlaku selama sekitar empat bulan. Kredit ini memungkinkan proses *development* dilakukan tanpa hambatan biaya selama periode pengembangan utama, terutama pada tahap yang membutuhkan banyak pemanggilan LLM untuk keperluan Testing, *parsing*, dan iterasi *prompt*. Di sisi teknis, penulis tetap melakukan optimasi pemanggilan API, seperti menggabungkan beberapa *prompt* menjadi satu proses terpadu, menghindari pemanggilan ulang yang tidak perlu, serta menggunakan sistem validasi ketat sebelum *node* AI

dijalankan. Penulis juga mengaktifkan mekanisme *caching* pada data tertentu agar tidak terus memanggil API untuk informasi statis. Kombinasi antara efisiensi teknis dan dukungan *free trial* ini membuat pengembangan proyek tetap dapat berjalan optimal tanpa biaya tambahan selama masa praktik kerja.

2. Pemanfaatan Lingkungan Lokal dan *Workaround* untuk Keterbatasan n8n *Cloud*.

   Untuk menyiasati keterbatasan fitur dan *workflow* pada *free trial* n8n *Cloud*, penulis mengalihkan seluruh proses *development* ke versi lokal n8n menggunakan Docker. Dengan menjalankan server n8n secara lokal, penulis dapat menggunakan fitur lebih leluasa, mengakses *execution history* tanpa batas, serta menguji berbagai integrasi API tanpa dibatasi *tier* berbayar. Pendekatan ini membuat *workflow* tetap dapat berkembang secara penuh meskipun tanpa infrastruktur berbayar. Ketika perlu melakukan *deployment* atau demonstrasi, penulis dapat langsung memindahkan *workflow* kembali ke n8n *Cloud* dengan melakukan penyesuaian ringan pada *node*, kredensial, dan *environment*. Dengan demikian, proses pengembangan dapat berlangsung optimal tanpa memerlukan biaya tambahan.

3. Pembelajaran Bertahap dan Iterasi Sistematis untuk Menguasai *Automation Workflow* serta *Agentic AI*

   Untuk mengatasi kurangnya pengalaman awal dalam *automation workflow* dan *agentic* AI, penulis menerapkan pendekatan belajar bertahap dengan fokus pada pemahaman konsep fundamental sebelum memasuki integrasi kompleks. penulis memulai dengan mempelajari dasar-dasar *node* di n8n, cara kerja *webhook*, serta pola komunikasi antar *node* sebelum beralih ke modul yang lebih rumit seperti *wait*, *switch*, *parsing*, dan *approval*

logic. Selain itu, penulis melakukan banyak percobaan kecil sehingga setiap bagian *workflow* dapat dipahami secara jelas sebelum digabungkan menjadi alur *end-to-end*. Untuk memahami *agentic AI*, penulis mempelajari teknik *prompt engineering*, struktur JSON LLM, dan cara mengatur sesi memori sehingga *agent* dapat mempertahankan konteks dengan baik. Pengalaman *debugging* yang cukup intens juga membantu penulis memahami *error pattern* dan memperbaiki masalah secara lebih cepat dan terarah.