BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Kegiatan magang dilaksanakan pada perusahaan CV. Inovasi Artificial Intelligence Indonesia (AI.DECE) dengan jabatan *AI Engineer Intern* di bawah pengawasan *Technical Team Manager* dan supervisor. Fokus utama adalah perancangan Platform AI Reporting yang menyediakan empat fitur inti, yaitu *AI Conversational Analytics*, *Automated Data Visualization*, *Dynamic Excel Export*, dan *OCR Data Intake*. Platform ini dirancang agar dapat diintegrasikan ke dalam sistem internal perusahaan klien dengan konfigurasi fitur yang dapat disesuaikan berdasarkan kebutuhan masing-masing, serta menciptakan ekosistem digital yang memfasilitasi interaksi dinamis antara pengembang, pengguna, dan aplikasi pendukung [53]. Pendekatan ekosistem ini memberikan keleluasaan klien dalam memilih modul sesuai kebutuhan operasional [54]. Perancangan ini dilakukan kolaboratif dengan supervisor (*Founder*/CEO) dan pembimbingan teknis dari manajer *Technical Team*.

Pembimbingan dikoordinasikan oleh Bapak Ivan Handryks Sitanaya (*Founder* dan CEO) sebagai supervisor melalui pertemuan rutin setiap Rabu dan Jumat pukul 10.00 WIB hingga pukul 12.00 WIB, meliputi presentasi capaian, umpan balik, pemberian tugas, dan diskusi strategis.

Komunikasi dan koordinasi dengan rekan tim dilakukan melalui dua pendekatan, yaitu pertemuan tatap muka dan komunikasi virtual menggunakan platform seperti WhatsApp, Microsoft Teams, dan Google Meet. Periode magang memberikan pengalaman kolaborasi langsung dengan anggota tim profesional, pembelajaran mengenai alur kerja dalam industri teknologi, serta kesempatan berkontribusi pada inisiatif-inisiatif pengembangan yang sedang dijalankan perusahaan.

3.2 Tugas yang Dilakukan

Pengalaman magang yang dilakukan di perusahaan AI.DECE pada jabatan *AI Engineer Intern* meliputi kegiatan pembangunan sistem platform berbasis kecerdasan buatan yang merupakan produk andalan perusahaan, yakni Platform AI Reporting. Pembangunan platform ini difokuskan pada implementasi empat fungsi

kunci, yaitu *AI Conversational Analytics*, *Automated Data Visualization*, *Dynamic Excel Export*, dan *OCR Data Intake*, beserta perancangan tampilan antarmuka web. Implementasi sistem ini memanfaatkan berbagai teknologi yang diintegrasikan secara menyeluruh guna menghasilkan performa maksimal, meliputi Flask sebagai kerangka kerja, LangChain, *Large Language Model* (LLM) dari OpenAI dengan model `gpt-5-mini`, mekanisme autentikasi *JSON Web Token (JWT)*, layanan *Cloudinary*, pustaka visualisasi seperti `matplotlib` sebagai *Plot Agent*, serta teknologi *Optical Character Recognition* (OCR) [31], [32]. Implementasi ini bertujuan memperluas kapabilitas platform agar mampu berintegrasi dengan infrastruktur sistem yang dimiliki oleh mitra klien perusahaan.

Sepanjang periode magang, sejumlah aplikasi pendukung digunakan untuk memfasilitasi proses pembangunan sistem secara efisien dan sistematis. Visual Studio Code (VS Code) dipilih sebagai *Integrated Development Environment* (IDE) primer dalam aktivitas penulisan dan penyuntingan kode program. Sementara itu, GitHub difungsikan sebagai alat *version control system* untuk mengatur serta melacak setiap modifikasi yang terjadi pada kode proyek. Pengelolaan percabangan repositori mengikuti pola penamaan `origin/dev/jackson/` sebagai identifikasi untuk setiap iterasi pengembangan yang dilakukan.

Implementasi keempat fitur inti Platform AI Reporting melibatkan pemanfaatan basis data MySQL yang dihubungkan melalui SQLAlchemy dengan Alembic, memungkinkan komunikasi yang efektif antara Flask dan sistem basis data. Basis data ini menjadi tempat penyimpanan berbagai elemen informasi, termasuk catatan transaksi penjualan, berkas yang diunggah pengguna, serta data memori dan histori interaksi yang dihasilkan oleh fitur *chatbot*.

Keseluruhan alur pengembangan proyek dijalankan secara kolaboratif dengan menggunakan GitHub, di mana pekerjaan terbagi dalam satu repositori bernama `ai-reporting-plne` yang memiliki dua direktori utama, yaitu direktori *client* untuk pengembangan sisi antarmuka pengguna berdasarkan respons dari API Endpoint, dan direktori *server* untuk penanganan logika *backend*. Validasi fungsionalitas dilakukan menggunakan Postman, kemudian *output* pengujian ditransformasikan menjadi aplikasi web menggunakan *framework* Next.js yang dikombinasikan dengan Tailwind CSS untuk *styling*.

Di samping kontribusi pada Platform AI Reporting, aktivitas magang juga meliputi keterlibatan dalam berbagai inisiatif pengembangan lainnya seperti AI Invoice untuk klien Tokyo Consulting, pembangunan sistem AI OCR, penerapan solusi *open-source* RagFlow, serta tugas klasifikasi tarif paket ISP menggunakan

algoritma DBSCAN. Tujuan dari inisiatif-inisiatif tersebut adalah memperbarui desain antarmuka menjadi lebih kontemporer, adaptif, dan atraktif secara visual. Pemahaman komprehensif terhadap *codebase* yang telah ada sebelumnya menjadi faktor krusial untuk menjaga kestabilan sistem dan mereduksi risiko munculnya *bug* atau *error* dalam proses pengembangan.

3.3 Uraian Pelaksanaan Magang

Pelaksanaan kegiatan magang di perusahaan AI.DECE berlangsung selama lebih kurang enam bulan sesuai dengan ketentuan yang tercantum dalam kontrak magang. Durasi kegiatan magang tersebut terhitung sejak tanggal 4 Agustus 2025 hingga 6 Februari 2025. Perhitungan waktu pelaksanaan tidak mencakup hari Minggu serta hari libur nasional maupun internasional, kecuali dalam kondisi tertentu yang memerlukan penanganan mendesak. Tabel 3.1 menampilkan kronologi kegiatan yang dikerjakan secara berkala setiap pekan selama program magang berlangsung di perusahaan AI.DECE.

Tabel 3.1. Pekerjaan setiap minggu selama periode magang

Minggu Ke -	Pekerjaan yang Dilakukan
1	Mengikuti pengenalan lingkungan kerja dan briefing ruang lingkup pengembangan Platform *AI Reporting*; Mempelajari tumpukan teknologi utama (Flask, Next.js, Tailwind, MySQL, Cloudinary) dan konsep *chatbot* berbasis LLM; Mengkaji dokumentasi *agent plotting*, pembuatan file Excel dari *DataFrame*, serta melakukan eksplorasi awal konsep *voicebot* STT/TTS untuk mendukung pengembangan ke depan.
2	Mengumpulkan kebutuhan bisnis lebih rinci dari supervisor dan klien untuk setiap fitur utama platform; Menyusun *use case* pengguna dan admin serta merancang skema basis data awal melalui ERD; Menyusun kebutuhan dan kontrak API untuk *chatbot*, riwayat percakapan, proses LLM, serta merancang desain antarmuka awal halaman *chatbot* menggunakan Adobe Photoshop.
	Lanjut pada halaman berikutnya

Lanjutan Tabel 3.1

Minggu Ke -	Pekerjaan yang Dilakukan
3	Menyempurnakan rancangan antarmuka *chatbot*, *sidebar*, dan *welcome page* berdasarkan masukan klien; Melakukan diskusi desain agar tampilan selaras dengan kebutuhan bisnis; Merancang dan mengintegrasikan sistem basis data menggunakan migrasi Alembic serta mengembangkan modul `SQLChatHistoryManager` dengan model `User`, `Room`, dan `Message` untuk pengelolaan riwayat percakapan.
4	Merancang alur halaman *login* dan *register* untuk pengguna dan admin di sisi antarmuka; Mengimplementasikan *endpoint* autentikasi pada `main.py` yang terhubung dengan `SQLChatHistoryManager` untuk pembuatan pengguna baru dan penyimpanan password ter-*hash*; Mengembangkan *endpoint* manajemen *room* dan pesan sehingga daftar *room* dan riwayat percakapan dapat diakses secara aman dan terstruktur.
5	Menambahkan *endpoint* pendukung pembuatan akun dengan validasi khusus agar nama pengguna admin tidak disalahgunakan; Merancang dan mengimplementasikan klasifikasi `next_agent` dan `data_action` untuk membedakan permintaan teks, visualisasi, dan ekspor data; Menghubungkan `VisualizerBot` dengan aplikasi Flask dan membangun endpoint *post_prompt* yang mengelola alur pengiriman pertanyaan ke LLM dan penyimpanan balasan di basis data.
6	Memvalidasi perilaku klasifikasi `next_agent` dan `data_action` untuk skenario fitur *AI Conversational Analytics* dengan respons teks; Mengimplementasikan *endpoint* `/api/analytics/weekly_activity` yang menghitung aktivitas pesan harian per pengguna; Menginisialisasi proyek *frontend* Next.js beserta konfigurasi Tailwind dan *routing* dasar serta menerapkan JWT pada alur *login* dan *register* di sisi klien.
	Lanjut pada halaman berikutnya

Lanjutan Tabel 3.1

Minggu Ke -	Pekerjaan yang Dilakukan
7	Menyusun kerangka komponen *chatbot* di Next.js dengan pengelolaan *state rooms*, *messages*, dan `selectedRoomId`; Mengembangkan komponen `Sidebar` dan `ChatInterface` untuk menampilkan daftar *room*, mengirim pesan, serta menampilkan indikator bot sedang mengetik; Mengintegrasikan komponen `Welcome` dan proteksi rute sehingga halaman *chatbot* hanya dapat diakses oleh pengguna yang telah terautentikasi.
8	Mengimplementasikan halaman *dashboard* dengan kartu statistik yang menampilkan total *room*, aktivitas 24 jam terakhir, dan *room* terbaru; Menambahkan visualisasi aktivitas mingguan sederhana dengan memanfaatkan data `WeeklyActivity` dan menampilkan kondisi khusus ketika pengguna belum memiliki percakapan; Mengoptimalkan pemanggilan API, menambah penanganan token kedaluwarsa, dan melakukan pengujian secara *end-to-end*.
9	Mengeksplorasi rancangan *voicebot* berbasis STT menggunakan model Whisper sebagai pengembangan lanjutan; Menyesuaikan jalur klasifikasi *next_agent* dan *data_action* agar permintaan analitik dengan nilai FETCH dan PLOT dialirkan ke modul *plotter* yang menghasilkan kode visualisasi Python; Menyempurnakan alur pada fitur *Automated Data Visualization* dengan memastikan `image_base64` valid, mengunggah grafik ke Cloudinary, serta melakukan pengujian hingga grafik tampil di antarmuka.
10	Merancang lebih rinci skema data hasil kueri yang akan diekspor ke Excel sebagai bagian dari fitur *Dynamic Excel Export*; Mengkaji dan mengonfigurasi pustaka `xlsxwriter` untuk penulisan berkas Excel di *backend* serta mengarahkan permintaan dengan `next_agent = EXPORT` ke jalur pembuatan Excel; Memastikan berkas Excel dapat dihasilkan, diunggah ke Cloudinary, dan diunduh kembali oleh pengguna melalui antarmuka dengan pengalaman yang konsisten.
Lanjut pada halaman berikutnya	

Lanjutan Tabel 3.1

Minggu Ke -	Pekerjaan yang Dilakukan
11	Melakukan pengujian menyeluruh terhadap alur pada fitur *Dynamic Excel Export* mulai dari permintaan pengguna hingga berkas Excel berhasil diunduh; Menyelesaikan *endpoint* admin untuk transaksi penjualan sehingga operasi CRUD penuh terhadap data hasil ekstraksi OCR dapat dilakukan; Menyiapkan struktur dasar halaman `AdminPage` di Next.js dengan tabel transaksi dan formulir tambah data manual serta menambahkan fitur edit dan hapus data; Mendiskusikan perkembangan platform bersama klien dan supervisor termasuk usulan fitur WhatsApp Chatbot dan analisis awal atas prioritas pengembangannya.
12	Mempelajari dokumentasi WhatsApp Chatbot namun memutuskan memprioritaskan penyelesaian *pipeline* pada fitur *OCR Data Intake*; Menambahkan fungsi `extract_text_and_meta_from_pdf` dan `extract_transaction_from_text` untuk mengubah PDF menjadi teks dan JSON transaksi menggunakan PyPDF2 dan ChatOpenAI; Mengimplementasikan *endpoint* `/api/admin/transactions/upload` dan `/api/admin/ocr-history` sehingga proses unggah PDF, ekstraksi OCR dan LLM, penyimpanan transaksi, dan penelusuran riwayat dapat dikelola melalui antarmuka admin.
13	Mengintegrasikan fitur unggah PDF pada `AdminPage` dengan state `ocrProgress` agar admin dapat memantau tahapan *uploading*, *processing*, dan *success* secara *real-time*; Menampilkan riwayat OCR di sisi klien melalui `adminService.listOcrHistory` dengan informasi nama file, ukuran, status, dan tautan ke transaksi terkait; Menambahkan opsi pemilihan `id_transaksi` target saat unggah sehingga data lama dapat diperbarui serta melakukan pengujian terhadap seluruh alur OCR dan modul admin.
	Lanjut pada halaman berikutnya

Lanjutan Tabel 3.1

Minggu Ke -	Pekerjaan yang Dilakukan
14	Memulai proyek AI Invoice untuk klien Tokyo Consulting dengan menganalisis contoh faktur dan mengidentifikasi *field* penting yang perlu diekstrak; Merancang arsitektur layanan AI Invoice berbasis Flask dan Next.js beserta skema basis datanya; Membangun prototipe *backend* dan halaman *frontend* untuk pengunggahan, peninjauan, dan koreksi hasil ekstraksi; Melakukan pengujian komprehensif dengan variasi format *invoice* dan menyusun dokumentasi teknis agar sistem mudah dioperasikan kembali oleh tim klien.
15	Mengalihkan fokus ke perancangan layanan AI OCR generik yang dapat digunakan untuk berbagai jenis dokumen bisnis; Menyusun prioritas tipe dokumen dan merancang *pipeline* yang dapat dikonfigurasi ulang tanpa mengubah kode inti; Mengimplementasikan layanan AI OCR generik di Flask dan menghubungkannya dengan antarmuka Next.js untuk uji unggah dan perbandingan hasil ekstraksi; Melakukan *profiling* performa *pipeline*, mengoptimalkan bagian yang lambat, serta menyiapkan skrip *deployment* Docker Compose.
16	Mengeksplorasi dan mengonfigurasi RagFlow sebagai solusi *open source Retrieval Augmented Generation*; Merancang skenario penggunaan RagFlow untuk menghubungkan dokumen finansial klien dan menyediakan antarmuka tanya jawab berbasis dokumen; Membangun adaptor antara RagFlow dan aplikasi Flask serta halaman Next.js sederhana untuk pencarian jawaban; Melakukan uji coba internal dengan supervisor dan menyusun dokumentasi integrasi RagFlow termasuk arsitektur sistem dan alur *ingestion* data.
	Lanjut pada halaman berikutnya

Lanjutan Tabel 3.1

Minggu Ke -	Pekerjaan yang Dilakukan
17	Merancang eksperimen klasifikasi harga paket ISP menggunakan algoritma DBSCAN dengan mengumpulkan sampel paket dan menentukan fitur utama; Mengimplementasikan skrip Python berbasis Flask yang menjalankan DBSCAN dan mengekspos hasil klaster sebagai API serta membangun halaman analisis di Next.js untuk visualisasi hasil; Melakukan penyetelan hiperparameter DBSCAN, mengevaluasi kualitas pengelompokan, merapikan repositori kode beserta berkas `README.md`, dan mulai kembali meninjau fitur Platform AI Reporting untuk disusun daftar perbaikan prioritasnya.
18	Menyesuaikan arsitektur sistem basis data dengan memisahkan beban kerja ke dua basis data terpisah sebagai langkah awal *horizontal scaling*/*database partitioning*; Memverifikasi bahwa alur Platform AI Reporting telah mencakup seluruh fitur utama, yaitu *AI Conversational Analytics*, *Automated Data Visualization*, *Dynamic Excel Export*, dan *OCR Data Intake*, serta memperkuat pengelolaan memori percakapan untuk mendukung pertanyaan lanjutan; Melakukan revisi tampilan antarmuka *chatbot* dan *dashboard*, menjalankan pengujian regresi menyeluruh, memastikan data dan riwayat tersimpan dengan baik, serta menyusun dokumentasi akhir platform dalam berkas `README.md`.

3.4 Perangkat Penunjang Pelaksanaan Magang

Selama pelaksanaan kegiatan magang, pemanfaatan berbagai perangkat lunak (*software*), perangkat keras (*hardware*), serta kerangka kerja (*framework*) dan *library* menjadi komponen penting yang menunjang proses perancangan Platform AI Reporting. Adapun penjelasan mengenai masing-masing komponen tersebut akan diuraikan pada bagian-bagian berikut.

3.4.1 Perangkat Lunak yang Digunakan

Perangkat lunak (*software*) merupakan sekumpulan program yang berfungsi untuk mengendalikan operasi komputer serta menjalankan berbagai aplikasi di

dalamnya [55]. Keberadaan perangkat lunak tersebut berperan dalam mendukung kelancaran proses perancangan Platform AI Reporting. Oleh karena itu, perangkat lunak yang digunakan selama pelaksanaan kegiatan magang akan diuraikan sebagai berikut.

Tabel 3.2. Daftar perangkat lunak, spesifikasinya (versi), dan kegunaannya

Nama Perangkat	Spesifikasi (Versi)	Kegunaan
Visual Studio Code (VS Code)	1.90.1	*Code Editor*
Docker	28.0.1	*Platform Containerization*
Git	2.45.2.windows.1	*Version Control System*
Postman	v11.40.2-250409-1302	Pengujian API *endpoint*
Python	3.12.3	Bahasa pemrograman utama
XAMPP	3.3.0	Sistem Basis Data menggunakan MySQL

3.4.2 Perangkat Keras yang Digunakan

Perangkat keras (*hardware*) merupakan komponen fisik yang berfungsi untuk menerima, mengolah, menyimpan, serta menghasilkan keluaran berupa informasi dari hasil pemrosesan data [55]. Keberadaan perangkat keras memiliki peran penting dalam menunjang kelancaran proses perancangan Platform AI Reporting. Oleh karena itu, perangkat keras yang digunakan selama pelaksanaan kegiatan magang akan dijelaskan sebagai berikut.

Tabel 3.3. Daftar perangkat keras dan spesifikasinya

Nama Perangkat	Spesifikasi
Laptop Pribadi	Asus ROG Zephyrus G16 GU603VV (2023)
Processor	13th Gen Intel® Core™ i7-13620H Processor 2.4 GHz (24M Cache, up to 4.9 GHz, 10 cores: 6 P-cores and 4 E-cores)
RAM (Random Access Memory)	16 GB DDR4
Lanjut pada halaman berikutnya	

Lanjutan Tabel 3.3

Nama Perangkat	Spesifikasi
Storage	1 TB PCIe® 4.0 NVMe™ M.2 SSD
Operating System	Windows 11 Home 64-bit
Display	16-inch FHD+ 165Hz 16:10 (1920 x 1200, WUXGA) (Non-Touch)
GPU (Graphics Processing Unit)	NVIDIA® GeForce RTX™ 4060

3.4.3 Kerangka Kerja dan *Library* yang Digunakan

Framework atau kerangka kerja merupakan teknologi yang digunakan untuk mempermudah proses pengembangan sistem, baik aplikasi berbasis web maupun aplikasi *desktop* [56]. Sementara itu, *library* atau pustaka kode sumber (*source code library*) merupakan kumpulan referensi kode program yang dapat dimanfaatkan oleh pengembang perangkat lunak untuk mendukung proses pembelajaran, evaluasi, serta pengembangan perangkat lunak secara lebih efisien. Pustaka kode sumber tersebut menyediakan berbagai fasilitas, seperti dokumentasi, contoh implementasi, bantuan dalam perbaikan *bug*, penjelasan kesalahan, serta potongan kode, *template*, pola, dan rekomendasi pemrograman yang berguna [57]. Dengan dukungan *framework* dan *library* tersebut, proses perancangan Platform AI Reporting dapat dilaksanakan secara lebih optimal. Adapun *framework* dan *library* yang digunakan selama pelaksanaan kegiatan magang akan diuraikan sebagai berikut.

Tabel 3.4. Daftar *framework* & *library* dan versinya

Nama	Versi	Nama	Versi
aiohappyeyeballs	2.6.1	aiohttp	3.13.2
aiosignal	1.4.0	alembic	1.17.2
annotated-types	0.7.0	anyio	4.11.0
asttokens	3.0.1	attrs	25.4.0
blinker	1.9.0	certifi	2025.11.12
charset-normalizer	3.4.4	click	8.3.1
cloudinary	1.44.1	colorama	0.4.6
Lanjut pada halaman berikutnya			

Lanjutan Tabel 3.4

Nama	Versi	Nama	Versi
contourpy	1.3.3	cycler	0.12.1
dataclasses-json	0.6.7	decorator	5.2.1
distro	1.9.0	executing	2.2.1
Flask	3.1.2	flask-cors	6.0.1
Flask-Migrate	4.1.0	Flask-SQLAlchemy	3.1.1
fonttools	4.60.1	frozenlist	1.8.0
greenlet	3.2.4	h11	0.16.0
httpcore	1.0.9	httpx	0.28.1
httpx-sse	0.4.3	idna	3.11
ipython	9.7.0	ipython_pygments_lexers	1.1.1
itsdangerous	2.2.0	jedi	0.19.2
Jinja2	3.1.6	jiter	0.12.0
jsonpatch	1.33	jsonpointer	3.0.0
kiwisolver	1.4.9	langchain	1.1.0
langchain-classic	1.0.0	langchain-community	0.4.1
langchain-core	1.1.0	langchain-openai	1.1.0
langchain-text-splitters	1.0.0	langgraph	1.0.4
langgraph-checkpoint	3.0.1	langgraph-prebuilt	1.0.5
langgraph-sdk	0.2.10	langsmith	0.4.49
Mako	1.3.10	MarkupSafe	3.0.3
marshmallow	3.26.1	matplotlib	3.10.7
matplotlib-inline	0.2.1	multidict	6.7.0
mypy_extensions	1.1.0	mysql-connector-python	9.5.0
numpy	2.2.6	openai	2.8.1
opencv-python-headless	4.12.0.88	orjson	3.11.4
ormsgpack	1.12.0	packaging	25.0
pandas	2.3.3	parso	0.8.5

Lanjut pada halaman berikutnya

Lanjutan Tabel 3.4

Nama	Versi	Nama	Versi
pillow	12.0.0	pip	25.2
prompt_toolkit	3.0.52	propcache	0.4.1
pure_eval	0.2.3	pydantic	2.12.5
pydantic_core	2.41.5	pydantic-settings	2.12.0
Pygments	2.19.2	PyJWT	2.10.1
PyMySQL	1.1.2	pyparsing	3.2.5
PyPDF2	3.0.1	python-dateutil	2.9.0.post0
python-dotenv	1.2.1	pytz	2025.2
PyYAML	6.0.3	redis	7.1.0
regex	2025.11.3	requests	2.32.5
requests-toolbelt	1.0.0	six	1.17.0
sniffio	1.3.1	SQLAlchemy	2.0.44
stack-data	0.6.3	tenacity	9.1.2
tiktoken	0.12.0	tqdm	4.67.1
traitlets	5.14.3	typing_extensions	4.15.0
typing-inspect	0.9.0	typing-inspection	0.4.2
tzdata	2025.2	urllib3	2.5.0
wcwidth	0.2.14	Werkzeug	3.1.3
xlsxwriter	3.2.9	xxhash	3.6.0
yarl	1.22.0	zstandard	0.25.0
axios	^1.10.0	framer-motion	^12.19.1
gsap	^3.13.0	lucide-react	^0.518.0
next	14.0.0	react	^18
react-dom	^18	*types/node*	^20.19.1
types/react	^18	*types/react − dom*	^18
autoprefixer	^10	eslint	^8
eslint-config-next	14.0.0	postcss	^8
tailwindcss	^3	typescript	^5

3.5 Proses Pelaksanaan Magang

Uraian berikut menjelaskan proses perancangan Platform AI Reporting pada perusahaan AI.DECE, yang meliputi tahapan perancangan fitur-fitur utama

seperti *AI Conversational Analytics*, *Automated Data Visualization*, *Dynamic Excel Export*, dan *OCR Data Intake*, serta perancangan tampilan antarmuka, diagram, implementasi kode pemrograman beserta penjelasan, dan hasil pengujiannya.

3.5.1 Analisis Cara Kerja dan Perencanaan Perancangan

A Identifikasi Kebutuhan Awal Perancangan

Sebelum dilakukan proses perancangan, Platform AI Reporting didasari atas pemaparan yang dibutuhkan oleh klien perusahaan AI.DECE yang bergerak di bidang penjualan. Tentu hal ini melibatkan beberapa parameter yang sekaligus sebagai data utama yang digunakan untuk proses pengolahan, analisa, dan pelaporan data, sehingga belum sepenuhnya mampu memberikan pengalaman interaktif serta efisiensi yang optimal bagi pengguna. Selain itu, tampilan antarmuka juga membatasi kemudahan akses dan interaksi pengguna dan harus dirancang dari awal. Penjabaran atas permasalahan yang dihadapi sesuai kebutuhan klien dijelaskan sebagai berikut.

1. **Fitur AI Conversational Analytics**
   Fitur *AI Conversational Analytics* dirancang untuk dapat memberikan respons cerdas terhadap data yang diunggah dan menjadikannya sebagai *knowledge base*. Selain itu, fitur ini juga mendukung percakapan berkelanjutan atau *follow-up questions* sehingga interaksi antara pengguna dan sistem menjadi lebih natural dan informatif terkait dengan analisa data.

2. **Fitur Automated Data Visualization**
   Fitur *Automated Data Visualization* dirancang untuk memungkinkan pengguna berinteraksi dengan sistem dalam bentuk visualisasi grafik sebagai salah satu hasil pelaporan data dengan grafik yang sesuai diinginkan oleh pengguna.

3. **Fitur Dynamic Excel Export**
   Fitur *Dynamic Excel Export* dirancang untuk memberikan dalam bentuk format Excel sebagai hasil pelaporan sesuai yang diinginkan oleh pengguna.

4. **Fitur OCR Data Intake**
   Fitur *Ekstraksi OCR Data Intake* dirancang untuk mempermudah admin dalam melakukan proses penginputan data. Awalnya, proses penginputan

data dilakukan secara manual atau menggunakan Excel memakan konsumsi sumber daya dan waktu dalam jumlah yang banyak. Maka dari itu, proses pengunggahan dokumen PDF dan ekstraksinya menggunakan teknologi OCR akan mempermudah klien dalam proses pemasukan data.

5. **Tampilan Antarmuka Pengguna**
   Tampilan antarmuka pengguna dirancang agar platform dapat diakses secara langsung melalui *frontend* berbasis web. Antarmuka ini berfungsi untuk memvisualisasikan data, mengelola percakapan, serta mempermudah pengguna dalam berinteraksi dengan seluruh fitur yang terdapat pada Platform AI Reporting.

B Arsitektur dan Teknologi yang Digunakan

Platform AI Reporting dibangun dengan mengintegrasikan sejumlah teknologi terkini di bidang *Artificial Intelligence*, pemrosesan bahasa alami (NLP), dan pengembangan aplikasi web. Kombinasi teknologi-teknologi tersebut memungkinkan terciptanya sebuah ekosistem yang adaptif dan mampu memberikan respons secara cerdas. Bagian berikut menjelaskan setiap elemen teknologi beserta kontribusinya terhadap keseluruhan fungsionalitas sistem.

1) LangChain

LangChain adalah kerangka kerja *open-source* berbasis Python untuk memfasilitasi pengembangan aplikasi LLM dengan arsitektur modular yang terorganisir. Kerangka kerja ini mengintegrasikan komponen esensial seperti RAG, pemuat dokumen, pengelolaan memori, basis data vektor, dan pemformatan *prompt*, serta menyediakan lapisan abstraksi berupa *chains*, *agents*, dan *tools* untuk menyederhanakan pengembangan aplikasi *chatbot* dan sistem QA kontekstual. Pada Platform AI Reporting, LangChain digunakan untuk menyusun *prompt* dan historis dialog, mengarahkan eksekusi LangGraph, membangkitkan dan menjalankan kueri SQL, serta menyajikan hasil atau visualisasi sebagai *output chatbot* [58], [59].

LangChain dipilih karena fleksibilitas, modularitas, dan ekosistem *open-source*-nya yang dinamis. Dibandingkan pendekatan konvensional, LangChain menyediakan mekanisme pengelolaan data dan koordinasi antar-komponen yang lebih terstruktur dengan kemampuan *reasoning* berbasis konteks dialog [59]. Alternatif seperti Haystack dan LlamaIndex tersedia namun lebih rumit dalam

implementasi, terbatas dalam konfigurasi, memerlukan penanganan status eksplisit, dan kurang optimal untuk skala *enterprise* [47]. Hal ini telah dicantumkan pada Tabel 1.1. Tantangan utama LangChain adalah penggunaan memori signifikan pada konteks besar dan dependensi kuat terhadap model LLM, namun dokumentasi lengkap dan integrasi bawaan dengan berbagai penyedia LLM menjadikannya opsi yang sangat kompetitif untuk sistem LLM profesional dan terintegrasi [59].

2) Flask

Flask adalah kerangka kerja web Python minimalis yang digunakan sebagai *backend* API pada Platform AI Reporting [44]. Fungsi utamanya meliputi pengelolaan siklus permintaan-respons, pemrosesan data dari *frontend* dengan memanfaatkan kerangka kerja LangChain dan proses model bahasa besar, serta mengembalikan hasil dalam format JSON [60]. Flask berperan sebagai jembatan komunikasi antara klien dan server dengan kelebihan berupa fleksibilitas tinggi, arsitektur ringan, dan kemudahan pengembangan [44].

Flask dipilih dibanding Django atau FastAPI karena kesederhanaan struktur, sintaks yang mudah dipelajari, dan integrasi natural dengan Python [61]. Meskipun FastAPI menawarkan kinerja superior, implementasinya memerlukan kurva pembelajaran dan konfigurasi yang lebih kompleks. Keterbatasan Flask adalah belum adanya dukungan bawaan untuk pemrosesan *asynchronous* yang dapat menghambat pada beban konkuren tinggi [62]. Namun, untuk sistem AI yang menekankan integrasi antar-komponen Python, Flask tetap menjadi pilihan relevan berkat kesederhanaan arsitektur dan dukungan komunitas yang kuat.

3) Large Language Models (LLM)

Model `gpt-5-mini` dari OpenAI yang diimplementasikan pada Platform AI Reporting berfungsi sebagai mesin utama untuk menghasilkan respons, klasifikasi dokumen, dan interaksi dialog cerdas pada fitur *chatbot*. Model tersebut melakukan analisis terhadap konteks pertanyaan dari pengguna, menggabungkannya dengan data yang diperoleh melalui SQLAgent dari basis data, lalu membentuk respons yang natural dan sesuai konteks dengan memanfaatkan *prompt* LangChain yang telah diperkuat oleh konteks hasil *retrieval* serta memori dari interaksi sebelumnya [63].

GPT-5 dipilih karena keunggulannya dalam berbagai evaluasi, termasuk mencapai peringkat 81.76 dari 403 soal pada LiveOIBench untuk pemrograman olimpiade [64], mengungguli model pendahulu dalam *benchmark* kimia ChemIQ

untuk tugas struktur molekul dan konversi SMILES ke IUPAC [65]. Kombinasi kemampuan *reasoning*, pemrograman, dan pemahaman konseptual ini relevan untuk penalaran *prompt* pengguna dan generasi kode SQLAgent pada Platform AI Reporting. Perbandingan performa dapat dilihat pada Gambar 3.1.

Model	Gold (%)	Medals (%)	Relative Score(%)	Human Percentile (%)	Pass Rate (%)	Elo
Proprietary LLMs						
GPT-5	**50.00**	**88.89**	**67.21**	**81.76**	**63.03**	**2414**
Gemini-2.5-Pro	31.94	77.78	51.33	71.80	44.46	2192
GPT-O3-Mini-High	26.39	72.22	47.69	64.28	44.19	2088
Gemini-2.5-Flash	15.28	62.5	41.29	56.81	36.06	1945
GPT-4.1	4.17	40.28	24.78	35.99	18.32	1482
Open-weight Thinking LLMs						
GPT-OSS-120B-High	**50.00**	87.50	62.78	72.88	60.14	2205
GPT-OSS-120B	29.17	73.61	49.23	59.90	47.78	2032
GPT-OSS-20B	19.44	68.06	42.36	53.94	42.80	1901
Seed-OSS	15.28	68.06	42.58	53.81	40.09	1873
Qwen3-32B	9.72	54.17	32.86	42.00	27.70	1665
DeepSeek-R1	6.94	52.78	33.43	42.29	28.87	1617
Qwen3-14B	5.56	45.83	27.24	34.59	22.73	1402
DeepSeek-R1-Distill-Llama-70B	1.39	33.33	20.50	32.30	16.88	1284
Open-weight Non-Thinking LLMs						
DeepSeek-V3	4.17	34.72	21.70	31.76	17.10	1283
Qwen3-32B-Non-Thinking	1.39	16.67	12.92	24.64	8.78	1040

Gambar 3.1. Perbandingan model LLM berdasarkan kemampuan *reasoning*, pemrograman, dan penalaran konseptual

Sumber: [64]

Kendala GPT-5 meliputi biaya operasional tinggi, dependensi infrastruktur *cloud* OpenAI, risiko privasi data, dan potensi kesalahan pada detail kritis terutama domain medis [65], [66]. Meski demikian, GPT-5 tetap menjadi pilihan primer berkat kualitas respons, kapabilitas penalaran, dan ekosistem pengembangan yang matang untuk sistem seperti Platform AI Reporting.

4) Cloudinary

Cloudinary adalah layanan pengelolaan aset media berbasis *cloud* yang diintegrasikan untuk menyimpan hasil OCR dokumen (PDF), dokumen Excel dari fitur *Dynamic Excel Export*, dan gambar *plot* dari *Automated Data Visualization*. Layanan ini memungkinkan penyimpanan efisien dan akses melalui URL aman dengan modifikasi *real-time* menggunakan parameter spesifik.

Keputusan menggunakan Cloudinary sebagai platform penyimpanan dilatarbelakangi oleh keunggulannya dalam menyajikan visualisasi hasil ekstraksi dokumen yang dapat ditampilkan secara langsung pada antarmuka pengguna dengan proses integrasi yang sederhana. Opsi lain seperti Google Drive kurang kompetitif karena keterbatasan kapasitas pada paket gratisnya serta skema penetapan harga yang tidak ideal untuk implementasi skala perusahaan. Walaupun

terdapat batasan kuota pada tier gratis Cloudinary dan tantangan dalam menangani berkas berukuran besar secara langsung, namun karakteristik adaptabilitasnya yang tinggi, kemudahan dalam proses pengintegrasian, serta tersedianya fitur pengolahan media secara otomatis menjadikan platform ini sebagai pilihan paling sesuai untuk mengelola aset media digital dalam sistem kecerdasan buatan yang berbasis web.

5) JSON Web Token (JWT)

JSON Web Token (JWT) adalah standar terbuka untuk representasi klaim terverifikasi dalam bentuk objek JSON yang tersandi atau bertanda tangan digital [67]. Dalam Platform AI Reporting, JWT digunakan untuk otentikasi (*login*, *register*, *sessionless authentication*), pertukaran identitas antar layanan mikro, dan delegasi akses API untuk ekstraksi OCR dan visualisasi, dengan implementasi *signing* menggunakan kunci rahasia pada .env repositori *backend* [67].

Kelebihan JWT meliputi sifat *stateless* yang memudahkan skala horizontal tanpa penyimpanan sesi terpusat, ukuran ringkas untuk *header* HTTP, dan fleksibilitas klaim kustom untuk metadata seperti peran pengguna dan ruang lingkup akses, menjadikannya ideal untuk arsitektur *microservice* dan *API-first*. Namun, JWT memiliki kelemahan pada pencabutan token karena *stateless*, risiko keamanan jika kunci bocor, dan *overhead* kinerja untuk klaim besar. Mitigasi dilakukan melalui TTL singkat, *refresh token*, penyimpanan kunci aman, dan pengawasan akses ketat [67], [68].

6) Optical Character Recognition (OCR)

Optical Character Recognition (OCR) adalah teknik mengonversi citra teks (seperti *scan* PDF dan gambar dokumen) menjadi teks digital yang dapat diproses mesin [69], [70]. Pada Platform AI Reporting, OCR berfungsi mengekstrak teks, tabel, dan metadata dari PDF transaksi atau dokumen bukti untuk disimpan ke basis data dan dianalisis lebih lanjut.

Kelebihan OCR modern berasal dari integrasi *deep learning* dan model multimodal yang menawarkan akurasi tinggi dalam mengenali teks dengan variasi font, tata letak kompleks, dan struktur tidak beraturan. Model seperti GPT-4o memiliki kemampuan OCR yang lebih adaptif karena dapat mengekstraksi teks sekaligus memahami konteks visual dokumen, termasuk tabel dan elemen non-teks, bahkan melampaui performa OCR tradisional pada dokumen historis dan tulisan tangan [71], [72]. Pendekatan ini melengkapi OCR konvensional seperti Tesseract atau PaddleOCR yang tetap efektif untuk kasus standar namun

memerlukan komputasi berat lokal dan sulit diintegrasi secara *cloud*.

Kekurangan OCR meliputi sensitivitas terhadap kualitas input (hasil *scan* buram, kontras rendah, *noise*), tantangan pada dokumen non-Latin atau tulisan tangan, serta kesalahan interpretasi struktur dokumen [69], [73]. Selain itu, *pipeline* OCR akurat memerlukan *fine-tuning* pada dataset domain-spesifik, infrastruktur GPU untuk model besar, dan sering memerlukan verifikasi manusia untuk menjamin kualitas data produksi, yang menambah biaya dan kompleksitas integrasi.

7) Plot Agent

Plot Agent adalah komponen agen perangkat lunak yang mengubah permintaan atau hasil analisis menjadi visualisasi, seperti *pie chart*, *histogram*, *scatter plot*, atau *line*, menggunakan *library* Python seperti Matplotlib dan Seaborn. Dalam Platform AI Reporting, *Plot Agent* menerima parameter atau kodingan Python untuk menghasilkan skrip *plotting* yang dapat dieksekusi dan merender gambar berkualitas dengan kemampuan anotasi, legenda dinamis, serta metadata untuk aksesibilitas [74, 75].

Kelebihan *Plot Agent* meliputi otomatisasi *end-to-end* dari data dan instruksi bahasa alami ke visualisasi, konsistensi *styling* korporat, dan kemampuan menghasilkan kode yang dapat direplikasi serta dikustomisasi lebih lanjut [75]. Skalabilitas dari pembuatan grafik laporan hingga respons interaktif pada *dashboard* analitik, serta integrasi dengan kerangka kerja *multi-agent* dapat menambah fitur seperti deskripsi otomatis dan kolaborasi antar agen [76, 77].

Kekurangan yang harus diperhatikan adalah kebutuhan validasi otomatis, kompleksitas pengendalian opsi *styling* dan kompatibilitas antar *library*, serta ketergantungan pada model bahasa yang memerlukan *prompting* yang hati-hati. Oleh karena itu diperlukan *pipeline testing* dan kebijakan *review* untuk menjamin keandalan visualisasi yang dihasilkan [74, 75].

Secara keseluruhan, kombinasi teknologi yang diterapkan pada Platform AI Reporting yang telah disebutkan sebelumnya membentuk ekosistem AI modern yang terintegrasi secara menyeluruh. Integrasi ini menghasilkan sistem yang cerdas, adaptif, dan efisien dalam memberikan interaksi kontekstual serta pengolahan data secara *real-time*. Meskipun masing-masing teknologi memiliki keterbatasan tertentu, kolaborasi antar komponen tersebut menciptakan fondasi kuat bagi pengembangan platform AI yang tidak hanya mampu memahami konteks data dan dokumen, tetapi juga menghasilkan respons yang akurat, cepat, dan relevan untuk mendukung kebutuhan analisis dan otomasi cerdas di era digital saat ini.

C Metodologi Perancangan

Metodologi perancangan yang diterapkan pada perusahaan AI.DECE menggunakan kerangka kerja PDCA (*Plan-Do-Check-Action*) sebagai pendekatan untuk memastikan proses perancangan berlangsung secara sistematis dan terstruktur. PDCA berfungsi sebagai siklus evaluasi berkelanjutan yang membantu meningkatkan kualitas serta efektivitas sistem dalam setiap tahap perancangannya.

Gambar 3.2. Siklus PDCA dalam perancangan

Sumber: [29]

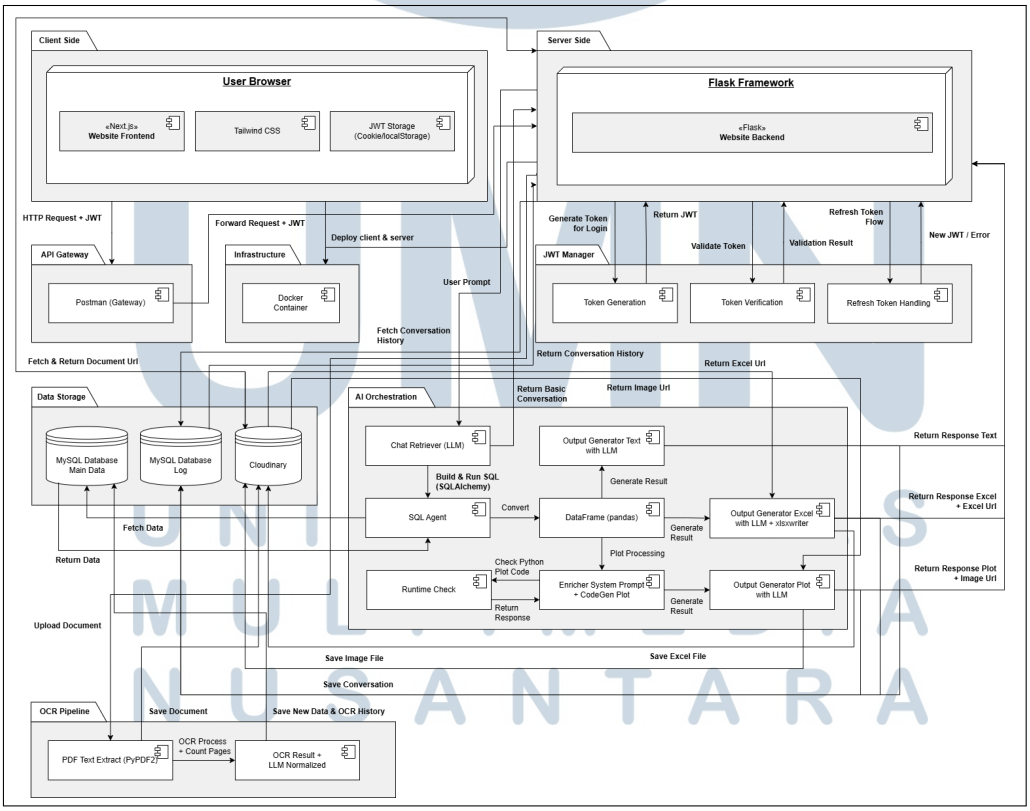
Penerapan kerangka kerja PDCA dalam perancangan Platform AI Reporting diuraikan sebagai berikut:

1. **Plan (Perencanaan):** Mencakup analisis kebutuhan sistem, penyusunan *flowchart*, perancangan keempat fitur utama Platform AI Reporting, serta pembagian *sprint* kerja guna memastikan setiap tahapan proyek tersusun secara terstruktur.
2. **Do (Pelaksanaan):** Tahap implementasi dilakukan melalui proses pemrograman, mencakup *scripting*, *setup environment*, serta integrasi antar komponen teknologi seperti LangChain, LLM, Flask, OCR, dan lain sebagainya yang mendukung fungsi utama platform.
3. **Check (Pemeriksaan):** Hasil implementasi diuji melalui serangkaian pengujian fungsional dan integrasi, kemudian dievaluasi bersama supervisor dan *Technical Team Manager* untuk memastikan kesesuaian dengan tujuan sistem yang dirancang.

4. **Act (Tindakan):** Berdasarkan hasil evaluasi dan umpan balik pengguna, dilakukan optimalisasi dan perbaikan pada Platform AI Reporting agar sistem semakin stabil, responsif, dan sesuai dengan kebutuhan pengguna akhir.

D Architecture Diagram

Visualisasi struktur sistem perangkat lunak dapat diwujudkan melalui *architecture diagram*, yaitu suatu skema yang memetakan elemen-elemen pokok beserta keterkaitan di antara elemen tersebut. Fungsi utama dari skema ini adalah menyajikan pemahaman komprehensif tentang konfigurasi teknis sistem, mencakup mekanisme interaksi antara *user interface*, peladen, pangkalan data, dan unsur-unsur penunjang lain yang terlibat. Peranan skema arsitektural ini sangat krusial dalam memaparkan bagaimana informasi mengalir dan bagaimana operasi-operasi dieksekusi dalam sistem, sehingga aktivitas perancangan, implementasi, maupun perawatan sistem dapat diorganisir dengan lebih sistematis dan produktif [78]. Skema arsitektural yang diimplementasikan untuk Platform AI Reporting dapat dilihat pada Gambar 3.3.



Gambar 3.3. *Architecture diagram* pada Platform AI Reporting

Platform AI Reporting dirancang menggunakan arsitektur terdistribusi yang memisahkan tanggung jawab berdasarkan lapisan klien, server, orkestrasi AI, penyimpanan data, dan pemrosesan OCR. Pada sisi klien, pengguna berinteraksi melalui antarmuka web berbasis *Next.js* yang bergaya menggunakan *Tailwind CSS*. Seluruh permintaan yang membutuhkan autentikasi akan menyertakan *JWT* yang sebelumnya disimpan di sisi klien melalui *localStorage*. Permintaan dari pengguna akan dilewatkan melalui *API Gateway* yang pada diagram diwakili oleh *Postman* sebagai media uji coba dan pengujian API. Seluruh komunikasi kemudian diteruskan menuju *Flask Backend*. Saat melakukan *deploy* menggunakan *Docker Container* untuk memastikan portabilitas dan konsistensi lingkungan sistem.

Pada sisi server, *Flask Framework* berfungsi sebagai komponen utama dalam memproses *request* pengguna, melakukan koordinasi ke dalam modul-modul pemrosesan AI maupun ke basis data, dan mengembalikan respons ke klien. Mekanisme autentikasi dipisahkan pada *JWT Manager* yang menangani pembuatan token, verifikasi token, serta *refresh token handling*. Garis pada diagram menunjukkan bahwa seluruh alur autentikasi satu pintu selalu melalui lapisan ini sebelum diizinkan mengakses fitur internal seperti analitik data dan CRUD yang hanya khusus untuk peran tertentu seperti admin.

Komponen inti dari fitur utama platform berada pada bagian *AI Orchestration*. Alur dimulai dari *Chat Retriever (LLM)* yang menerima kombinasi antara *prompt* pengguna dan *system prompt*, kemudian berkolaborasi dengan SQLAgent. Agen ini akan melakukan *Build & Run SQL* menggunakan *SQLAlchemy* untuk mengambil data dari *MySQL Database Main Data*. Hasil eksekusi kueri dikonversi menjadi *DataFrame* melalui pustaka *pandas* yang selanjutnya dapat diolah lebih lanjut untuk menghasilkan keluaran berbentuk teks, grafik, atau file Excel. Untuk menghasilkan grafik, alur berlanjut pada komponen *Plot Processing* diikuti *Runtime Check* yang memastikan keamanan skrip Python yang akan dieksekusi. Setelah validasi, instruksi pemrosesan diperkaya menggunakan *Enricher System Prompt + CodeGen Plot* sebelum akhirnya hasil visual dikirim kembali ke pengguna melalui *Output Generator Plot with LLM*. Sementara itu, alur keluaran berupa teks langsung diteruskan dari *Output Generator Text with LLM*, dan untuk ekspor excel memanfaatkan *Output Generator Excel with LLM + xlsxwriter*. Seluruh proses pada bagian ini terlihat jelas terhubung melalui garis yang memusat menuju hasil akhir *Generate Result*.

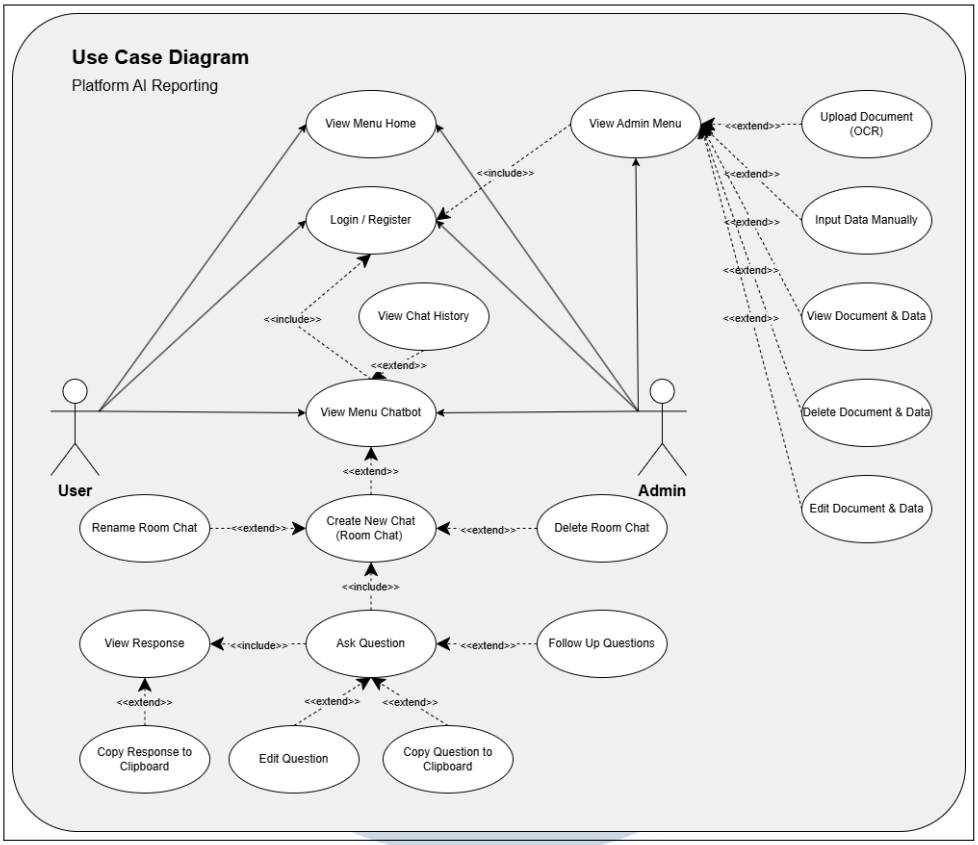
Fitur *OCR Data Intake* menjadi jalur masuk data bagi admin untuk mengunggah PDF yang kemudian diproses pada *pipeline* OCR. Dimulai dari *PDF*

Text Extract (PyPDF2) untuk menghitung halaman serta mengambil konten dasar, kemudian dilanjutkan pada tahap *OCR Process + Count Pages* untuk mendeteksi serta menormalisasi teks menggunakan LLM. Hasil akhirnya disimpan kembali ke dalam basis data sehingga admin mampu melakukan koreksi jika ditemukan kesalahan ekstraksi teks. File yang diunggah dan aset hasil pengolahan disimpan secara eksternal melalui layanan *Cloudinary* yang tergambar memiliki koneksi dua arah ke *pipeline* OCR serta *MySQL Database Log Data* untuk pencatatan aktivitas.

Arsitektur data menggunakan *MySQL Database* yang terpisah menjadi basis data utama dan basis data log sehingga operasi transaksi dan audit dapat berjalan optimal. Selain itu, terdapat fitur histori percakapan yang disimpan melalui mekanisme penyimpanan data internal agar pengguna dapat kembali pada percakapan sebelumnya layaknya *ChatGPT*. Secara keseluruhan, garis-garis koneksi dalam diagram menggambarkan aliran data yang terstruktur—dimulai dari browser, melewati *gateway*, diverifikasi melalui *JWT Manager*, diproses oleh *backend* dan *AI Orchestration*, hingga akhirnya hasil dikembalikan ke pengguna dalam bentuk respons informatif yang aman, cepat, dan fleksibel.

E Use Case Diagram

Unified Modeling Language (UML) adalah standar bahasa pemodelan untuk keperluan perancangan, visualisasi, dan dokumentasi sistem informasi berorientasi objek. Berbagai diagram dalam UML menawarkan perspektif yang beragam selama proses pengembangan, memfasilitasi pemahaman visual atas struktur statis dan dinamika perilaku sistem. *Use Case Diagram* merupakan salah satu diagram yang umum dimanfaatkan untuk merepresentasikan fungsi-fungsi sistem berdasarkan perspektif pengguna. Diagram tersebut mengilustrasikan bagaimana aktor (pengguna) berinteraksi dengan sistem untuk memenuhi objektif spesifik, dan memegang peranan krusial pada fase analisis persyaratan fungsional. Melalui hal ini, *Use Case Diagram* berkontribusi dalam menyamakan persepsi antara pengembang dan pengguna pada fase inisial desain perangkat lunak [79]. Berdasarkan hal tersebut, *Use Case Diagram* untuk Platform AI Reporting dapat dilihat pada Gambar 3.4.



Gambar 3.4. *Use case diagram* pada Platform AI Reporting

Diagram *use case* yang disajikan merepresentasikan ruang lingkup fungsional Platform AI Reporting yang telah diperbarui sesuai dengan kebutuhan sistem berbasis percakapan dan pengelolaan data. Diagram ini melibatkan dua aktor utama, yaitu *User* dan *Admin*, yang masing-masing memiliki hak akses dan tanggung jawab berbeda terhadap sistem.

Baik *User* maupun *Admin* dapat mengakses antarmuka utama sistem melalui *View Menu Home* serta melakukan autentikasi menggunakan *Login / Register*. Proses autentikasi menjadi prasyarat untuk mengakses fitur lanjutan, yang ditunjukkan melalui relasi *include* pada *View Menu Chatbot* dan *View Admin Menu*. Hal ini menegaskan bahwa seluruh interaksi inti sistem bersifat terproteksi dan memerlukan identitas pengguna yang valid.

Fitur utama sistem direpresentasikan melalui modul *Chatbot*, yang dapat diakses oleh kedua aktor melalui *View Menu Chatbot*. Di dalam modul ini, pengguna dapat melihat riwayat percakapan (*View Chat History*) serta membuat sesi percakapan baru (*Create New Chat (Room Chat)*), yang dimodelkan sebagai relasi *extend* karena bersifat opsional dan bergantung pada kebutuhan pengguna.

Setiap *Room Chat* merepresentasikan satu konteks percakapan yang terpisah.

Proses inti percakapan dimulai dari *Ask Question*, yang memiliki relasi *include* terhadap *Create New Chat (Room Chat)*. Relasi ini menunjukkan bahwa setiap pertanyaan yang diajukan oleh *User* atau *Admin* harus berada dalam sebuah *Room Chat*, baik dengan membuat room baru maupun menggunakan room yang sudah ada. Setelah pertanyaan diajukan, sistem akan selalu menampilkan hasil pemrosesan melalui *View Response*, yang dimodelkan sebagai relasi *include* karena respons merupakan konsekuensi langsung dari proses bertanya.

Fungsionalitas tambahan pada alur percakapan direpresentasikan melalui relasi *extend*. Pengguna dapat melakukan *Follow Up Questions* untuk memperdalam konteks analisis, mengedit pertanyaan (*Edit Question*), serta menyalin pertanyaan ke papan klip (*Copy Question to Clipboard*). Pada sisi respons, pengguna juga dapat menyalin hasil jawaban chatbot melalui *Copy Response to Clipboard*. Selain itu, *User* memiliki kemampuan untuk mengganti nama *Room Chat* (*Rename Room Chat*), sedangkan *Admin* memiliki kewenangan tambahan untuk menghapus *Room Chat* (*Delete Room Chat*), yang keduanya dimodelkan sebagai ekstensi dari *Create New Chat (Room Chat)*.

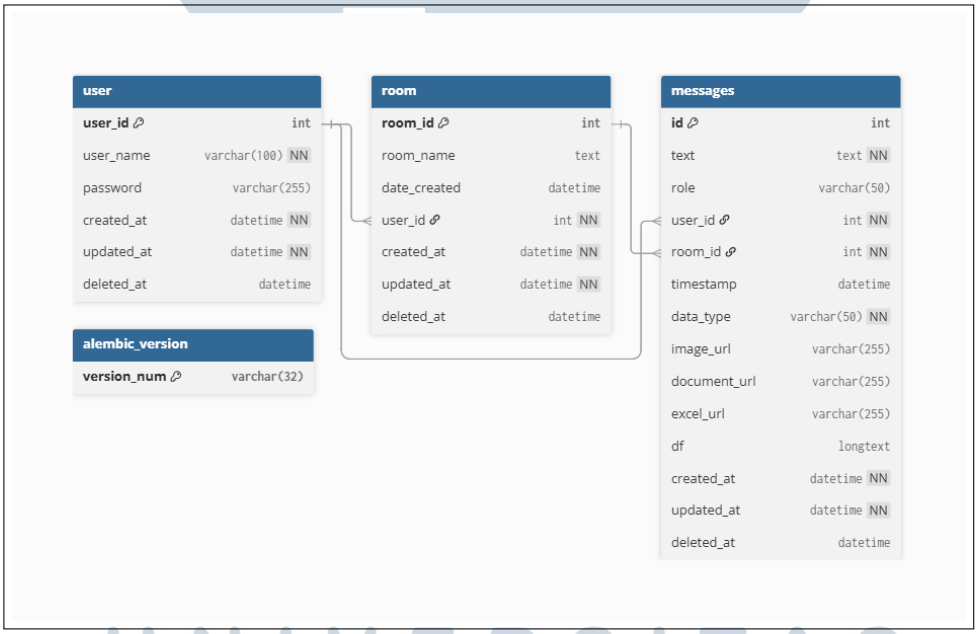
Aktor *Admin* memiliki akses eksklusif ke *View Admin Menu*, yang menjadi pusat pengelolaan sumber data sistem. Dari menu ini, *Admin* dapat melakukan unggah dokumen menggunakan OCR (*Upload Document (OCR)*), memasukkan data secara manual (*Input Data Manually*), melihat data yang tersimpan (*View Document & Data*), serta melakukan pengubahan (*Edit Document & Data*) dan penghapusan data (*Delete Document & Data*). Seluruh aktivitas tersebut dimodelkan sebagai relasi *extend* karena hanya dijalankan ketika *Admin* memilih fungsi tertentu dari menu administrasi.

Secara keseluruhan, *use case diagram* ini menekankan bahwa Platform AI Reporting berpusat pada interaksi percakapan berbasis konteks *Room Chat*, yang terintegrasi erat dengan mekanisme autentikasi, penyimpanan histori, serta pengelolaan data oleh *Admin*. Pendekatan ini memastikan bahwa setiap analisis yang dihasilkan oleh chatbot memiliki konteks yang jelas, data sumber yang terkelola dengan baik, serta kontrol akses yang sesuai dengan peran pengguna.

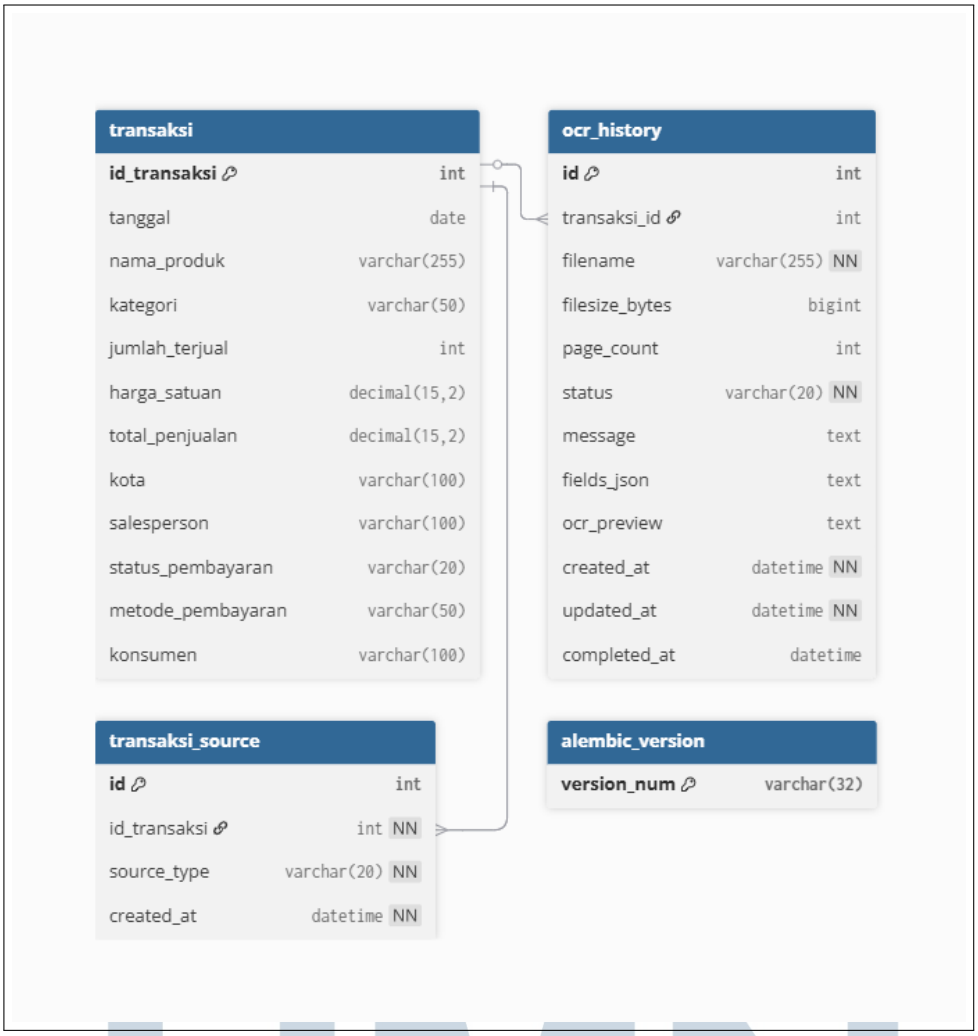
F Entity Relationship Diagram

Diagram basis data atau yang dikenal sebagai *Entity Relationship Diagram* (ERD) merupakan model konseptual tingkat tinggi yang digunakan dalam

perancangan basis data untuk merepresentasikan struktur logis suatu sistem melalui hubungan antara entitas, atribut, serta relasi yang dimilikinya. ERD berfungsi sebagai media visualisasi yang menggambarkan keterkaitan antar data di dalam sistem, sehingga membantu menjamin integritas, konsistensi, dan efisiensi struktur data sebelum dilakukan tahap implementasi fisik basis data. Selain itu, diagram ini memiliki peran penting sebagai sarana komunikasi antara analis sistem, pengembang, dan pemangku kepentingan guna menyamakan persepsi terkait kebutuhan data serta proses bisnis yang akan dimodelkan [80]. Oleh karena itu, ERD pada Platform AI Reporting dirancang menggunakan dua basis data terpisah, yaitu basis data `ai_reporting_data` yang digunakan untuk menyimpan data hasil OCR dan data penjualan sebagaimana ditunjukkan pada Gambar 3.6, serta basis data `ai_reporting_log` yang berfungsi untuk menyimpan data pengguna, ruang percakapan, serta log pesan atau percakapan sebagaimana ditampilkan pada Gambar 3.5.



Gambar 3.5. *Entity Relationship Diagram* untuk basis data log pada Platform AI Reporting



Gambar 3.6. *Entity Relationship Diagram* untuk basis data utama pada Platform AI Reporting

Sebagai bagian dari perancangan Platform AI Reporting, pembagian data ke dalam dua basis data dilakukan untuk mengantisipasi pertumbuhan volume data yang signifikan di masa mendatang. Dengan memecah data ke dua sistem basis data secara paralel, sistem dapat mendistribusikan beban I/O dan *query* ke dua server secara bersamaan, sehingga mencegah penurunan performa yang lazim terjadi pada sistem monolitik tunggal ketika data dan beban meningkat drastis. Hal ini dilakukan dengan pendekatan *Horizontal Scaling* atau *Database Partitioning* [81]. Teknik ini telah diidentifikasi dalam riset sebagai metode penskalaan horizontal yang efektif untuk meningkatkan skalabilitas dan ketersediaan sistem terhadap beban data dalam jumlah yang banyak [82].

1) Basis Data `ai_reporting_data`

Tabel 3.5. Struktur Tabel `alembic_version`

Atribut	Tipe Data	Constraint
`version_num`	varchar(32)	PK

- Tabel `alembic_version` bertugas untuk menyimpan versi migrasi skema basis data menggunakan Alembic.

Tabel 3.6. Struktur Tabel `transaksi`

Atribut	Tipe Data	Constraint
`id_transaksi`	int	PK, Auto Increment
`tanggal`	date	
`nama_produk`	varchar(255)	
`kategori`	varchar(50)	
`jumlah_terjual`	int	
`harga_satuan`	decimal(15,2)	
`total_penjualan`	decimal(15,2)	
`kota`	varchar(100)	
`salesperson`	varchar(100)	
`status_pembayaran`	varchar(20)	
`metode_pembayaran`	varchar(50)	
`konsumen`	varchar(100)	

- Tabel `transaksi` menyimpan seluruh data transaksi yang diinput secara manual, hasil import, atau hasil OCR. Data ini menjadi sumber analisis untuk fitur *AI Conversational Analytics*, *Automated Data Visualization*, dan *Dynamic Excel Export*.

Tabel 3.7. Struktur Tabel ocr_history

Atribut	Tipe Data	Constraint
id	int	PK, Auto Increment
transaksi_id	int	FK ke transaksi.id_transaksi, Null
filename	varchar(255)	Not Null
filesize_bytes	bigint	
page_count	int	Default 0
status	varchar(20)	Not Null
message	text	
fields_json	text	
ocr_preview	text	
created_at	datetime	Default: current_timestamp()
updated_at	datetime	Default: current_timestamp()
completed_at	datetime	

- Tabel ocr_history mencatat setiap proses OCR termasuk status, pesan, dan preview sehingga memudahkan audit data oleh admin.

Tabel 3.8. Struktur Tabel transaksi_source

Atribut	Tipe Data	Constraint
id	int	PK, Auto Increment
id_transaksi	int	Not Null, FK ke transaksi.id_transaksi
source_type	varchar(20)	Not Null
created_at	datetime	Default: current_timestamp()

- Tabel transaksi_source mencatat asal data transaksi, misalnya manual, OCR, upload, atau import, untuk keperluan audit dan penelusuran.

2) Basis Data `ai_reporting_log`

Tabel 3.9. Struktur Tabel `user`

Atribut	Tipe Data	Constraint
`user_id`	int	PK, Auto Increment
`user_name`	varchar(100)	Not Null, Unique
`password`	varchar(255)	
`created_at`	datetime	Default: current_timestamp()
`updated_at`	datetime	Default: current_timestamp()
`deleted_at`	datetime	

- Tabel `user` bertugas untuk menyimpan informasi pengguna sistem, digunakan untuk autentikasi JWT dan pengelolaan hak akses.

Tabel 3.10. Struktur Tabel `room`

Atribut	Tipe Data	Constraint
`room_id`	int	PK, Auto Increment
`room_name`	text	
`date_created`	datetime	
`user_id`	int	Not Null, FK ke `user.user_id`
`created_at`	datetime	Default: current_timestamp()
`updated_at`	datetime	Default: current_timestamp()
`deleted_at`	datetime	

- Tabel `room` bertugas untuk menyimpan ruang atau sesi percakapan pengguna dengan *chatbot*.

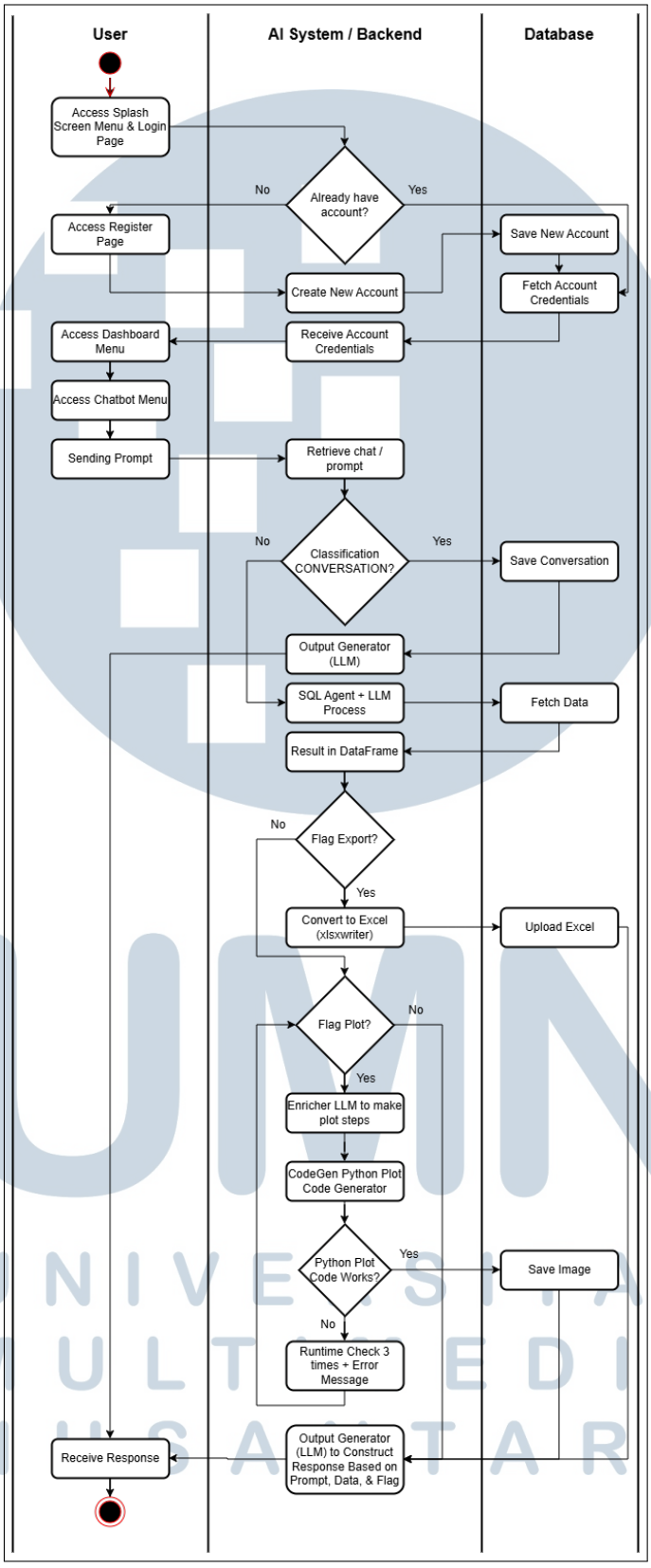
Tabel 3.11. Struktur Tabel `messages`

Atribut	Tipe Data	Constraint
`id`	int	PK, Auto Increment
`text`	text	Not Null
`role`	varchar(50)	
`user_id`	int	Not Null, FK ke `user.user_id`
`room_id`	int	Not Null, FK ke `room.room_id`
`timestamp`	datetime	
`data_type`	varchar(50)	Not Null
`image_url`	varchar(255)	
`document_url`	varchar(255)	
`excel_url`	varchar(255)	
`df`	longtext	
`created_at`	datetime	Default: current_timestamp()
`updated_at`	datetime	Default: current_timestamp()
`deleted_at`	datetime	

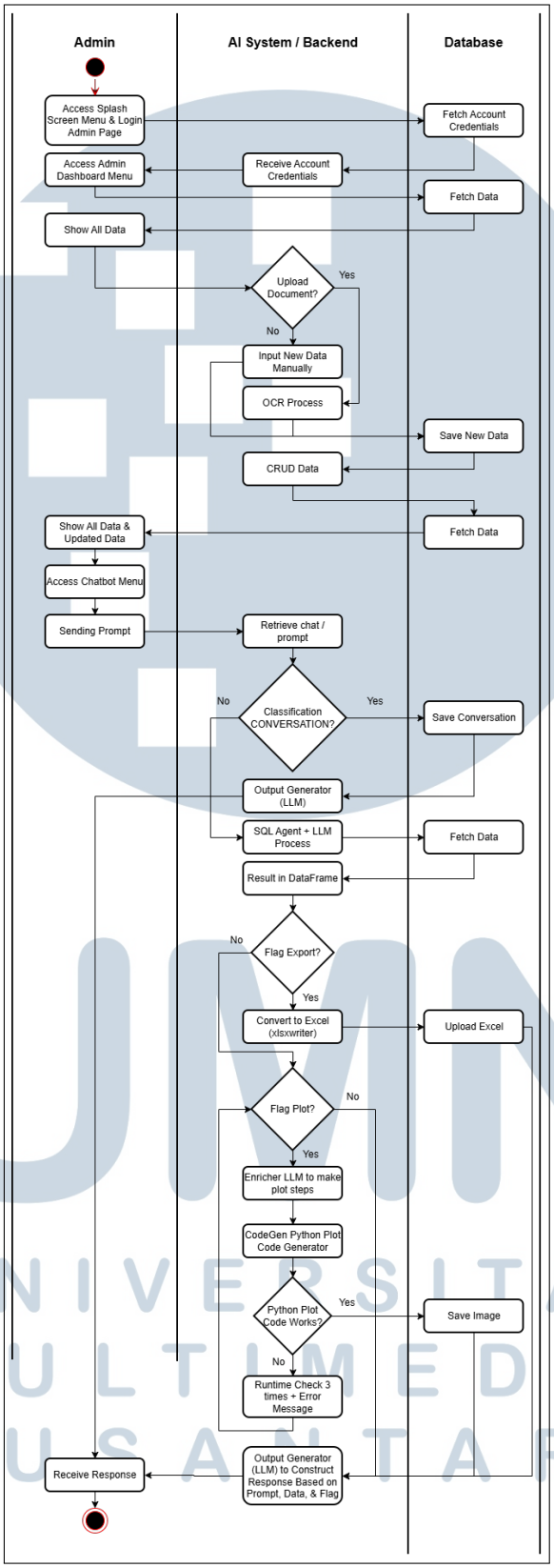
- Tabel `messages` bertugas untuk menyimpan seluruh interaksi percakapan, termasuk teks, media, dan hasil ekspor Excel, sehingga mendukung fitur histori percakapan pada sistem.

G Activity Diagram

Sebagai komponen dari UML, *activity diagram* berfungsi sebagai instrumen pemodelan yang mengilustrasikan rangkaian proses dan aktivitas dalam sebuah sistem. Diagram ini mendeskripsikan tahapan yang dieksekusi untuk mencapai suatu objektif spesifik, sekaligus memvisualisasikan hubungan antara berbagai aktivitas, mekanisme dalam menentukan keputusan, serta kemungkinan adanya proses yang berjalan secara bersamaan yang dapat berdampak pada alur eksekusi [83]. Melalui representasi grafis ini, dimungkinkan untuk memperoleh gambaran komprehensif terkait dinamika kerja sistem, logika pengambilan keputusan, dan hasil dari interaksi yang berlangsung di dalamnya [84]. Dalam konteks Platform AI Reporting, implementasi diagram aktivitas terbagi ke dalam dua kategori, yakni diagram aktivitas untuk *user* pada Gambar 3.7, serta diagram aktivitas untuk *admin* pada Gambar 3.8.



Gambar 3.7. *Activity diagram user* pada Platform AI Reporting



Gambar 3.8. *Activity diagram admin* pada Platform AI Reporting

Platform AI Reporting merupakan sebuah sistem terintegrasi yang dirancang untuk mendukung analitik berbasis percakapan dengan memanfaatkan *Large Language Model* (LLM), visualisasi data otomatis, ekspor dinamis ke format Excel, serta mekanisme *data intake* melalui unggahan dokumen secara manual dan *Optical Character Recognition* (OCR). Diagram aktivitas yang disajikan pada versi terbaru menggambarkan alur kerja *end-to-end* dari sistem ini dengan pemisahan yang jelas antara alur pengguna biasa (*user*) dan alur administrator (*admin*), dimana kedua alur tersebut berbagi mekanisme *chatbot* yang sama tetapi memiliki perbedaan pada proses autentikasi dan sumber data yang digunakan.

Pada alur pengguna, aktivitas dimulai ketika pengguna mengakses *splash screen* dan halaman autentikasi. Sistem menyediakan opsi untuk melakukan *register* bagi pengguna baru atau *login* bagi pengguna yang telah memiliki akun. Jika pengguna belum memiliki akun, sistem akan membuat akun baru dan menyimpan kredensial yang diperlukan. Sebaliknya, jika akun sudah ada, sistem akan mengambil kredensial tersebut untuk membentuk sesi aktif. Seluruh proses ini diasumsikan menggunakan mekanisme otorisasi berbasis token, seperti *JSON Web Token* (JWT), sehingga setiap permintaan lanjutan ke *dashboard* maupun ke layanan *chatbot* dilakukan dalam konteks sesi yang telah terverifikasi.

Setelah berhasil masuk ke sistem, pengguna dapat mengakses *dashboard* dan memilih menu *chatbot*. Ketika pengguna mengirimkan sebuah *prompt*, sistem akan memeriksa konteks percakapan yang ada. Pada diagram aktivitas, relasi antara *Ask Question* dan *Create New Chat (Room Chat)* menunjukkan bahwa tindakan pengiriman pertanyaan dapat menginisiasi pembuatan ruang percakapan baru apabila belum tersedia, sehingga pengelolaan konteks percakapan dilakukan secara implisit berdasarkan kebutuhan interaksi. Prompt yang dikirimkan pengguna kemudian dikombinasikan dengan *system prompt* yang telah didefinisikan oleh sistem untuk membentuk konteks lengkap sebelum diteruskan ke LLM.

Proses LLM pada sistem ini memiliki peran ganda. Selain menghasilkan respons tekstual dalam bentuk jawaban percakapan, LLM juga bertugas menyusun instruksi atau kueri yang akan diteruskan ke modul SQLAgent. SQLAgent bekerja bersama LLM untuk membangun kueri yang sesuai dengan struktur basis data dan mengeksekusinya secara aman. Hasil eksekusi kueri tersebut kemudian diproses dan direpresentasikan dalam bentuk *DataFrame*, yang menjadi basis untuk seluruh keluaran analitik lanjutan pada sistem.

Setelah *DataFrame* terbentuk, sistem melakukan proses klasifikasi untuk menentukan jalur keluaran yang sesuai. Jika tidak diperlukan pemrosesan lanjutan,

sistem akan langsung menghasilkan respons percakapan dan menyimpannya sebagai bagian dari histori interaksi pengguna. Dengan kata lain, proses langsung ini dinamakan klasifikasi ke dalam variabel `CONVERSATION` yang tidak memerlukan data untuk menghasilkan jawaban. Penjelasan ini akan ditelusuri lebih lengkap pada Subbab 3.5.2 bagian Mekanisme Klasifikasi Fitur.

Selain itu, apabila hasil analisis memerlukan visualisasi, sistem akan mengaktifkan jalur *plot* atau klasifikasi dengan variabel `PLOT`. Pada jalur ini, sebuah modul *Enricher LLM* digunakan untuk menyusun langkah-langkah pembuatan grafik secara terstruktur. Instruksi tersebut kemudian diterjemahkan menjadi kode Python oleh modul *CodeGen*. Kode yang dihasilkan akan dieksekusi dengan mekanisme pemeriksaan *runtime* hingga maksimal tiga kali percobaan. Jika eksekusi berhasil, gambar hasil visualisasi akan disimpan dan dikirimkan kembali kepada pengguna. Jika seluruh percobaan gagal, sistem akan mengembalikan pesan kesalahan yang informatif.

Selain visualisasi, sistem juga mendukung kebutuhan ekspor data ke dalam format Excel. Ketika klasifikasi menandai bahwa hasil analitik perlu diekspor atau yang dikenal dengan variabel bernilai `EXPORT`, *DataFrame* akan dikonversi menjadi berkas Excel menggunakan pustaka seperti *xlsxwriter*. Berkas hasil konversi tersebut kemudian diunggah ke media penyimpanan dan disediakan untuk diunduh oleh pengguna. Seluruh berkas keluaran, baik gambar maupun Excel, dikaitkan dengan sesi pengguna untuk memastikan konsistensi histori dan kemudahan pelacakan.

Alur administrator pada diagram aktivitas terbaru menunjukkan mekanisme yang berbeda pada tahap awal, namun tetap terintegrasi dengan *pipeline chatbot* yang sama. Administrator mengakses *splash screen* dan langsung melakukan *login* tanpa melalui proses *register*. Setelah masuk, administrator memiliki kemampuan untuk melakukan *data intake* melalui dua jalur, yaitu unggahan dokumen yang akan diproses menggunakan OCR atau input data secara manual. Dokumen yang diunggah akan diekstraksi isinya oleh modul OCR dan hasil ekstraksi tersebut disajikan untuk diverifikasi. Administrator dapat melakukan operasi CRUD terhadap data hasil OCR guna memperbaiki kesalahan atau melengkapi informasi yang belum akurat sebelum data disimpan secara permanen.

Di sisi lain, data yang telah diverifikasi oleh administrator kemudian tersedia sebagai sumber data utama bagi modul SQLAgent dan juga sebagai *knowledge base* untuk layanan *chatbot*. Dengan demikian, ketika administrator maupun pengguna melakukan interaksi analitik melalui *chatbot*, seluruh kueri dan visualisasi

yang dihasilkan merujuk pada data yang telah tervalidasi. Administrator juga dapat mengakses menu *chatbot* untuk melakukan analisis lanjutan menggunakan mekanisme yang identik dengan alur pengguna biasa.

Secara keseluruhan, diagram aktivitas versi terbaru menekankan pemisahan tanggung jawab antara pengguna dan administrator, kontrol alur yang jelas melalui titik-titik keputusan seperti penentuan percakapan, visualisasi, dan ekspor data, serta penanganan kesalahan yang terstruktur. Pendekatan ini memastikan bahwa sistem tidak hanya fleksibel dalam mendukung berbagai bentuk keluaran analitik, tetapi juga aman, dapat diaudit, dan konsisten antara sumber data dan hasil yang disajikan kepada pengguna.

3.5.2 Implementasi Perancangan

A Kebutuhan Antarmuka Pengguna

1) Halaman dan Komponen yang Diperlukan

Pengembangan antarmuka pengguna (UI) pada Platform AI Reporting meliputi perancangan berbagai halaman dan komponen utama yang bertujuan untuk memberikan pengalaman pengguna yang intuitif dan efisien, serta mendukung seluruh fungsionalitas sistem. Setiap menu dirancang dengan memperhatikan kemudahan navigasi, konsistensi tampilan, serta keterpaduan dengan fitur-fitur utama berbasis kecerdasan buatan. Adapun penjelasan lebih lanjut mengenai menu-menu inti yang dikembangkan akan diuraikan sebagai berikut.

- **Menu Landing Page / Splash Screen**
  Menu ini merupakan halaman utama Platform AI Reporting yang menampilkan *highlight* fitur-fitur unggulan dan visual untuk mengarahkan pengguna melakukan *login* atau *register*. Desain antarmuka fokus pada visual yang menarik dan navigasi yang jelas agar pengguna memahami fungsi utama platform dengan cepat.

- **Menu Login**
  Menu ini menyediakan formulir *login* bagi pengguna terdaftar untuk mengakses halaman *dashboard*. Sistem akan melakukan proses autentikasi, dan apabila kredensial yang dimasukkan valid, pengguna akan diarahkan ke halaman *dashboard*. Formulir ini dirancang dengan tampilan yang responsif serta mudah digunakan pada berbagai jenis perangkat.

- **Menu Register**

  Menu ini memungkinkan pengguna baru untuk membuat akun di platform. Form registrasi dilengkapi dengan validasi kekuatan password dan pemeriksaan kesesuaian data, sehingga memastikan keamanan akun pengguna. Setelah pendaftaran berhasil, pengguna dapat langsung mengakses *dashboard*.

- **Menu Dashboard**

  Menu ini berperan sebagai pusat kendali pengguna setelah proses *login*. Pada halaman *dashboard*, pengguna dapat melihat daftar *room*, informasi statistik aktivitas mingguan, serta mengakses fitur *chatbot*. Selain itu, tersedia tombol *sign-out* yang memungkinkan pengguna keluar dari akun secara aman. Perancangan antarmuka pada menu ini bertujuan untuk memudahkan pengguna dalam memantau serta mengelola interaksi mereka dengan platform.

- **Menu Chatbot**

  Menu ini menghadirkan pengalaman interaktif menggunakan *chatbot* berbasis AI. Pengguna dapat mengajukan pertanyaan dan mendapatkan respons cerdas secara *real-time*. Tampilan difokuskan pada percakapan yang bersih, nyaman, dan mudah diikuti.

- **Menu Admin Login**

  Menu ini menyediakan form *login* khusus untuk *administrator* platform. Setelah autentikasi berhasil, admin akan diarahkan ke konsol admin. Sistem dirancang untuk menjaga keamanan akses dan membedakan peran pengguna biasa dengan admin.

- **Menu Admin**

  Menu ini merupakan konsol admin untuk mengelola seluruh aktivitas platform. Fitur yang tersedia meliputi CRUD data, unggah file PDF untuk proses OCR, dan terdapat tombol *sign out* untuk keluar, serta akses ke menu *chatbot*. Antarmuka dirancang efisien agar admin dapat memonitor dan mengelola data secara cepat dan terstruktur.

2) API Endpoint

Pada tahap awal perancangan sistem, seluruh interaksi dengan layanan *backend* diuji menggunakan *Postman* guna memastikan setiap fungsi API berjalan

sesuai dengan kebutuhan sebelum dilakukan integrasi dengan antarmuka *frontend*. Pendekatan ini bertujuan untuk mempermudah proses verifikasi serta *debugging* terhadap setiap *endpoint* yang telah dirancang. Daftar API *endpoint* tersebut memiliki peran krusial dalam mendukung pengembangan *frontend* pada Platform AI Reporting, khususnya dalam menjamin komunikasi data yang efektif antara server dan antarmuka pengguna. Setiap *endpoint* dikembangkan agar *frontend* dapat mengakses data secara langsung dari server, termasuk layanan *chatbot* dengan pengelolaan *memory*, serta pengawasan proses *login* dan *register*. Oleh karena itu, perincian API *endpoint* beserta penjelasannya akan dipaparkan pada bagian berikut.

1. **Authentication & User Management**

   - `POST /api/auth/login`
     API *Endpoint* ini digunakan untuk login pengguna biasa dengan menggunakan password demo. Jika autentikasi berhasil, sistem akan mengembalikan *JSON Web Token* (JWT) yang digunakan untuk mengakses endpoint lain yang memerlukan autentikasi.
   - `POST /api/admin/login`
     API *Endpoint* ini digunakan untuk login administrator menggunakan kredensial yang tersimpan di environment. Admin yang berhasil login akan menerima token untuk mengakses konsol admin dan fitur terkait.
   - `POST /api/create_user`
     API *Endpoint* ini berfungsi untuk membuat akun pengguna baru. Terdapat validasi untuk mencegah pembuatan akun dengan username `admin`.

2. **Rooms, Chat, & Analytics**

   - `POST /api/new_room`
     API *Endpoint* ini digunakan untuk membuat room chat baru. Akses dibatasi menggunakan Bearer token untuk memastikan keamanan.
   - `GET /api/<user_id>/rooms`
     API *Endpoint* ini menampilkan daftar room milik pengguna tertentu. Diperlukan token dan sistem melakukan validasi kecocokan user.
   - `GET /api/analytics/weekly_activity`
     API *Endpoint* ini mengembalikan statistik aktivitas pengguna selama 7 hari terakhir. Akses membutuhkan token yang valid.

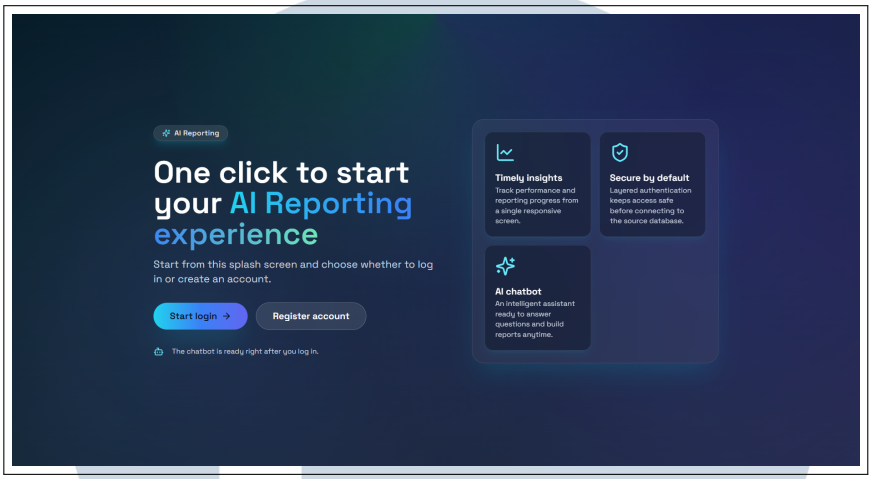
- `POST /api/rooms/<room_id>/messages/bot`
  API *Endpoint* ini digunakan untuk mengirim prompt ke bot dalam sebuah room. Bot dapat menghasilkan teks, gambar, atau file Excel. Akses dibatasi untuk pemilik room dengan token aktif.
- `GET /api/rooms/<room_id>/messages`
  API *Endpoint* ini mengambil histori pesan dari room tertentu. Diperlukan token dan validasi kepemilikan room.

3. **Admin Transactions**
   - `GET /api/admin/transactions` dan `POST /api/admin/transactions`
     API *Endpoint* ini digunakan untuk melihat daftar transaksi atau membuat transaksi baru. Akses hanya untuk admin dengan token yang valid.
   - `PUT /api/admin/transactions/<trans_id>` dan `DELETE /api/admin/transactions/<trans_id>`
     API *Endpoint* ini digunakan untuk memperbarui atau menghapus transaksi tertentu. Akses dibatasi untuk admin dengan token.
   - `POST /api/admin/transactions/upload`
     API *Endpoint* ini memungkinkan admin mengunggah file PDF, kemudian sistem melakukan OCR dan ekstraksi dengan LLM untuk menyimpan atau memperbarui data transaksi. Token admin diperlukan.

3) Tampilan User Interface

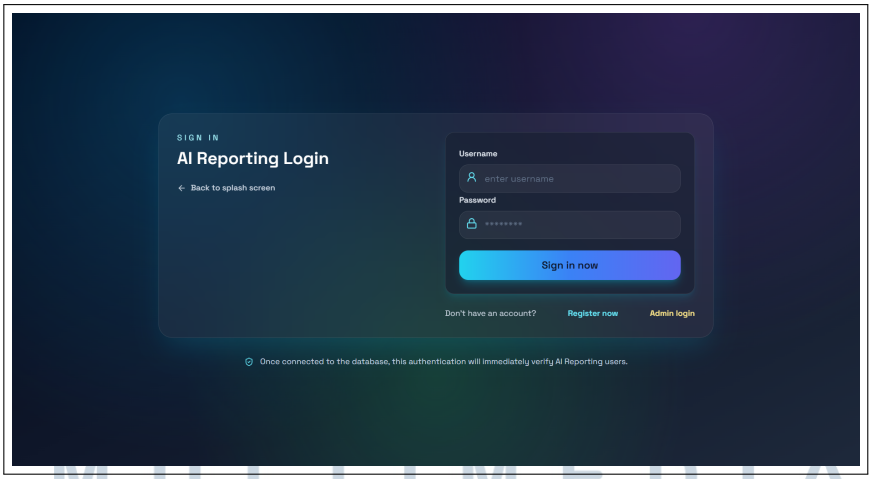
Tampilan antarmuka pengguna atau *user interface* pada Platform AI Reporting dirancang berdasarkan kebutuhan halaman dan komponen yang telah dijabarkan pada subbab sebelumnya. Desain antarmuka mengedepankan prinsip keterbacaan, kemudahan penggunaan, dan responsivitas agar dapat menyesuaikan diri pada berbagai ukuran layar perangkat. Elemen-elemen UI dipilih untuk memberikan pengalaman yang ramah pengguna (user-friendly) sekaligus mempertahankan konsistensi visual dan hierarki informasi yang jelas.Selain itu, tahap konsepsi awal dilakukan melalui perancangan visual menggunakan perangkat lunak Adobe Photoshop, kemudian dilanjutkan dengan penyajian *prototype* secara langsung saat pemaparan desain keseluruhan untuk memudahkan evaluasi

fungsionalitas dan alur interaksi. Adapun tampilan antarmuka yang dihasilkan adalah sebagai berikut.



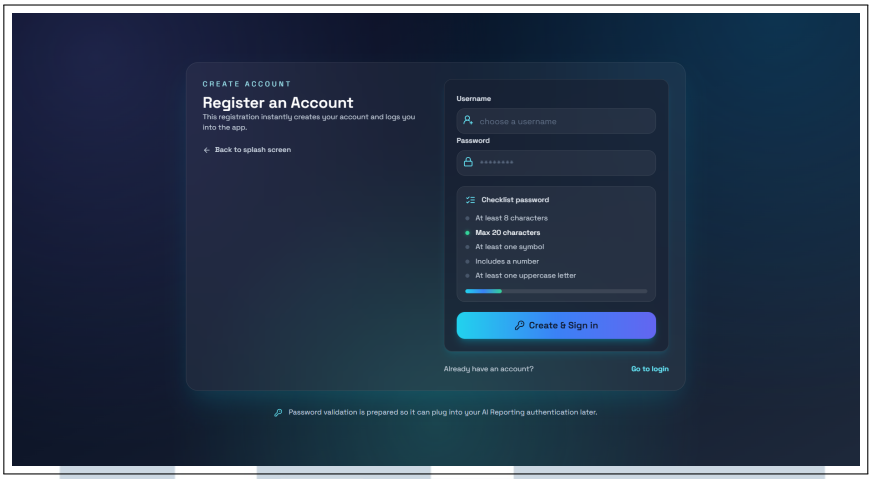
Gambar 3.9. Tampilan halaman *home / splash screen* pada Platform AI Reporting

Gambar 3.9 menunjukkan tampilan halaman utama (home) pada Platform AI Reporting yang berfungsi sebagai titik masuk pengguna, menampilkan ringkasan fitur unggulan, dan akses ke halaman *login* atau *register*. Desain antarmuka menekankan navigasi yang jelas dan informasi penting yang mudah diakses sehingga pengguna dapat langsung memilih alur kerja yang diinginkan.



Gambar 3.10. Tampilan halaman *login user* pada Platform AI Reporting

Gambar 3.10 menunjukkan tampilan halaman *login* pengguna yang ringkas dan aman, dilengkapi form untuk mengisi username dan password, dan navigasi ke halaman *register* apabila belum memiliki akun yang terdaftar, serta tombol *login* untuk admin.

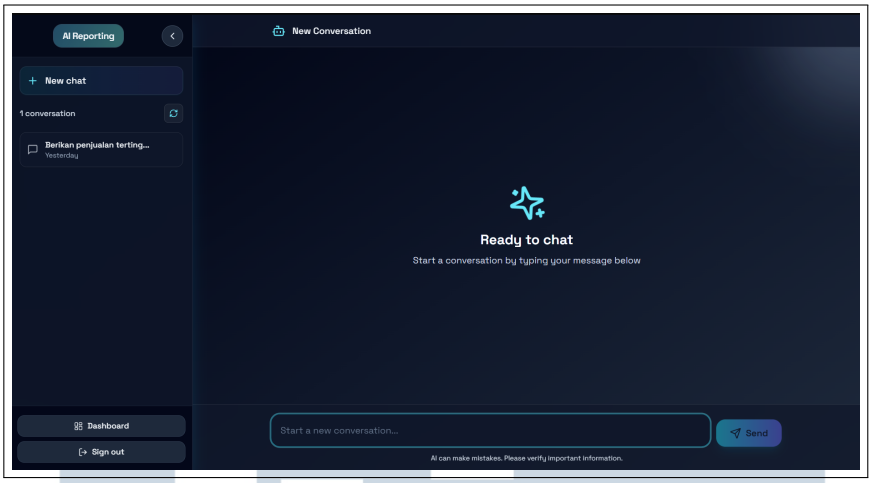


Gambar 3.11. Tampilan halaman *register user* pada Platform AI Reporting

Gambar 3.11 menunjukkan antarmuka pendaftaran akun baru bagi pengguna, menampilkan field dasar seperti username dan password, serta tombol konfirmasi pendaftaran, serta petunjuk singkat tentang persyaratan keamanan kata sandi dan verifikasi email untuk memastikan akun siap digunakan.

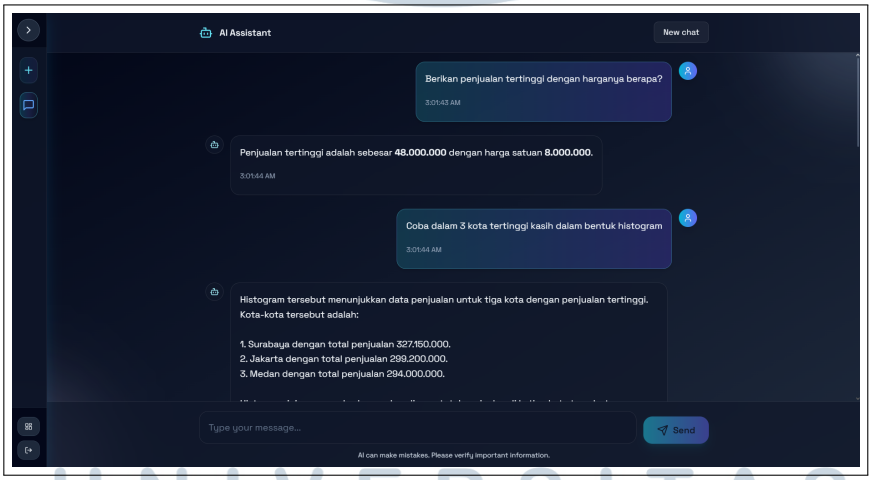
Gambar 3.12. Tampilan halaman *dashboard user* pada Platform AI Reporting

Gambar 3.12 menunjukkan tampilan *dashboard* utama setelah *login*, berisi ringkasan metrik aktivitas mingguan, akses cepat ke menu *chatbot*, daftar histori percakapan, dan log aktivitas saat mengakses Platform AI Reporting, sehingga pengguna mendapat gambaran kondisi data dan aktivitas sistem dalam satu layar.



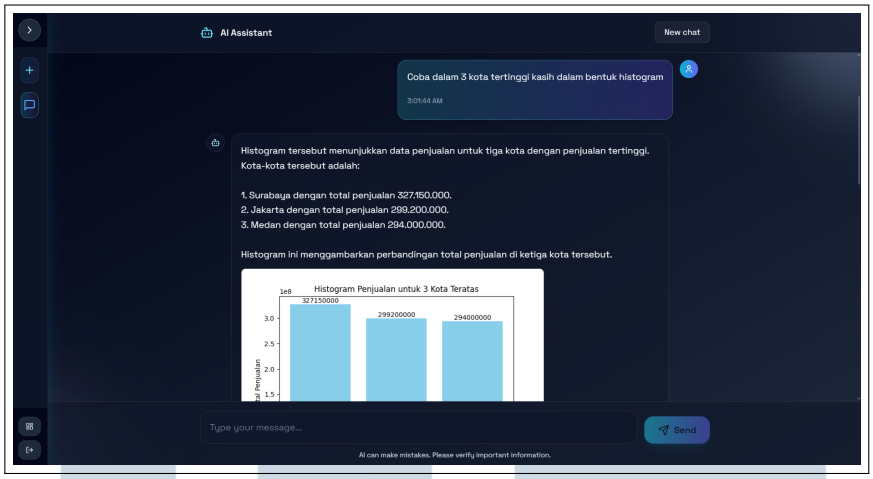
Gambar 3.13. Tampilan halaman *chatbot* pada Platform AI Reporting

Gambar 3.13 menunjukkan antarmuka *chatbot* yang memungkinkan pengguna mengajukan pertanyaan berbasis *natural language*, melihat respon yang dihasilkan LLM, yaitu teks, grafik, atau file Excel, serta menelusuri histori percakapan mirip pengalaman ChatGPT untuk menjaga konteks dan riwayat interaksi.

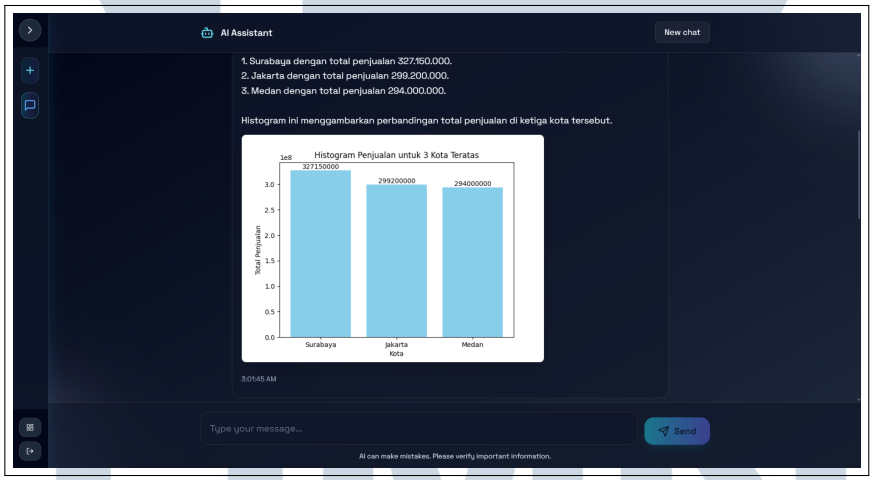


Gambar 3.14. Tampilan halaman *chatbot* dengan fitur *AI Conversational Analytics* pada Platform AI Reporting

Gambar 3.14 menunjukkan fitur *AI Conversational Analytics* yang memperlihatkan bagaimana *prompt* pengguna digabungkan dengan *system prompt* untuk menghasilkan kueri SQL via LLM. Antarmuka menampilkan area input, *preview query* yang dihasilkan, dan hasil respons *chatbot*.

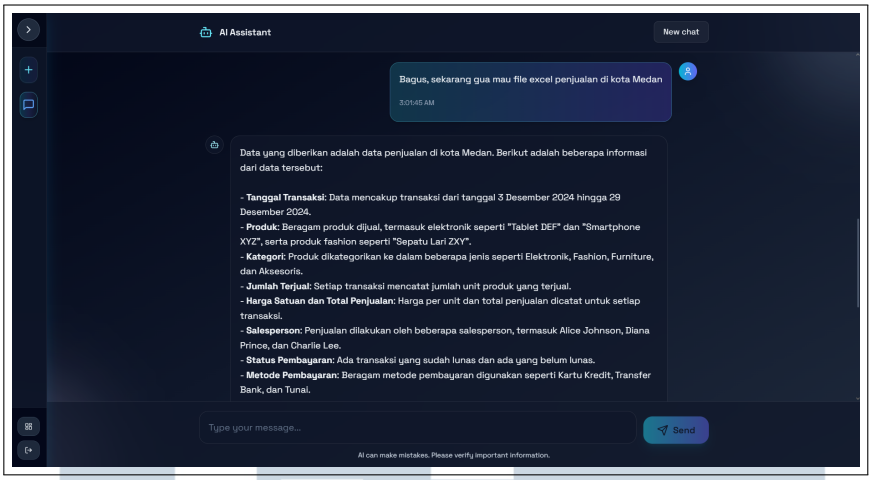


Gambar 3.15. Tampilan halaman *chatbot* dengan fitur *Automated Data Visualization* pada Platform AI Reporting (1)

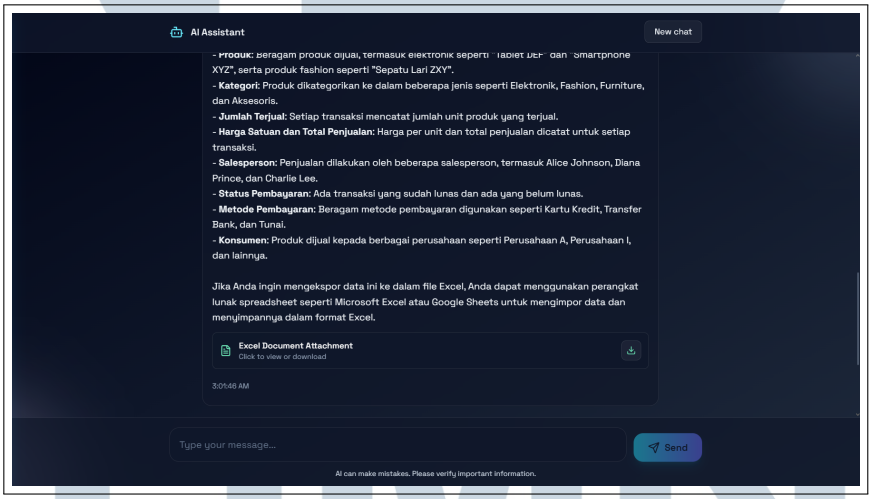


Gambar 3.16. Tampilan halaman *chatbot* dengan fitur *Automated Data Visualization* pada Platform AI Reporting (2)

Gambar 3.15 dan 3.16 secara bersama menunjukkan fitur *Automated Data Visualization*, dengan Gambar 3.15 memperlihatkan *prompt* user yang bertanya untuk memberikan jawaban dalam bentuk grafik kepada *chatbot*, sedangkan Gambar 3.16 memperlihatkan hasil plot yang dihasilkan agen *plotting* otomatis sebagai jawaban dari hasil *prompt* user. Keseluruhan alur memungkinkan pengguna mendapatkan grafik yang relevan langsung dari pertanyaan pengguna.

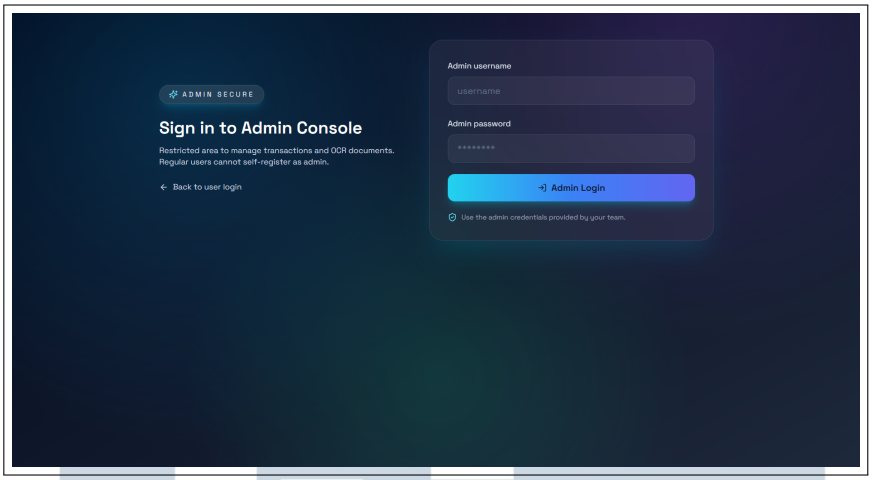


Gambar 3.17. Tampilan halaman *chatbot* dengan fitur *Dynamic Excel Export* pada Platform AI Reporting (1)



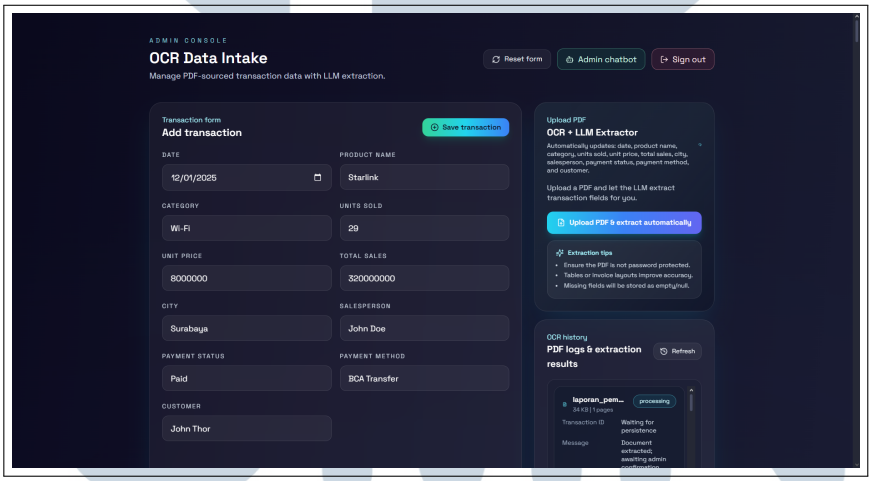
Gambar 3.18. Tampilan halaman *chatbot* dengan fitur *Dynamic Excel Export* pada Platform AI Reporting (2)

Gambar 3.17 dan 3.18 bersama-sama menjelaskan fitur *Dynamic Excel Export*, dengan Gambar 3.17 menampilkan *prompt* user yang bertanya untuk memberikan jawaban dalam bentuk file Excel kepada *chatbot*, sedangkan Gambar 3.18 memperlihatkan tampilan file Excel yang dihasilkan berdasarkan hasil *prompt* pengguna dan opsi unduh. Fitur ini mempermudah pengguna mendapatkan *deliverable data* yang siap untuk dipakai tanpa pengerjaan manual.



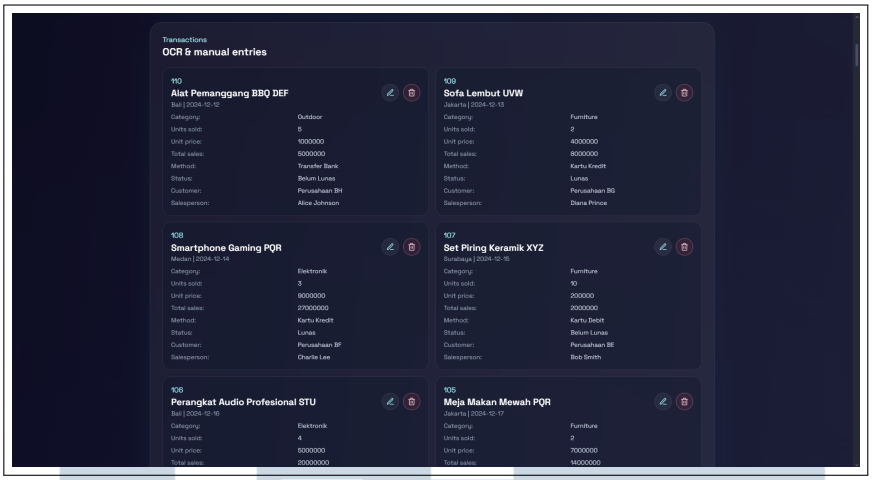
Gambar 3.19. Tampilan halaman *login admin* pada Platform AI Reporting

Gambar 3.19 menunjukkan halaman *login* khusus admin yang memisahkan jalur autentikasi administrator dari pengguna biasa, menyediakan akses ke fungsi manajemen data, pengaturan OCR, dan monitoring data.



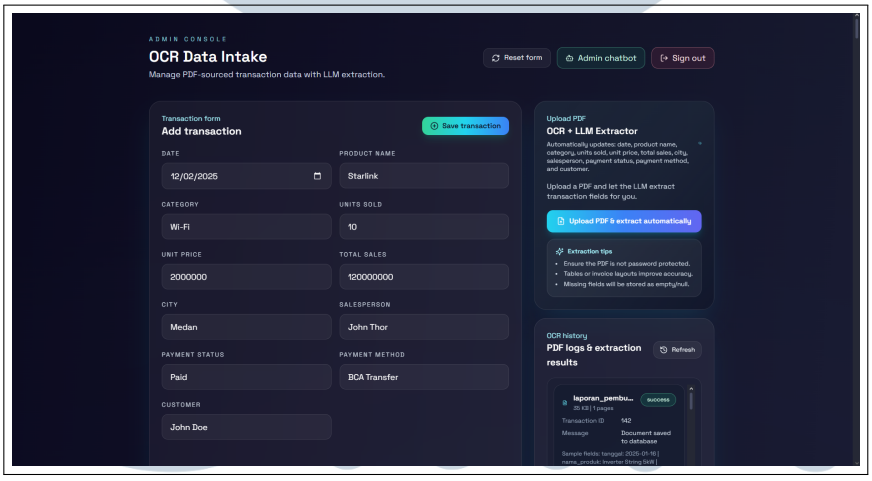
Gambar 3.20. Tampilan halaman *dashboard admin* pada Platform AI Reporting

Gambar 3.20 menunjukkan panel menu admin yang menampilkan navigasi untuk manajemen data, kontrol OCR dalam bentuk penggungahan dokumen PDF, *review hasil ekstraksi* dengan histori ekstraksinya, aksi untuk mengubah data, menghapus data, dan akses ke menu *chatbot*.



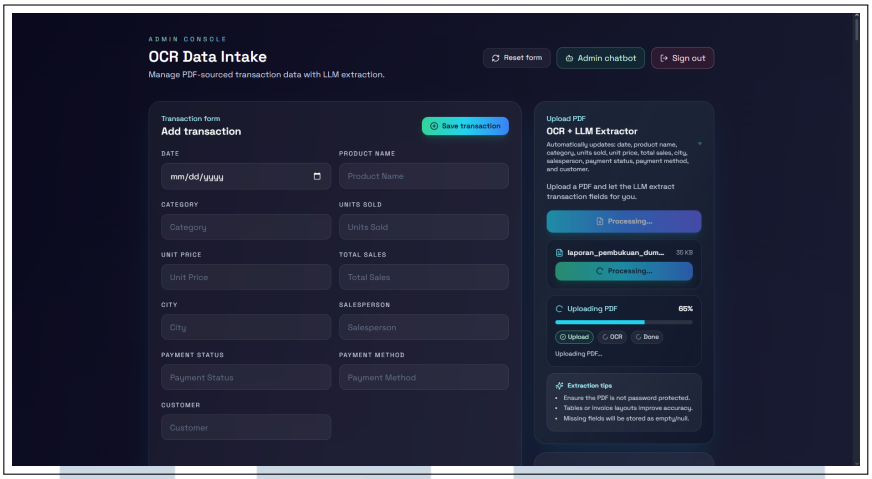
Gambar 3.21. Tampilan halaman *dashboard admin* berisikan data pada Platform AI Reporting

Gambar 3.21 menunjukkan tampilan data yang menampilkan data yang tersimpan, baik dari hasil inputan manual admin atau melalui proses pengunggahan dokumen PDF.



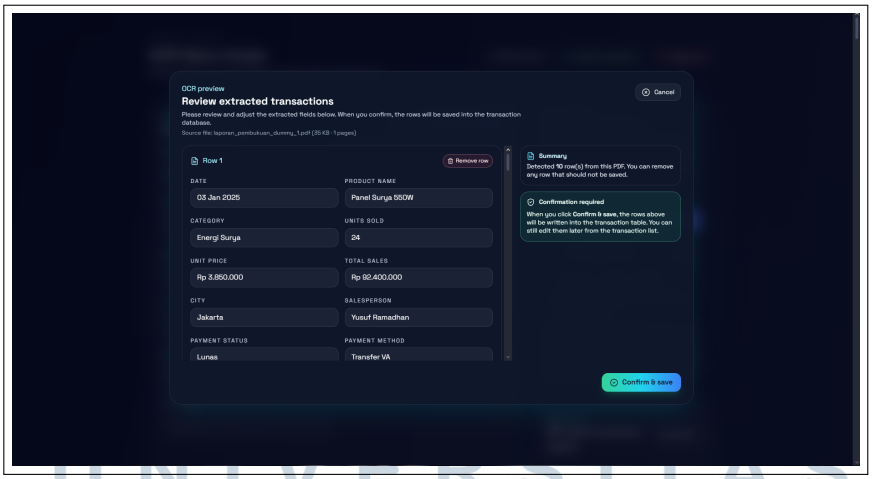
Gambar 3.22. Tampilan halaman *dashboard admin* saat melakukan inputan manual pada Platform AI Reporting

Gambar 3.43 menunjukkan antarmuka input data manual yang digunakan oleh admin untuk memasukkan atau memperbaiki data secara langsung (CRUD), menyediakan field yang terstruktur, validasi input, serta tombol simpan dan batal untuk memastikan akurasi data sebelum disimpan ke basis data.



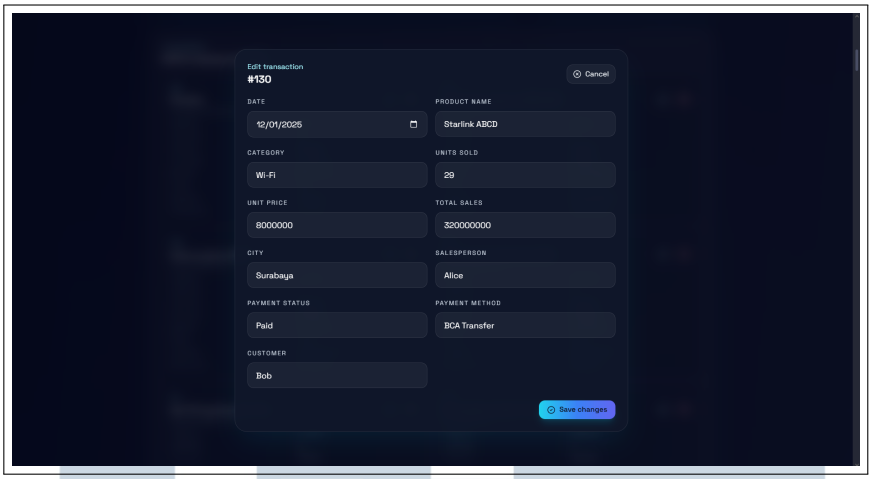
Gambar 3.23. Tampilan halaman *dashboard admin* saat melakukan proses OCR pada Platform AI Reporting

Gambar 3.45 menunjukkan alur pemrosesan OCR dengan admin mengunggah file PDF, sistem mengekstrak isi dokumen tersebut, menyimpan berkas asli ke Cloudinary, dan hasil ekstraksi ke *database*. Antarmuka juga menampilkan status pemrosesan dan ringkasan hasil ekstraksi untuk verifikasi manual jika diperlukan.



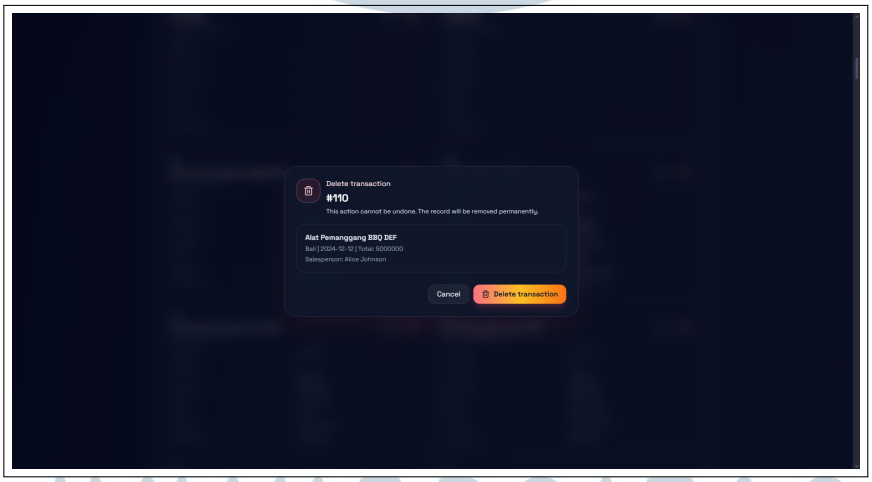
Gambar 3.24. Tampilan halaman *dashboard admin* saat melakukan konfirmasi hasil OCR pada Platform AI Reporting

Gambar 3.46 menunjukkan tampilan konfirmasi hasil ekstrak dari pemrosesan OCR. Dengan adanya tampilan ini memungkinkan admin untuk mengubah data apabila tidak sesuai dengan pemrosesan OCR. Apabila seluruh data hasil ekstraksi OCR telah selesai, maka data tersebut akan dimasukkan ke basis data `ai_reporting_data`.



Gambar 3.25. Tampilan halaman *dashboard admin* saat mengubah data pada Platform AI Reporting

Gambar 3.51 menunjukkan tampilan untuk mengubah data yang memungkinkan admin mengoreksi hasil ekstraksi OCR atau memperbarui data secara manual, menampilkan field yang dapat diedit, dan tombol simpan untuk menerapkan perubahan ke basis data.



Gambar 3.26. Tampilan halaman *dashboard admin* saat menghapus data pada Platform AI Reporting

Gambar 3.26 menunjukkan mekanisme penghapusan data yang disertai dialog konfirmasi untuk mencegah penghapusan tidak sengaja.

B Mekanisme Klasifikasi Fitur

Secara arsitektural, keempat fitur pada Platform AI Reporting dibangun di atas satu mekanisme klasifikasi percakapan yang terpusat di `ConversationNode`. Setiap kali pengguna mengirimkan *prompt*, node ini memanggil LLM dengan konteks berupa pertanyaan terbaru, riwayat percakapan, dan skema database, lalu meminta model untuk mengembalikan objek JSON dengan beberapa field kunci, yaitu `next_agent`, `data_action`, `response_status`, dan `export_csv`. Field `next_agent` mengklasifikasikan jenis tugas menjadi tiga kategori utama, termasuk `"CONVERSATION"` untuk percakapan umum, `"FETCH"` untuk permintaan analitik data untuk fitur *AI Conversational Analytics*, dan `"EXPORT"` untuk fitur *Dynamic Excel Export* ketika pengguna secara eksplisit meminta ekspor data. Field `data_action` menentukan apakah jawaban data sebaiknya disajikan sebagai teks saja (`"NO_PLOT"`) atau perlu divisualisasikan (`"PLOT"`) sebagai representasi dari fitur *Automated Data Visualization*. Sementara itu, `export_csv` adalah nilai boolean yang menandakan apakah hasil data perlu diekspor ke Excel. Dengan kombinasi tiga sinyal ini, sistem *backend* dapat memutuskan apakah percakapan cukup dijawab di tingkat `ConversationNode` tanpa menyentuh SQLAgent sama sekali seperti kasus `next_agent="CONVERSATION"`, atau perlu mengaktifkan SQLAgent untuk menjalankan *query* ke *database* seperti kasus `"FETCH"` maupun `"EXPORT"`. Dengan kata lain, percakapan dalam sistem ini bisa murni percakapan tanpa sentuhan SQL, maupun percakapan yang melibatkan SQLAgent untuk mengambil data aktual, tergantung klasifikasi yang dihasilkan oleh LLM dan *prompt* user.

```
class ConversationNode(LLMNode):
    def call(self, state) -> Command:
        # Mengambil prompt pengguna, room id, dan user id dari state
        user_prompt = state["user_prompt"]
        room_id = state["room_id"]
        user_id = state["user_id"]
        # Mengambil history percakapan relevan dari DB
        messages = self.chatManager.get_relevant_messages(
            user_id=user_id,
            room_id=room_id,
        )
        # Mengambil jenis agen berikutnya (CONVERSATION/FETCH/EXPORT)
        next_agent = json_response["next_agent"]
        # Mengambil ringkasan jawaban teks dari LLM
```

```
        response_text = json_response["response"]
        # Mengambil aksi data (PLOT atau NO_PLOT)
        data_action = json_response["data_action"]
        # Mengambil status validitas pertanyaan
        response_status = json_response["response_status"]
        # Mengetahui apakah user ingin ekspor Excel
        export_csv = json_response["export_csv"]
        # Jika klasifikasi sebagai percakapan biasa
        if next_agent == "CONVERSATION":
            # Jawab langsung dengan teks tanpa agen SQL
            return Command(
                # Akhiri graf di ConversationNode
                goto=END,
                # Mengisi jawaban akhir ke summarized_output
                update={"summarized_output": response_text},
            )
        # Jika next_agent adalah FETCH atau EXPORT, lanjutkan ke agen SQL
        return Command(
            # Arahkan alur ke node DB_QUERY (Retriever)
            goto="DB_QUERY",
            update={
                # Menyimpan tujuan analitik sebagai goal untuk agen SQL
                "goal": response_text,
                # Menyimpan data_action untuk routing plot atau tidak
                "router_result": data_action,
                # Menyimpan flag ekspor Excel untuk tahap berikutnya
                "export_csv": export_csv,
                # Menyimpan kembali pertanyaan user ke state
                "user_question": user_prompt,
                # Menyimpan history untuk agen-agen berikutnya
                "history_conversation": formatted_messages,
            },
        )
```

Kode 3.1: Klasifikasi next_agent

Kode 3.1 memperlihatkan bahwa `ConversationNode` bukan sekadar menjawab, tetapi juga berperan sebagai *router* yang memutuskan apakah percakapan berhenti di sini sebagai `CONVERSATION`, atau dilanjutkan ke SQLAgent untuk `FETCH` dan `EXPORT`. Dengan demikian, percakapan bisa sepenuhnya

diselesaikan tanpa menyentuh basis data seperti jalur murni percakapan pada umumnya, atau sebaliknya memanfaatkan SQLAgent untuk mengambil dan mengolah data sebelum jawaban akhir dibentuk.

Untuk lebih jelasnya, arsitektur sistem beroperasi melalui alur pipa pemrosesan bertahap yang dimulai ketika *prompt* pengguna, riwayat percakapan, dan konteks basis data dianalisis oleh LLM *router* untuk mengklasifikasikan jenis respons yang diperlukan, baik sebagai respons percakapan langsung maupun kueri berbasis data, sekaligus menentukan kebutuhan visualisasi (grafik) dan ekspor ke format Excel. Apabila klasifikasi menunjukkan cukup dengan respons percakapan, LLM *router* menghasilkan jawaban yang langsung dikirimkan kepada pengguna. Sebaliknya, jika memerlukan pengambilan data, tujuan pemrosesan tersebut diteruskan ke LLM spesialis SQL yang menyusun *query* yang aman, mengeksekusinya pada basis data, dan mengumpulkan hasilnya dalam bentuk tabel. Tabel hasil kueri ini kemudian diproses oleh LLM *plotting* untuk menghasilkan kode Python yang memvisualisasikan data dalam bentuk grafik, sementara pada tahap akhir, LLM *summarizer* mengintegrasikan tabel data, grafik, dan intensi pengguna untuk menyusun respons komprehensif dalam bahasa Indonesia. Secara paralel, sistem mengonversi tabel menjadi berkas Excel apabila diminta, mengunggah aset visual dan berkas ke penyimpanan, kemudian mengembalikan ke *frontend* respons tekstual final beserta tautan ke grafik dan berkas Excel yang telah dihasilkan. LLM disini berperan sebagai *system prompt*.

C Fitur AI Conversational Analytics

1) Penjelasan Alur Kerja Fitur

Fitur *AI Conversational Analytics* merupakan inti dari mekanisme percakapan cerdas pada Platform AI Reporting. Fitur ini tidak hanya menangani percakapan biasa (`next_agent = "CONVERSATION"`), tetapi juga berperan sebagai *router* yang memutuskan kapan sebuah pesan harus diteruskan ke agen analitik data (`next_agent = "FETCH"` atau `"EXPORT"`). Keputusan ini diambil oleh komponen `ConversationNode` dengan mempertimbangkan tiga sinyal utama yang dikembalikan oleh *Large Language Model* (LLM), yaitu `next_agent`, `data_action`, dan `export_csv`. Sinyal `next_agent` mengklasifikasikan jenis tugas menjadi percakapan murni atau analitik data. `data_action` menentukan apakah jawaban analitik perlu divisualisasikan (`"PLOT"`) atau cukup dijelaskan dengan teks (`"NO_PLOT"`). Sedangkan `export_csv` menandakan apakah pengguna meminta

hasil data diekspor ke dalam berkas Excel (`True`) atau tidak (`False`). Dengan kombinasi ketiga sinyal ini, `ConversationNode` dapat memutuskan apakah *chatbot* cukup menjawab di tingkat percakapan saja, atau perlu mengaktifkan SQLAgent untuk mengambil data dari basis data sebagai representasi fitur *AI Conversational Analytics* dan lainnya.

Pada sisi agen, keputusan apakah percakapan akan berhenti sebagai obrolan murni atau diteruskan ke agen SQL ditentukan oleh nilai `next_agent` yang dikembalikan oleh LLM pada *ConversationNode*. Potongan Kode 3.2 memperlihatkan bagaimana JSON hasil LLM diparsing untuk mengambil `next_agent` dan bagaimana graf diarahkan ke jalur `"CONVERSATION"` atau `"FETCH"`.

```
# Memanggil LLM router untuk mendapatkan JSON keputusan
response = self.invoke_on_messages(
    {
        "user_question": query,
        "history_conversation": formatted_messages,
        "db_schema": self._db_schema,
    },
    split_content,
)
# Nilai-nilai kontrol yang dikembalikan LLM ConversationNode
# "CONVERSATION" atau "FETCH"
next_agent = json_response["next_agent"]
# Ringkasan intent / jawaban
response = json_response["response"]
# "PLOT" atau "NO_PLOT"
data_action = json_response["data_action"]
# True jika user ingin Excel
export_csv = json_response["export_csv"]
# Jika next_agent = "CONVERSATION", graf berhenti di sini dan
    response langsung menjadi summarized_output tanpa query SQL
if next_agent == "CONVERSATION":
    return Command(
        goto=END,
        update={
            "summarized_output": response,
        },
    )
# Jika next_agent = "FETCH", alur dialihkan ke agen SQL (DB_QUERY)
    dengan membawa goal, schema yang sudah di-split, dan flag-flag
    kontrol
```

```
return Command(
    goto="DB_QUERY",
    update={
        "goal": response,
        "chunked_schema": sql_schema,
        "router_result": data_action,
        "export_csv": export_csv,
        "user_question": query,
        "history_conversation": formatted_messages,
    },
)
```

Kode 3.2: Penentuan variabel next_agent

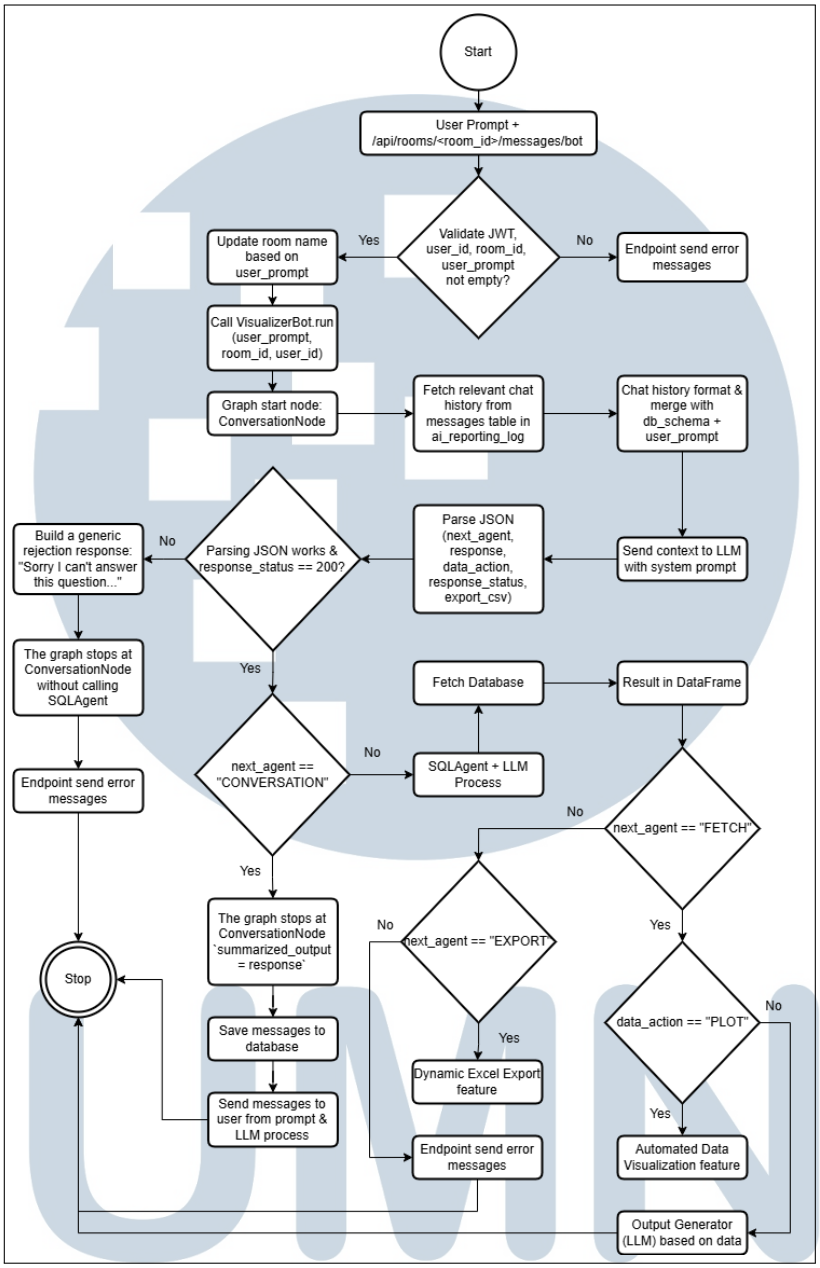
Untuk jalur teks murni, baik ketika `next_agent = "CONVERSATION"` (jawaban hanya percakapan) maupun ketika `next_agent = "FETCH"` tetapi `router_result = "NO_PLOT"` dan `export_csv = False` (Analitik data tanpa gambar dan tanpa Excel), *backend* hanya menyimpan pesan pengguna dan jawaban bot ke tabel `messages` dan mengirimkan respons teks ke klien tanpa URL gambar maupun berkas Excel. Potongan Kode 3.3 memperlihatkan jalur teks-saja tersebut pada *backend*.

```
# Menjalankan VisualizerBot; di dalamnya sudah ditentukan
    next_agent, router_result, export_csv, dan summarized_output
response = bot.run(user_prompt, numeric_room_id, user_id)
# Mengambil jawaban teks akhir dari VisualizerBot
resp_content = response.get("summarized_output")
# Inisialisasi link gambar dan Excel sebagai None
image_link = None
xlsx_link = None
# Membuat objek pesan user untuk disimpan ke DB
new_message = SQLChatHistoryManager.Message(
    text=user_prompt,
    timestamp=None,
    user_id=user_id,
    room_id=numeric_room_id,
    role="user",
    data_type="text",
)
chat_manager.enter_message(new_message)
# Membuat objek pesan bot untuk jawaban AI
bot_message = SQLChatHistoryManager.Message(
    text=resp_content,
    timestamp=None,
```

```
    # Tidak ada gambar pada jalur teks murni
    image_url=image_link,
    document_url=xlsx_link,
    # Tidak ada dokumen/Excel
    excel_url=xlsx_link,
    user_id=user_id,
    room_id=numeric_room_id,
    role="bot",
    data_type="text",
    df=None,
    # Tidak menyimpan DataFrame pada jalur ini
)
chat_manager.enter_message(bot_message)
# Mengirimkan respon HTTP ke klien
return jsonify(
    {
        "status_code": "BOT-000",
        "data": {
            "text": resp_content,  # Jawaban teks ke klien
            "document": xlsx_link, # None: tidak ada dokumen
            "excel": xlsx_link,    # None: tidak ada Excel
            "image": image_link,   # None: tidak ada gambar
        },
        "message": "Bot run success",
    }
), 200
```

Kode 3.3: Bentuk pengiriman respons ke klien

Secara ringkas, alur proses pada fitur *AI Conversational Analytics* dapat divisualisasikan melalui *flowchart* pada Gambar 3.27.



Gambar 3.27. *Flowchart* dari fitur *AI Conversational Analytics*

2) Hasil Pengujian dan Output Fitur

Pengujian pada fitur *AI Conversational Analytics* berfokus pada kemampuan sistem untuk mengklasifikasikan permintaan pengguna, baik berupa murni percakapan atau memerlukan akses data, memanfaatkan riwayat percakapan untuk *follow-up questions*, tampilan antarmuka, pesan *error*, riwayat percakapan tersimpan ke basis data, dan menghasilkan respons yang relevan tanpa atau dengan eksekusi SQL sesuai klasifikasi LLM. Pada Tabel 3.12 menyajikan uraian

pengujian, hasil yang diharapkan, hasil pengujian yang diperoleh, serta kesimpulan secara lebih rinci dan komprehensif.

Tabel 3.12. Hasil pengujian fitur *AI Conversational Analytics*

No.	Pengujian	Skenario	Hasil Pengujian	Kesimpulan
1	Routing klasifikasi `next_agent` ke `"CONVERSATION"`	Pengguna dapat melakukan percakapan murni seperti pertanyaan umum non-analitik	LLM mengembalikan JSON sebagai jawaban dan dikembalikan tanpa *query* SQL untuk pengguna	Sesuai dan Berhasil
2	Routing klasifikasi `next_agent` ke `"FETCH"`	Pengguna dapat melakukan percakapan berbasis data seperti pertanyaan analitik	LLM mengembalikan JSON sebagai jawaban dan dikembalikan dengan *query* SQL untuk pengguna	Sesuai dan Berhasil
3	Memeriksa kemampuan *memory management* untuk mekanisme *follow-up questions*	Pengguna dapat melakukan *follow-up questions*	Pengguna dapat memberikan pertanyaan lanjutan	Sesuai dan Berhasil
4	Memastikan ketersediaan *error messages*	Pengguna dapat memperoleh informasi terkait *error messages*	Tampilan beserta informasi *error messages* ditampilkan dengan jelas	Sesuai dan Berhasil

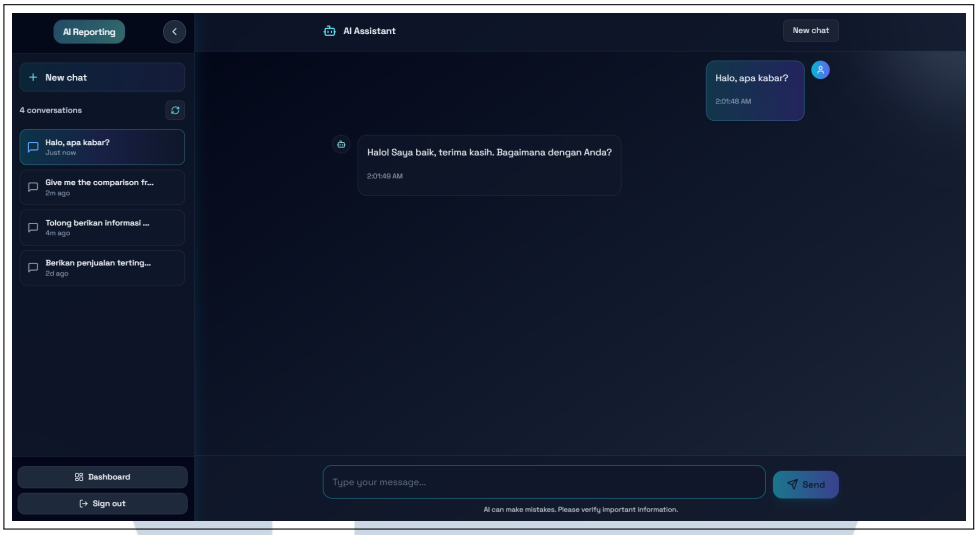
Lanjut pada halaman berikutnya

Lanjutan Tabel 3.12

No.	Pengujian	Skenario	Hasil Pengujian	Kesimpulan
5	Melakukan pengujian konsistensi tampilan responsif	Pengguna dapat mengakses fitur *AI Conversational Analytics* dengan antarmuka yang responsif	Antarmuka menu *AI Conversational Analytics* ditampilkan secara responsif pada seluruh ukuran *device*	Sesuai dan Berhasil
6	Memastikan percakapan tersimpan ke database	Pengguna dapat melihat riwayat percakapan	Percakapan tersimpan ke database `ai_reporting_log`	Sesuai dan Berhasil
7	Memastikan permintaan pengguna sesuai dengan respons *chatbot*	Pengguna dapat bertanya dan jawabannya sesuai harapan	Hasil respons *chatbot* sesuai dengan pertanyaan pengguna	Sesuai dan Berhasil

Adapun contoh *output* yang dapat dilihat pada fitur *AI Conversational Analytics* adalah pada Gambar 3.28 yang memperlihatkan tampilan percakapan sebelum pengaktifan agen SQL, dan Gambar 3.30 yang memperlihatkan tampilan setelah alur `FETCH` diproses melalui pengambilan data di basis data. Pada Gambar 3.28 tertera user bertanya mengenai ”Halo, apa kabar?”, dan *chatbot* merespons ”Halo! Saya baik, terima kasih. Bagaimana dengan Anda?”. Hal ini berarti sistem *backend* telah berhasil mengklasifikasikan `next_agent="CONVERSATION"`, di mana hasil responsnya tidak memerlukan pengambilan data di MySQL. Bukti nyatanya dapat dilihat pada Gambar 3.29 yang berisikan respons memilih variabel `next_agent` bernilai `CONVERSATION`. Sedangkan pada Gambar 3.30 tertera user bertanya mengenai ”Tolong berikan informasi barang dengan penjualan tertinggi di kota Medan”, dan *chatbot* merespons ”Produk dengan penjualan tertinggi di kota Medan adalah ”Smartphone Camera 108MP DEF” dari kategori Elektronik, dengan total penjualan sebesar 30.000.000.”. Hal ini berarti sistem *backend* telah berhasil mengklasifikasikan `next_agent="FETCH"`, di mana hasil responsnya memerlukan pengambilan data di MySQL. Bukti nyatanya dapat dilihat pada Gambar 3.31 yang

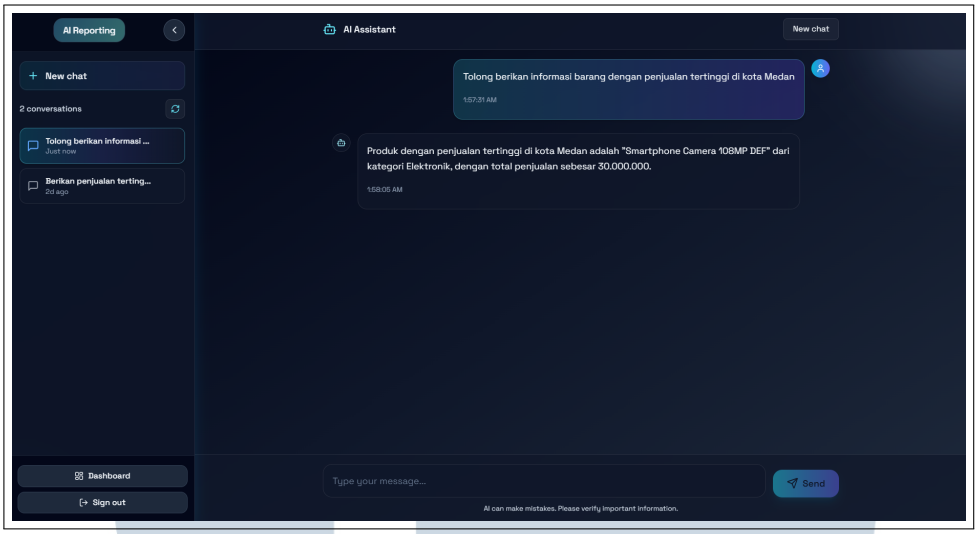
berisikan respons memilih variabel `next_agent` bernilai `FETCH`.



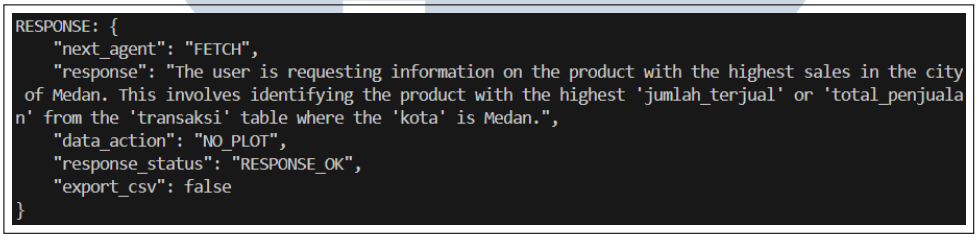
Gambar 3.28. Contoh percakapan pengguna pada fitur *AI Conversational Analytics* dengan `next_agent="CONVERSATION"`

```
RESPONSE: {
    "next_agent": "CONVERSATION",
    "response": "Halo! Saya baik, terima kasih. Bagaimana dengan Anda?",
    "data_action": "NO_PLOT",
    "response_status": "RESPONSE_OK",
    "export_csv": false
}
```

Gambar 3.29. Respons *backend* pada fitur *AI Conversational Analytics* dengan `next_agent="CONVERSATION"`



Gambar 3.30. Contoh percakapan pengguna pada fitur *AI Conversational Analytics* dengan `next_agent="FETCH"`



```
RESPONSE: {
    "next_agent": "FETCH",
    "response": "The user is requesting information on the product with the highest sales in the city
 of Medan. This involves identifying the product with the highest 'jumlah_terjual' or 'total_penjuala
n' from the 'transaksi' table where the 'kota' is Medan.",
    "data_action": "NO_PLOT",
    "response_status": "RESPONSE_OK",
    "export_csv": false
}
```

Gambar 3.31. Respons *backend* pada fitur *AI Conversational Analytics* dengan `next_agent="FETCH"`

D Fitur Automated Data Visualization

1) Penjelasan Alur Kerja Fitur

Fitur *Automated Data Visualization* bertanggung jawab untuk mengubah hasil analitik berbasis SQL menjadi visualisasi grafik yang dapat langsung ditampilkan pada antarmuka pengguna. Berbeda dengan jalur teks murni, fitur ini diaktifkan ketika `ConversationNode` mengklasifikasikan pesan sebagai tugas analitik dengan `next_agent = "FETCH"` dan secara eksplisit menandai bahwa jawaban perlu divisualisasikan melalui `data_action = "PLOT"`. Nilai `data_action` tersebut diteruskan sebagai `router_result` dan diproses di dalam `VisualizerBot` dan mengalirkan state ke simpul `DB_QUERY`, seperti yang ditunjukkan pada Kode 3.2. Hal ini dilanjuti proses mengeksekusi kueri SQL untuk menuju ke `SQLToCSV` yang mengubah hasil kueri menjadi `pandas.DataFrame`

seperti yang ditunjukkan pada Kode 3.4. DataFrame tersebut diolah oleh *system prompt* untuk mendapatkan hasil gambarnya, dapat dilihat pada Kode 3.5. Setelah itu, respons akan dikembalikan ke klien dengan `bot_message`-nya telah memiliki gambar plot yang dibuat seperti yang ditunjukkan pada Kode 3.3.

```
# Menjalankan query SQL dan mengubah hasilnya menjadi pandas.
    DataFrame
df = pd.read_sql(state["sql_query"], self._db_engine)
# Membentuk string schema kolom (CSV-like) dari nama-nama kolom
    DataFrame
headers = ", ".join(df.columns)
headers = f"{{{headers}}}"
# Mengambil konteks/penjelasan tujuan query dari state["goal"]
context = state["goal"]
# Menyusun pesan bertipe "data" untuk menyimpan konteks analitik
    ke riwayat chat
data_msg = {
    "text": context,
    "data_type": "data",
    "room_id": state["room_id"],
    "user_id": state["user_id"],
    "timestamp": None,
    "role": "bot",
}
# Mencatat pesan beserta DataFrame ke dalam SQLChatHistoryManager
    sebagai satu entri data
self.chat_manager.enter_message(SQLChatHistoryManager.Message(**
    data_msg), df)
```

Kode 3.4: Pengubahan SQL ke DataFrame

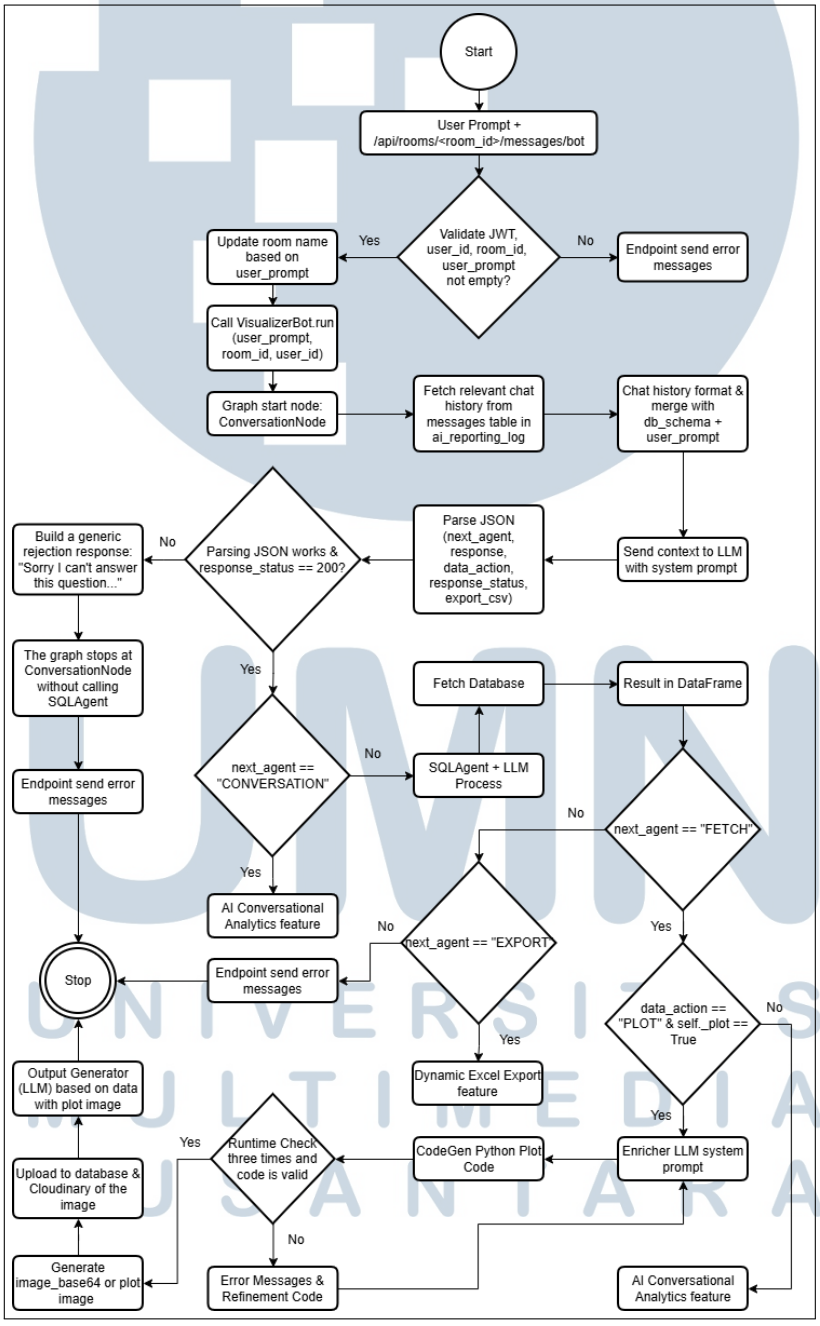
Sebelum dilakukan proses pengambilan hasil dari kueri SQL, kueri SQL yang dibentuk juga memiliki *system prompt* tersendiri, atau yang disebut dengan SQLAgent. Hal ini juga berlaku untuk keseluruhan fitur saat `next_agent` bernilai `FETCH`. Contoh sederhana dari pengaplikasian *system prompt* tersebut ditunjukkan seperti kode di bawah ini namun memiliki mekanisme kode pemrograman yang lebih rumit.

```
# Isi system prompt lengkap sesuai implementasi asli dengan
    matplotlib
SYSTEM_TEMPLATE = "You are an AI assistant that analyzes data and
    provides answers...."
# Isi user prompt lengkap sesuai implementasi asli
USER_TEMPLATE = "
```

```
User Question: {question}
Here is an SQL query to.... {sql_query}
The following is an explanation of information about the intent of
    the user's question (goal): {context}"
```

Kode 3.5: Contoh gambaran system prompt

Secara ringkas, alur kerja fitur *Automated Data Visualization* yang telah dijelaskan secara lengkap dapat disajikan dalam *flowchart* pada Gambar 3.32.



Gambar 3.32. *Flowchart* dari fitur *Automated Data Visualization*

2) Hasil Pengujian dan Output Fitur

Pengujian fitur *Automated Data Visualization* memeriksa deteksi kebutuhan visualisasi (`data_action="PLOT"`), pembuatan grafik yang sesuai dengan permintaan pengguna, pesan *error*, tampilan antarmuka, melakukan *follow-up questions*, mengunggah ke Cloudinary dan basis data, serta penempatan URL gambar ke riwayat chat sehingga dapat diakses oleh pengguna. Pada Tabel 3.13 menyajikan uraian pengujian, hasil yang diharapkan, hasil pengujian yang diperoleh, serta kesimpulan secara lebih rinci dan komprehensif.

Tabel 3.13. Hasil pengujian fitur *Automated Data Visualization*

No.	Pengujian	Skenario	Hasil Pengujian	Kesimpulan
1	Routing klasifikasi `next_agent` ke `"FETCH"` dan `data_action` bernilai `"PLOT"`	Pengguna dapat melakukan percakapan berbasis data dalam meminta grafik	LLM mengembalikan JSON sebagai jawaban dan dikembalikan dengan *query* SQL untuk pengguna dalam grafik	Sesuai dan Berhasil
2	Memeriksa kemampuan *memory management* untuk mekanisme *follow-up questions*	Pengguna dapat melakukan *follow-up questions*	Pengguna dapat memberikan pertanyaan lanjutan	Sesuai dan Berhasil
3	Memeriksa jenis grafik sesuai permintaan pengguna	Pengguna dapat meminta jenis grafik	Proses *plotting* sesuai dengan permintaan pengguna	Sesuai dan Berhasil

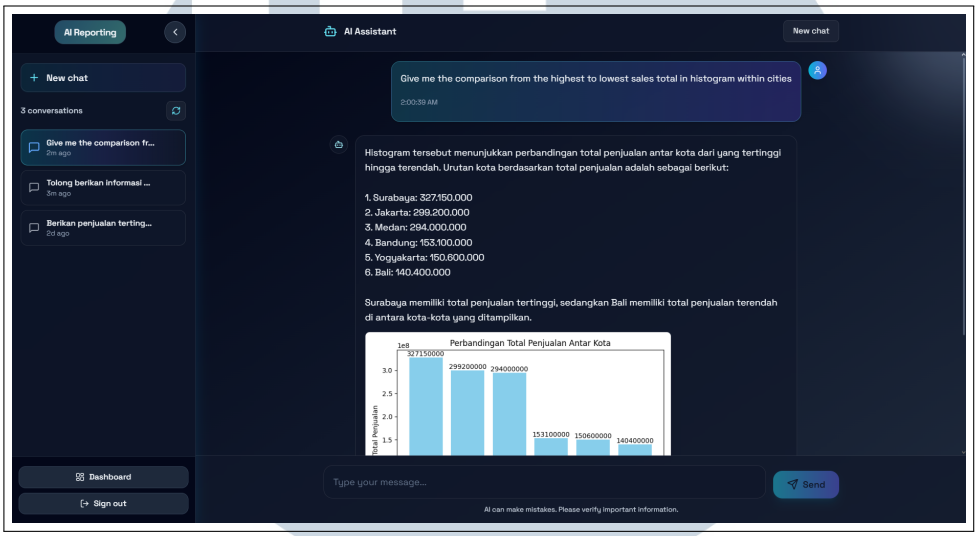
Lanjut pada halaman berikutnya

Lanjutan Tabel 3.13

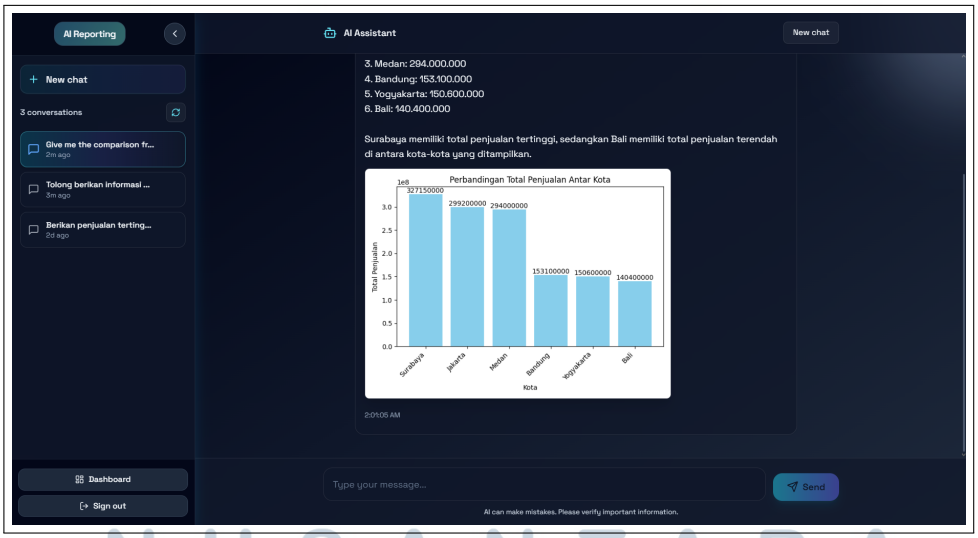
No.	Pengujian	Skenario	Hasil Pengujian	Kesimpulan
4	Memastikan ketersediaan *error messages*	Pengguna dapat memperoleh informasi terkait *error messages*	Tampilan beserta informasi *error messages* ditampilkan dengan jelas	Sesuai dan Berhasil
5	Melakukan pengujian konsistensi tampilan responsif	Pengguna dapat mengakses fitur *Automated Data Visualization* dengan antarmuka yang responsif	Antarmuka menu *Automated Data Visualization* ditampilkan secara responsif pada seluruh ukuran *device*	Sesuai dan Berhasil
6	Memastikan percakapan tersimpan ke database	Pengguna dapat melihat riwayat percakapan	Percakapan tersimpan ke database `ai_reporting_log`	Sesuai dan Berhasil
7	Memastikan gambar hasil *plotting* tersimpan ke database dan Cloudinary	Pengguna dapat melihat gambar dengan Url Cloudinary	File tersimpan ke database `ai_reporting_log` dan Cloudinary	Sesuai dan Berhasil
8	Memastikan permintaan pengguna sesuai dengan respons *chatbot*	Pengguna dapat bertanya dan jawabannya sesuai harapan	Hasil respons *chatbot* sesuai dengan pertanyaan pengguna	Sesuai dan Berhasil

Adapun contoh *output* yang relevan disajikan pada Gambar 3.33 dan 3.34, yang menampilkan hasil visualisasi otomatis yang diunggah dan direferensikan dalam riwayat chat berdasarkan pertanyaan pengguna. Pada Gambar 3.33 tertera user bertanya mengenai ”Give me the comparison from the highest to lowest sales total in histogram within cities”, dan *chatbot* merespons detail jawaban beserta bentuk grafik dalam histogram sebagaimana yang dapat dilihat pada

Gambar 3.34. Hal ini berarti sistem *backend* telah berhasil mengklasifikasikan `next_agent="FETCH"`, di mana hasil responsnya memerlukan pengambilan data di MySQL dan `data_agent="PLOT"` yang mengindikasikan adanya proses *plotting*. Bukti nyatanya dapat dilihat pada Gambar 3.35 yang berisikan respons memilih variabel `next_agent` bernilai `FETCH` dan `data_agent` bernilai `PLOT`.



Gambar 3.33. Contoh percakapan pengguna pada fitur *Automated Data Visualization* dengan `next_agent="FETCH"` dan `data_agent="PLOT"` (1)

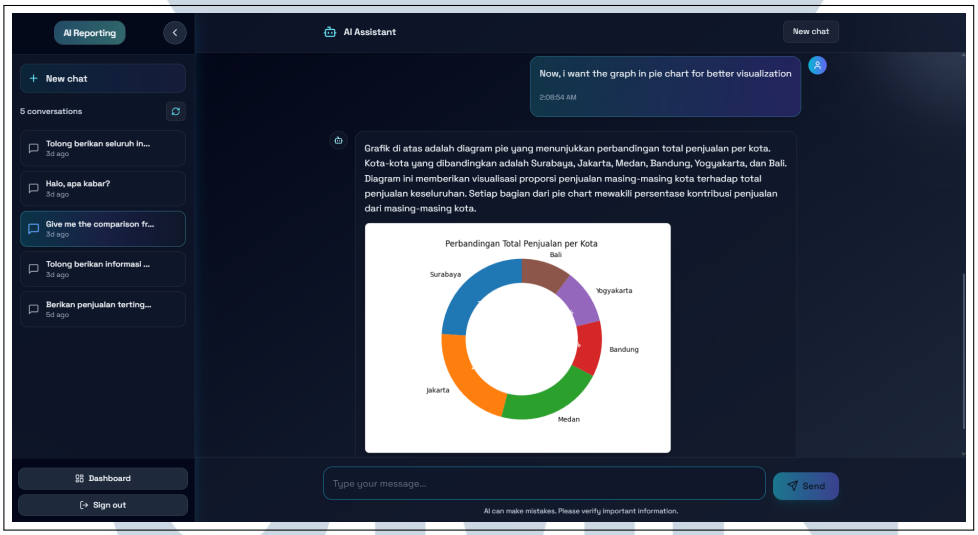


Gambar 3.34. Contoh percakapan pengguna pada fitur *Automated Data Visualization* dengan `next_agent="FETCH"` dan `data_agent="PLOT"` (2)

```
RESPONSE: {
    "next_agent": "FETCH",
    "response": "The user wants to compare sales totals across different cities and visualize this da
ta in a histogram, sorted from highest to lowest sales total.",
    "data_action": "PLOT",
    "response_status": "RESPONSE_OK",
    "export_csv": false
}
```


Gambar 3.35. Respons *backend* pada fitur *Automated Data Visualization* dengan `next_agent="FETCH"` dan `data_agent="PLOT"`

Selain itu, apabila pengguna meminta untuk mengubah grafiknya ke dalam bentuk lain, maka sistemnya akan berjalan ulang dan mengubahnya menjadi grafik yang diinginkan pengguna. Pada Gambar 3.36 tertera user bertanya lanjutan mengenai ”Now, i want the graph in pie chart for better visualization”, dan *chatbot* merespons detail jawaban beserta bentuk grafik dalam pie chart. Hal ini berarti hasil respons *chatbot* memenuhi permintaan pengguna.



Gambar 3.36. Contoh percakapan pengguna pada fitur *Automated Data Visualization* dengan pertanyaan lanjutan

E Fitur Dynamic Excel Export

1) Penjelasan Alur Kerja Fitur

Fitur *Dynamic Excel Export* menambahkan kemampuan Platform AI Reporting untuk menghasilkan file Excel secara dinamis dari hasil *query* dan *prompt user* yang dijalankan oleh SQLAgent. Kebutuhan *export* ini dideteksi oleh `ConversationNode` melalui field `export_csv` pada JSON hasil LLM oleh *system prompt*. Jika pengguna, dalam bahasa naturalnya,

menyatakan ingin mengekspor data ke Excel, LLM diinstruksikan untuk mengisi `export_csv=True`. Nilai ini kemudian diteruskan ke state graf dan sampai ke hasil *output* `VisualizerBot.run()` sebagai `response["export_csv"]`. Bersama dengan itu, agen `SQLToCSV` memastikan bahwa hasil query tersedia sebagai `pandas.DataFrame` pada field `response["pandas_dump"]` seperti yang telah ditunjukkan pada kode-kode sebelumnya.

Di endpoint `post_prompt`, *backend* memeriksa kedua kondisi tersebut sebelum memutuskan untuk membuat file Excel. Jika flag `export_csv` aktif dan `pandas_dump` tidak kosong, *backend* membuat file `.xlsx` di memori menggunakan `BytesIO` dan `pandas.ExcelWriter`. Kode 3.6 menunjukkan proses konversi DataFrame ke Excel.

```
# Mengambil ringkasan teks jawaban dari hasil bot
resp_content = response.get("summarized_output")
# Inisialisasi link Excel sebagai None
xlsx_link = None
# Mengecek apakah perlu ekspor Excel
if response.get("export_csv") and response["pandas_dump"] is not None:
    # Membuat buffer BytesIO untuk file Excel di memory
    export_out = BytesIO()
    # Mengambil DataFrame hasil query dari response
    df: pd.DataFrame = response["pandas_dump"]
    # Membuka writer Excel ke buffer memory
    with pd.ExcelWriter(export_out, engine="xlsxwriter") as writer:
        # Menulis DataFrame ke sheet Excel tanpa index
        df.to_excel(writer, index=False)
    # Mengatur pointer buffer ke awal sebelum upload
    export_out.seek(0)
    # Mengambil URL aman file Excel yang diunggah
    xlsx_link = cloudinary_upload["secure_url"]
```

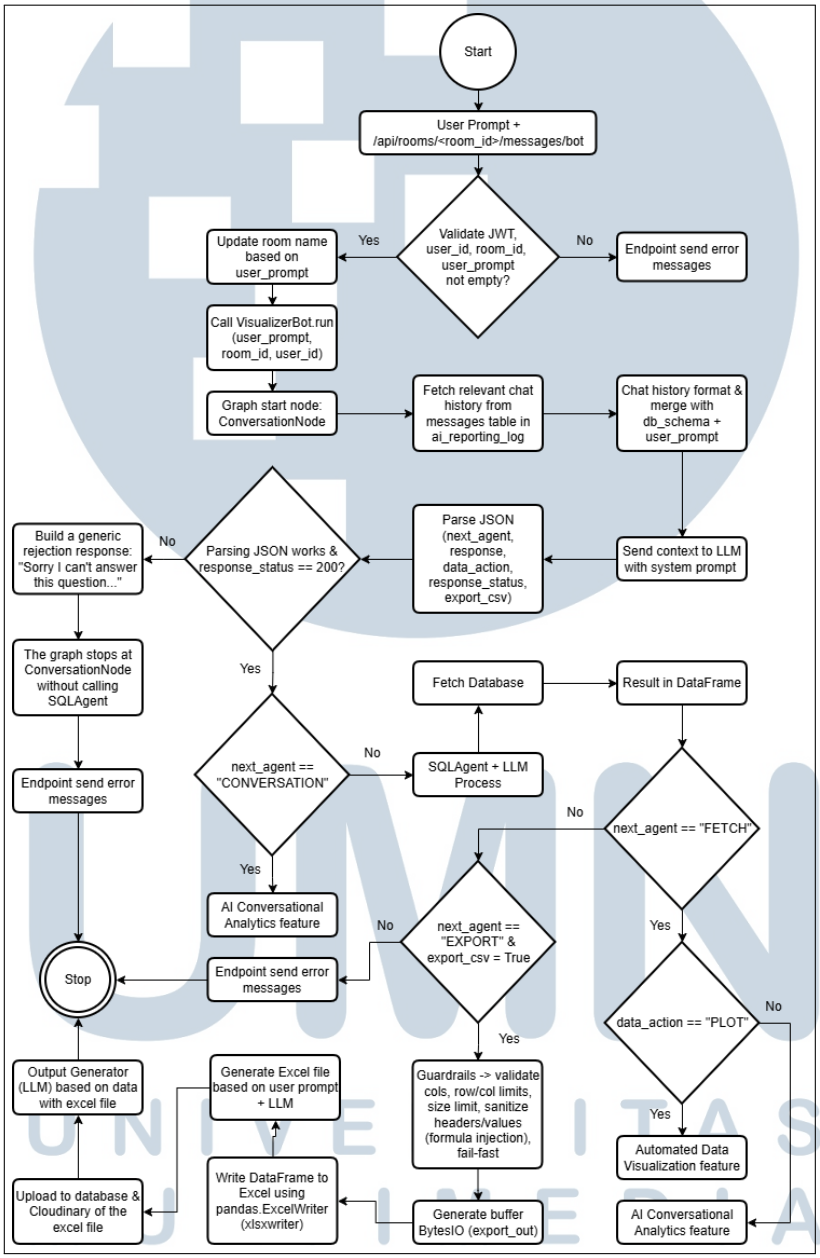
Kode 3.6: Konversi DataFrame ke Excel

Setelah `xlsx_link` diperoleh, *backend* menyimpan URL tersebut ke dalam field `document_url` dan `excel_url` di model `Message`, sehingga setiap jawaban yang menghasilkan file Excel memiliki referensi permanen di riwayat chat dan mengirimkan respons ke klien, seperti pada Kode 3.3, dengan variabel terbaru yang telah disebutkan sebelumnya.

Dengan desain ini, fitur *Dynamic Excel Export* menghubungkan klasifikasi LLM (`export_csv`), hasil query agen SQL (`pandas.DataFrame`), konversi ke

Excel di memory, upload ke Cloudinary, serta penyimpanan URL Excel ke dalam riwayat chat sehingga file tersebut dapat diakses kembali kapan saja.

Secara ringkas dari penjelasan di atas, mekanisme fitur *Dynamic Excel Export* ditampilkan dalam *flowchart* pada Gambar 3.37.



Gambar 3.37. *Flowchart* dari fitur *Dynamic Excel Export*

2) Hasil Pengujian dan Output Fitur

Pengujian fitur *Dynamic Excel Export* bertujuan memverifikasi apakah flag ekspor (`export_csv`) dikenali, melakukan *follow-up questions*, pesan *error*,

tampilan antarmuka, pembuatan file Excel, file Excel diunggah ke Cloudinary dan basis data, serta URL file dicatat pada riwayat chat sehingga dapat diunduh ulang. Pada Tabel 3.14 menyajikan uraian pengujian, hasil yang diharapkan, hasil pengujian yang diperoleh, serta kesimpulan secara lebih rinci dan komprehensif.

Tabel 3.14. Hasil pengujian fitur *Dynamic Excel Export*

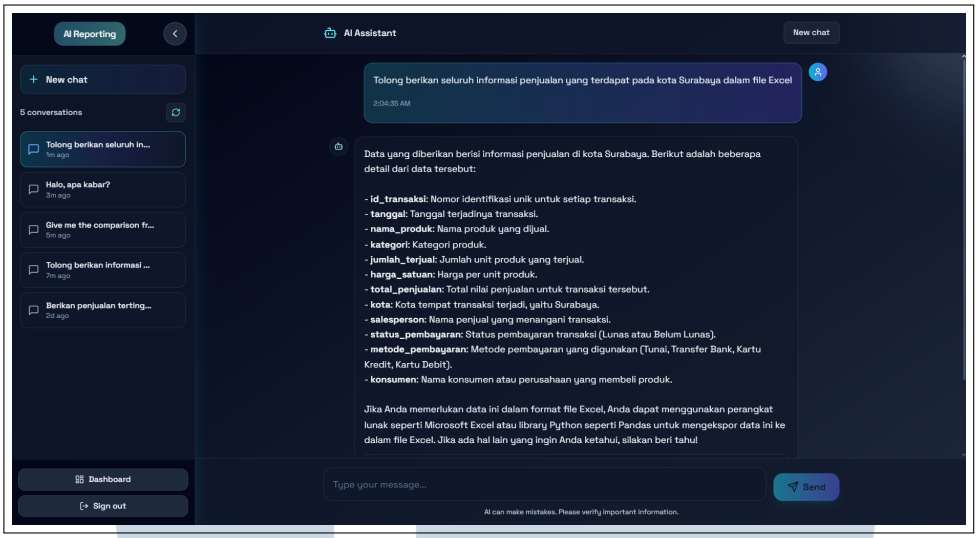
No.	Pengujian	Skenario	Hasil Pengujian	Kesimpulan
1	Routing klasifikasi `next_agent` ke `"EXPORT"` dan `export_csv` bernilai `True`	Pengguna dapat melakukan percakapan berbasis data dalam meminta file Excel	LLM mengembalikan JSON sebagai jawaban dan dikembalikan dengan *query* SQL untuk pengguna dalam file Excel	Sesuai dan Berhasil
2	Memeriksa kemampuan *memory management* untuk mekanisme *follow-up questions*	Pengguna dapat melakukan *follow-up questions*	Pengguna dapat memberikan pertanyaan lanjutan	Sesuai dan Berhasil
3	Memeriksa hasil pembuatan file Excel sesuai permintaan pengguna	Pengguna dapat meminta untuk mengisi file Excel dengan parameter apapun	Proses penulisan Excel sesuai dengan permintaan pengguna	Sesuai dan Berhasil
4	Memastikan ketersediaan *error messages*	Pengguna dapat memperoleh informasi terkait *error messages*	Tampilan beserta informasi *error messages* ditampilkan dengan jelas	Sesuai dan Berhasil

Lanjut pada halaman berikutnya

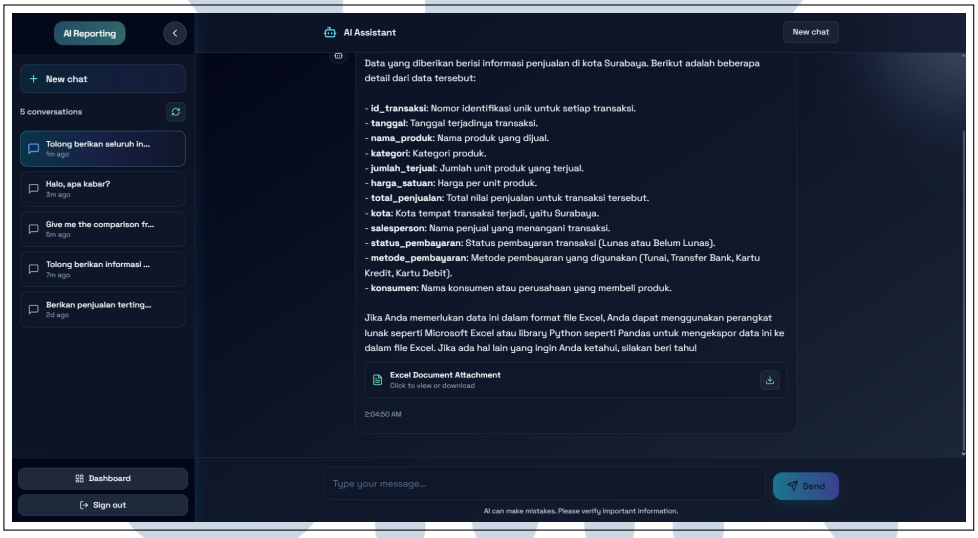
Lanjutan Tabel 3.14

No.	Pengujian	Skenario	Hasil Pengujian	Kesimpulan
5	Melakukan pengujian konsistensi tampilan responsif	Pengguna dapat mengakses fitur *Dynamic Excel Export* dengan antarmuka yang responsif	Antarmuka menu *Dynamic Excel Export* ditampilkan secara responsif pada seluruh ukuran *device*	Sesuai dan Berhasil
6	Memastikan percakapan tersimpan ke database	Pengguna dapat melihat riwayat percakapan	Percakapan tersimpan ke database `ai_reporting_log`	Sesuai dan Berhasil
7	Memastikan file Excel tersimpan ke database dan Cloudinary	Pengguna dapat melihat dan mengunduh file Excel dengan Url Cloudinary	File tersimpan ke database `ai_reporting_log` dan Cloudinary	Sesuai dan Berhasil
8	Memastikan permintaan pengguna sesuai dengan respons *chatbot*	Pengguna dapat bertanya dan jawabannya sesuai harapan	Hasil respons *chatbot* sesuai dengan pertanyaan pengguna	Sesuai dan Berhasil

Adapun contoh *output* dapat dilihat pada Gambar 3.38 dan 3.39 yang menggambarkan tampilan pertanyaan pengguna dalam hal meminta file Excel dengan data yang diinginkan pengguna. Pada Gambar 3.38 tertera user bertanya mengenai ”Tolong berikan seluruh informasi penjualan yang terdapat pada kota Surabaya dalam file Excel”, dan *chatbot* merespons detail jawaban beserta file Excel yang dapat diunduh sebagaimana yang dapat dilihat pada Gambar 3.39. Selain itu, pada Gambar 3.40 menunjukkan hasil file Excel juga menunjukkan semua kotanya berada di Surabaya sesuai dengan permintaan pengguna. Hal ini berarti sistem *backend* telah berhasil mengklasifikasikan `next_agent="EXPORT"`, di mana hasil responsnya memerlukan pengambilan data di MySQL dan mengolahnya dengan *xlsxwriter*. Bukti nyatanya dapat dilihat pada Gambar 3.41 yang berisikan respons memilih variabel `next_agent` bernilai `EXPORT`.



Gambar 3.38. Contoh percakapan pengguna pada fitur *Dynamic Excel Export* dengan `next_agent="EXPORT"` (1)



Gambar 3.39. Contoh percakapan pengguna pada fitur *Dynamic Excel Export* dengan `next_agent="EXPORT"` (2)

id_transaksi	tanggal	nama_produk	kategori	jumlah_terjual	harga_satuan	total_penjualan	kota	salesperson	status_pembayaran	metode_pembayaran	konsumen
3	2024-12-02	Tas Ransel PQR	Aksesoris	10	500000	5000000	Surabaya	Charlie Lee	Belum Lunas	Tunai	Perusahaan C
7	2024-12-06	Kamera GHI	Elektronik	1	5500000	5500000	Surabaya	Charlie Lee	Lunas	Transfer Bank	Perusahaan C
12	2024-12-11	Smartwatch GHI	Elektronik	6	2500000	15000000	Surabaya	Charlie Lee	Belum Lunas	Transfer Bank	Perusahaan G
19	2024-12-18	Tablet Pro QRS	Elektronik	3	8000000	24000000	Surabaya	Alice Johnson	Lunas	Kartu Kredit	Perusahaan N
22	2024-12-21	Meja Kantor ABC	Furniture	2	2000000	4000000	Surabaya	Charlie Lee	Lunas	Transfer Bank	Perusahaan Q
28	2024-12-27	Set Alat Makan PQR	Furniture	8	450000	3600000	Surabaya	Diana Prince	Lunas	Tunai	Perusahaan W
34	2024-12-30	Smartwatch Model TUV	Elektronik	4	3500000	14000000	Surabaya	Diana Prince	Belum Lunas	Kartu Debit	Perusahaan C
39	2024-12-25	Set Meja Belajar VWX	Furniture	5	2000000	10000000	Surabaya	Diana Prince	Lunas	Kartu Debit	Perusahaan H
45	2024-12-19	Smartphone Dual SIM PQR	Elektronik	10	4500000	45000000	Surabaya	Charlie Lee	Lunas	Kartu Kredit	Perusahaan N
53	2024-12-28	Smartphone High-end XYZ	Elektronik	3	7000000	21000000	Surabaya	Charlie Lee	Lunas	Tunai	Perusahaan C
59	2024-12-22	Tablet Surface PQR	Elektronik	6	8000000	48000000	Surabaya	Bob Smith	Belum Lunas	Kartu Kredit	Perusahaan I
64	2024-12-17	Set Panci Stainless STU	Furniture	3	1500000	4500000	Surabaya	Alice Johnson	Lunas	Transfer Bank	Perusahaan N
70	2024-12-11	Smartwatch Fitness GHI	Elektronik	6	2000000	12000000	Surabaya	Diana Prince	Belum Lunas	Transfer Bank	Perusahaan T
73	2024-12-29	Lemari Pakaian Modern DEF	Furniture	4	2500000	10000000	Surabaya	Bob Smith	Belum Lunas	Kartu Kredit	Perusahaan W
80	2024-12-22	Kamera DSLR PQR	Elektronik	4	8000000	32000000	Surabaya	Bob Smith	Lunas	Transfer Bank	Perusahaan AD
83	2024-12-19	Kursi Bar Bergaya Modern GHI	Furniture	4	1200000	4800000	Surabaya	Bob Smith	Lunas	Transfer Bank	Perusahaan AG
88	2024-12-14	Set Meja Makan Stainless LMN	Furniture	3	5000000	15000000	Surabaya	Alice Johnson	Lunas	Kartu Debit	Perusahaan AL
92	2024-12-30	Set Peralatan Outdoor DEF	Outdoor	5	750000	3750000	Surabaya	Bob Smith	Lunas	Kartu Debit	Perusahaan AP
96	2024-12-26	Paket Kamera Profesional GHI	Elektronik	2	15000000	30000000	Surabaya	Bob Smith	Lunas	Tunai	Perusahaan AT
102	2024-12-20	Peralatan Dapur Set UVW	Furniture	12	1500000	18000000	Surabaya	Alice Johnson	Lunas	Kartu Debit	Perusahaan AZ
107	2024-12-15	Set Piring Keramik XYZ	Furniture	10	200000	2000000	Surabaya	Bob Smith	Belum Lunas	Kartu Debit	Perusahaan BE

Gambar 3.40. Hasil pembuatan file Excel pada fitur *Dynamic Excel Export* sesuai permintaan pengguna


```
RESPONSE: {
    "next_agent": "EXPORT",
    "response": "User requests to export all sales information from Surabaya into an Excel file.",
    "data_action": "NO_PLOT",
    "response_status": "RESPONSE_OK",
    "export_csv": true
}
```


Gambar 3.41. Respons *backend* pada fitur *Dynamic Excel Export* dengan `next_agent="EXPORT"`

F Fitur OCR Data Intake

1) Penjelasan Alur Kerja Fitur

Fitur *OCR Data Intake* menyediakan jalur khusus bagi *administrator* untuk memasukkan data transaksi ke dalam sistem melalui unggahan dokumen PDF. Alur dimulai dari API endpoint `/api/admin/transactions/upload` dan fungsi `admin_upload_transaction` memvalidasi bahwa file yang diterima benar-benar PDF, membaca seluruh isi file, mengekstrak teks dan jumlah halaman menggunakan *system prompt* tersendiri, mencatat riwayat proses ke tabel `ocr_history`, memanggil LLM untuk mengubah teks bebas menjadi *payload* transaksi yang terstruktur, menormalkan *payload* tersebut yang merupakan bagia dari *system prompt tadi*, dan akhirnya menyimpan atau memperbarui data ke Tabel `transaksi`. Potongan awal fungsi ini diperlihatkan pada Kode 3.7.

```python
# Mendefinisikan route POST admin untuk upload PDF
@app.route("/api/admin/transactions/upload", methods=["POST"])
def admin_upload_transaction():
    # Inisialisasi ID riwayat OCR sebagai None
    history_id = None
    try:
```

```
7         # Memastikan ada field file di request
8         if "file" not in request.files:
9             return (
10                jsonify({"status_code": "ADM-400", "error": "File
    PDF wajib dikirim"}),
11                400,
12            )
13        # Mengambil objek file dari request
14        file = request.files["file"]
15        # Memastikan file memiliki ekstensi .pdf
16        if not file or not file.filename.lower().endswith(".pdf"):
17            return (
18                jsonify(
19                    {"status_code": "ADM-400", "error": "Hanya
    file PDF yang diterima"}
20                ),
21                400,
22            )
23        # Membaca seluruh isi file PDF ke bytes
24        file_bytes = file.read()
25        # Mengekstrak teks mentah dan jumlah halaman
26        raw_text, page_count = extract_text_and_meta_from_pdf(
    file_bytes)
27        # Mencatat entri riwayat awal di tabel ocr_history
28        history_id = create_ocr_history(
29            # Menyimpan nama file yang diunggah
30            filename=file.filename,
31            # Menyimpan ukuran file dalam bytes
32            filesize=len(file_bytes),
33            # Menyimpan jumlah halaman PDF
34            page_count=page_count,
35            # Menandai status awal sebagai processing
36            status="processing",
37            # Menyimpan pesan status awal
38            message="Mengunggah & memproses dokumen",
39        )
40        # Memanggil LLM untuk mengekstrak field transaksi
41        llm_result = extract_transaction_from_text(raw_text)
42        # Mengembalikan respon sukses ke klien admin
43        return (
44            jsonify(
45                {
46                    # Menandai status sukses ADM-000
```

```
                    "status_code": "ADM-000",
                    # Pesan status sukses
                    "message": "Dokumen berhasil diekstrak",
                    # Mengembalikan data transaksi yang tersimpan
                    "data": persisted,
                    # Mengembalikan detail riwayat
                    "history": get_ocr_history_entry(history_id)
                    if history_id
                    else None,
                }
            ),
            200,
        )
```

Kode 3.7: Endpoint upload PDF untuk proses OCR dan ekstraksi transaksi

Fungsi `extract_text_and_meta_from_pdf` yang dipanggil di awal bertugas melakukan ekstraksi teks dari PDF menggunakan `PyPDF2`. Fungsi ini membaca semua halaman, memanggil `extract_text()` pada masing-masing halaman, dan menggabungkannya menjadi satu string panjang, sekaligus menghitung jumlah halaman. Kode 3.8 menunjukkan implementasinya dengan *system prompt* tersendiri.

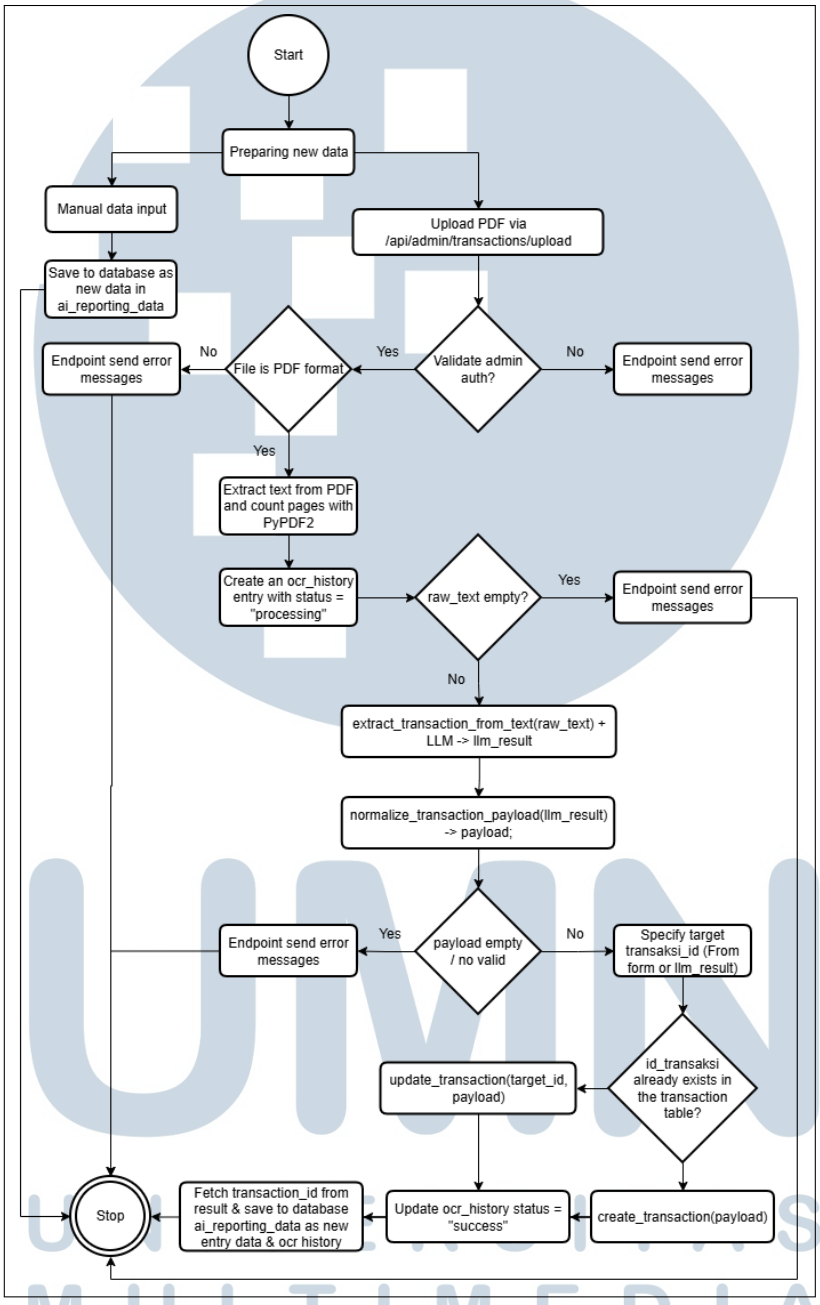
```
def extract_transaction_from_text(raw_text: str):
    # Jika teks OCR kosong, tidak ada transaksi yang bisa
    diekstrak
    if not raw_text:
        return []
    # System prompt khusus untuk mendefinisikan format dan kolom
    transaksi
    system_prompt = "..."  # Isi system prompt
    # Batasi panjang teks agar tetap aman dalam batas token LLM
    truncated = raw_text[:6000]
    # Susun pesan system + human yang akan dikirim ke LLM
    messages = [
        SystemMessage(content=...),
        HumanMessage(content=...)
        ),
    ]
    response = None
    try:
        # Minta LLM merespon dalam format JSON terstruktur
        response = model.invoke(messages, response_format={"type":
    "json_object"})
```

```
    except Exception as exc:
        print(f"[OCR] Structured LLM extraction failed, retrying without enforced JSON: {exc}")
        try:
            # Fallback: minta respons bebas, nanti diparse manual
            response = model.invoke(messages)
        except Exception as final_exc:
            print(f"[OCR] LLM extraction failed: {final_exc}")
            return []
    # Ambil konten mentah dari LLM dan parse menjadi struktur JSON Python
    raw_content = response.content
    content = raw_content if isinstance(raw_content, str) else raw_content
    parsed = _parse_json_like(content)
    if not parsed:
        print(f"[OCR] Could not parse LLM response: {content}")
        return []
    # Normalisasi struktur hasil parsing ke list[dict] transaksi
    records = []
    if isinstance(parsed, dict):
        rows = None
        for key in ("rows", "data", "transactions", "items"):
            if key in parsed:
                rows = parsed[key]
                break
        if isinstance(rows, list):
            records = [r for r in rows if isinstance(r, dict)]
        else:
            records = [parsed]
    elif isinstance(parsed, list):
        records = [r for r in parsed if isinstance(r, dict)]
    return records
```

Kode 3.8: Ekstraksi teks dan jumlah halaman dari file PDF

Dengan pembuatan komponen pengunggahan data, ekstrak isi PDF, mengubah menjadi *payload* untuk diproses oleh LLM, dan normalisasi tipe datanya, fitur *OCR Data Intake* membentuk sebuah *pipeline* yang lengkap dan dapat diaudit. Server mengekstrak teks dan jumlah halaman dan LLM mengubah teks tersebut menjadi JSON yang sesuai dengan skema transaksi, serta fungsi normalisasi membersihkan dan mengonversi tipe data.

Secara ringkas, proses pada fitur *OCR Data Intake* dapat digambarkan melalui *flowchart* pada Gambar 3.42.



Gambar 3.42. *Flowchart* dari fitur *OCR Data Intake*

2) Hasil Pengujian dan Output Fitur

Pengujian fitur *OCR Data Intake* difokuskan pada alur admin, dimulai dari upload PDF, CRUD data, ekstraksi teks, pemanggilan LLM untuk mengubah teks mentah menjadi *payload* transaksi atau data baru secara terstruktur, pesan *error*,

tampilan antarmuka, dan penyimpanan data di Tabel `transaksi`. Pada Tabel 3.15 menyajikan uraian pengujian, hasil yang diharapkan, hasil pengujian yang diperoleh, serta kesimpulan secara lebih rinci dan komprehensif.

Tabel 3.15. Hasil pengujian fitur *OCR Data Intake*

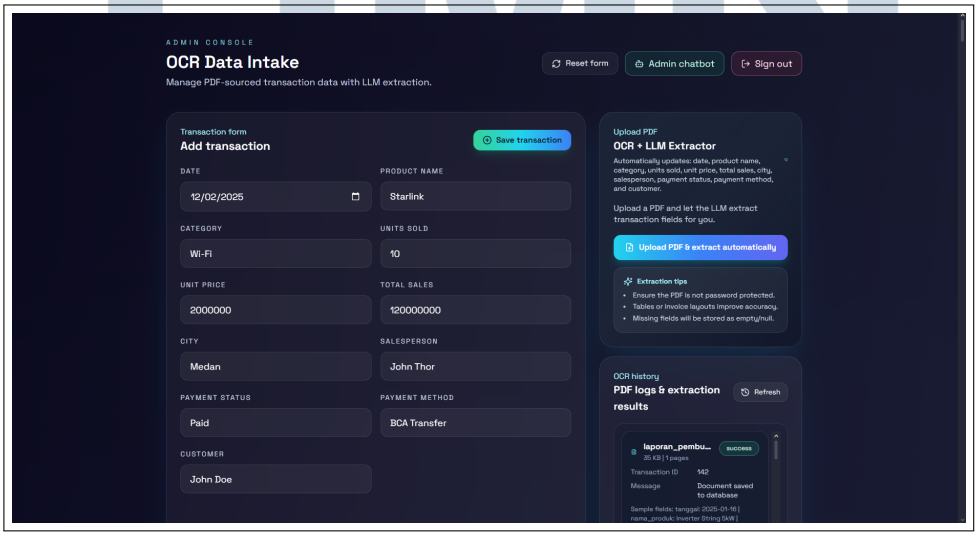
No.	Pengujian	Skenario	Hasil Pengujian	Kesimpulan
1	Upload PDF valid	Admin upload PDF dengan teks jelas	PDF dapat dikenali oleh LLM untuk diekstrak	Sesuai dan Berhasil
2	CRUD data	Admin dapat melakukan CRUD pada semua data	Admin dapat menambah, mengubah, maupun menghapus data dan sudah ter-*update* dengan database `ai_reporting_data`	Sesuai dan Berhasil
3	Kemampuan ekstraksi OCR via LLM	LLM mengubah teks menjadi JSON data	JSON sesuai field yang diizinkan	Sesuai dan Berhasil
4	Hasil ekstraksi PDF	Admin dapat melakukan penambahan data dengan benar tanpa cara inputan manual	Hasil ekstraksi sesuai secara keseluruhan	Sesuai dan Berhasil
5	Memastikan adanya *error messages*	Pengguna dapat melihat informasi *error messages*	Tampilan dan informasi *error messages* muncul	Sesuai dan Berhasil

Lanjut pada halaman berikutnya

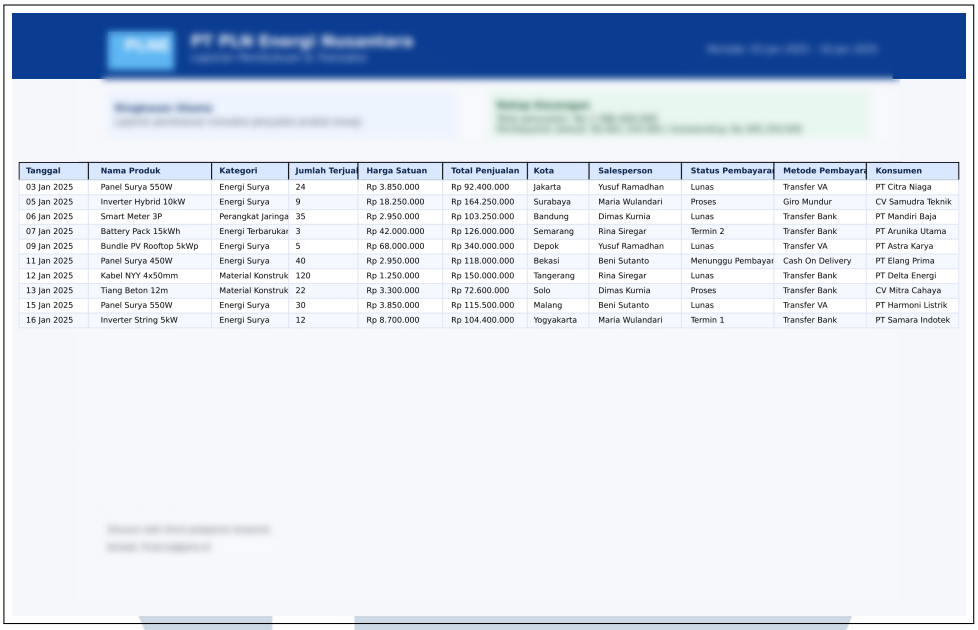
Lanjutan Tabel 3.15

No.	Pengujian	Skenario	Hasil Pengujian	Kesimpulan
6	Melakukan pengujian konsistensi tampilan responsif	Admin dapat mengakses fitur *OCR Data Intake* dengan antarmuka yang responsif	Antarmuka menu *OCR Data Intake* ditampilkan secara responsif pada seluruh ukuran *device*	Sesuai dan Berhasil
7	Memastikan data dan hasil ekstraksi OCR tersimpan ke database	Admin dapat melihat data baru di halaman admin	Data baru dan hasil ekstraksi OCR tersimpan ke database `ai_reporting_data`	Sesuai dan Berhasil

Adapun contoh implementasi fitur *OCR Data Intake* diawali dengan penginputan dokumen berekstensi PDF ke dalam halaman admin yang dapat dilihat pada Gambar 3.45. Untuk isi dari dokumen PDF sebagai *data dummy* dicantumkan pada Gambar 3.44. Akan tetapi, sebelumnya admin dapat melakukan penginputan data secara manual seperti yang ditunjukkan pada Gambar 3.43 dan datanya masuk ke basis data `ai_reporting_data`. Setelah proses OCR selesai, maka akan muncul sebuah tampilan konfirmasi hasil ekstraksi OCR yang dapat dilihat pada Gambar 3.46. Apabila pengecekan telah sesuai oleh admin, maka datanya akan dimasukkan ke basis data seperti yang ditunjukkan pada Gambar 3.47.

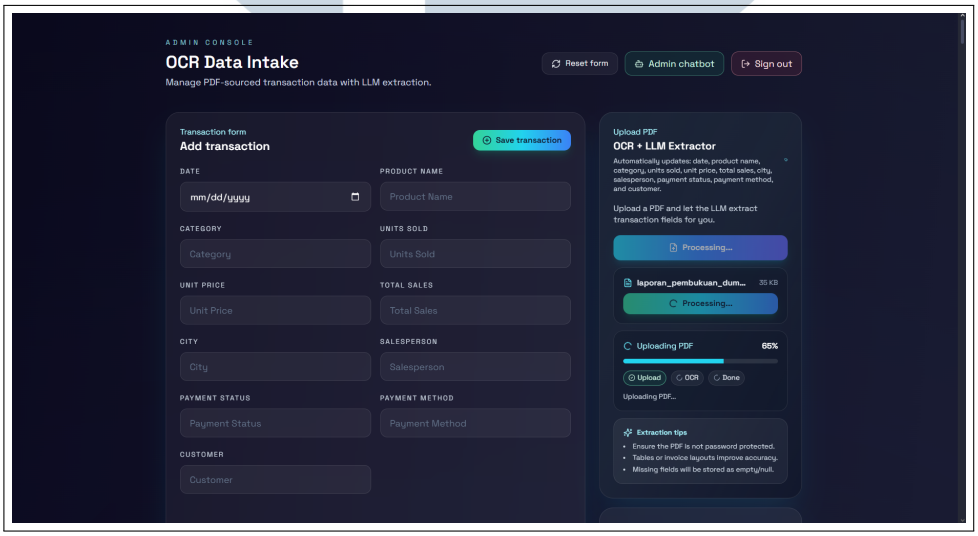


Gambar 3.43. Contoh inputan data secara manual pada fitur *OCR Data Intake*

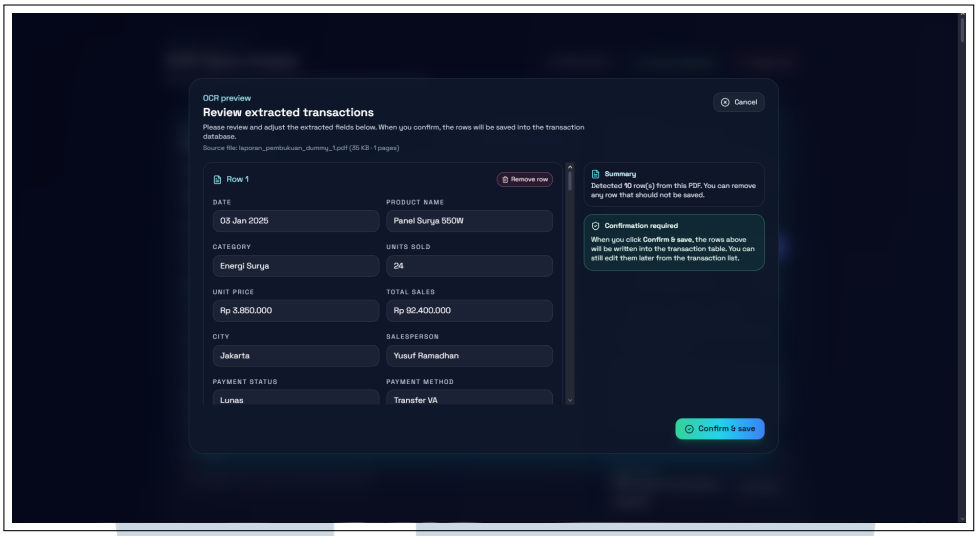


Tanggal	Nama Produk	Kategori	Jumlah Terjual	Harga Satuan	Total Penjualan	Kota	Salesperson	Status Pembayara	Metode Pembayara	Konsumen
03 Jan 2025	Panel Surya 550W	Energi Surya	24	Rp 3.850.000	Rp 92.400.000	Jakarta	Yusuf Ramadhan	Lunas	Transfer VA	PT Citra Niaga
05 Jan 2025	Inverter Hybrid 10kW	Energi Surya	9	Rp 18.250.000	Rp 164.250.000	Surabaya	Maria Wulandari	Proses	Giro Mundur	CV Samudra Teknik
06 Jan 2025	Smart Meter 3P	Perangkat Jaringa	35	Rp 2.950.000	Rp 103.250.000	Bandung	Dimas Kurnia	Lunas	Transfer Bank	PT Mandiri Baja
07 Jan 2025	Battery Pack 15kWh	Energi Terbarukar	3	Rp 42.000.000	Rp 126.000.000	Semarang	Rina Siregar	Termin 2	Transfer Bank	PT Arunika Utama
09 Jan 2025	Bundle PV Rooftop 5kWp	Energi Surya	5	Rp 68.000.000	Rp 340.000.000	Depok	Yusuf Ramadhan	Lunas	Transfer VA	PT Astra Karya
11 Jan 2025	Panel Surya 450W	Energi Surya	40	Rp 2.950.000	Rp 118.000.000	Bekasi	Beni Sutanto	Menunggu Pembayar	Cash On Delivery	PT Elang Prima
12 Jan 2025	Kabel NYY 4x50mm	Material Konstruk	120	Rp 1.250.000	Rp 150.000.000	Tangerang	Rina Siregar	Lunas	Transfer Bank	PT Delta Energi
13 Jan 2025	Tiang Beton 12m	Material Konstruk	22	Rp 3.300.000	Rp 72.600.000	Solo	Dimas Kurnia	Proses	Transfer Bank	CV Mitra Cahaya
15 Jan 2025	Panel Surya 550W	Energi Surya	30	Rp 3.850.000	Rp 115.500.000	Malang	Beni Sutanto	Lunas	Transfer VA	PT Harmoni Listrik
16 Jan 2025	Inverter String 5kW	Energi Surya	12	Rp 8.700.000	Rp 104.400.000	Yogyakarta	Maria Wulandari	Termin 1	Transfer Bank	PT Samara Indotek

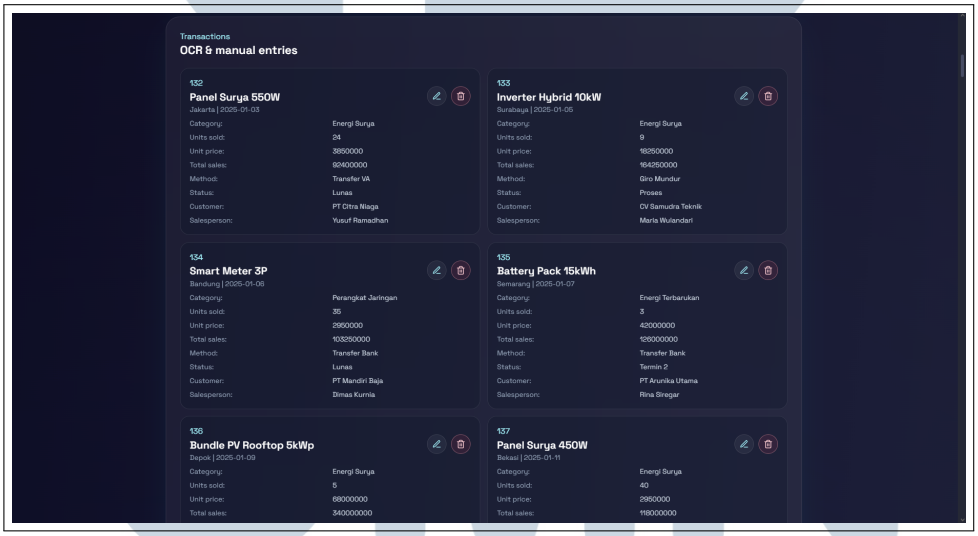
Gambar 3.44. Contoh tampilan *data dummy* untuk proses OCR pada fitur *OCR Data Intake*



Gambar 3.45. Contoh tampilan proses OCR pada fitur *OCR Data Intake*



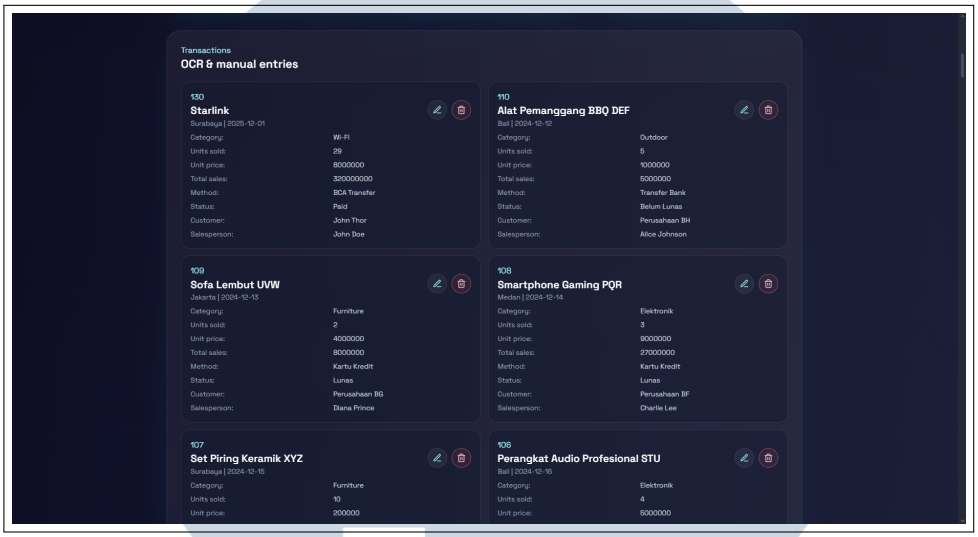
Gambar 3.46. Contoh tampilan konfirmasi hasil ekstrak dari proses OCR pada fitur *OCR Data Intake*



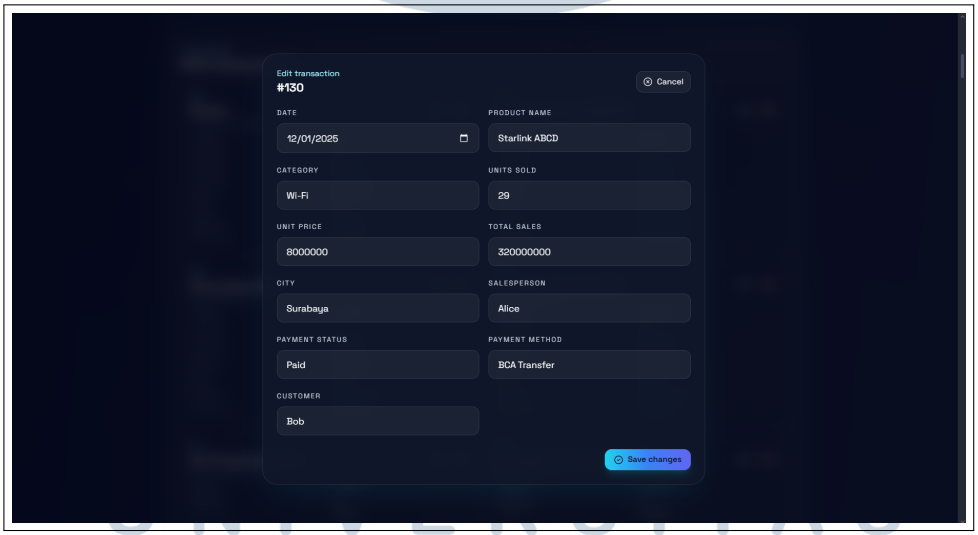
Gambar 3.47. Contoh tampilan hasil ekstrak dari proses OCR pada fitur *OCR Data Intake*

Selain itu, pada Gambar 3.48 dapat dilihat sebuah data dengan id 130 yang memiliki field nama produk (`Starlink`), lokasi (`Surabaya`), tanggal transaksi (2025-12-01), kategori (Wi-Fi), unit terjual (29), harga satuan (Rp,8.000.000), total penjualan (Rp,320.000.000), metode pembayaran (BCA Transfer), status (Paid), customer (John Thor), dan salesperson (John Doe). Untuk menjelaskan alur pengeditan, misalnya admin mengubah `name` menjadi `Starlink ABCD`, `customer` menjadi `Bob`, dan `salesperson` menjadi `Alice` seperti yang ditunjukkan pada Gambar 3.51, maka setelah admin memasukkan nilai baru dan menekan tombol

simpan sistem akan melakukan validasi input seperti cek kelengkapan dan format. Kemudian data barunya ditampilkan kembali pada halaman detail yang sudah diperbarui.



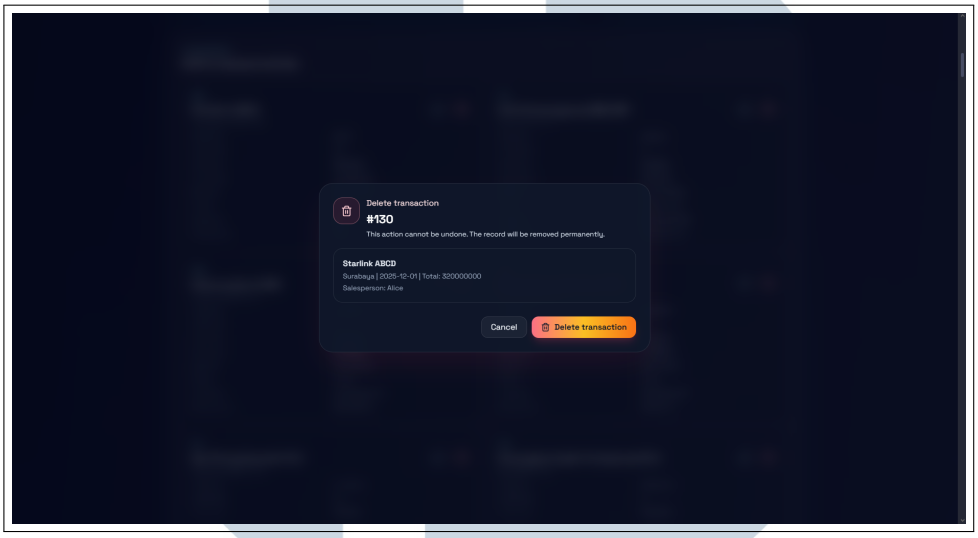
Gambar 3.48. Contoh tampilan data sebelum diubah



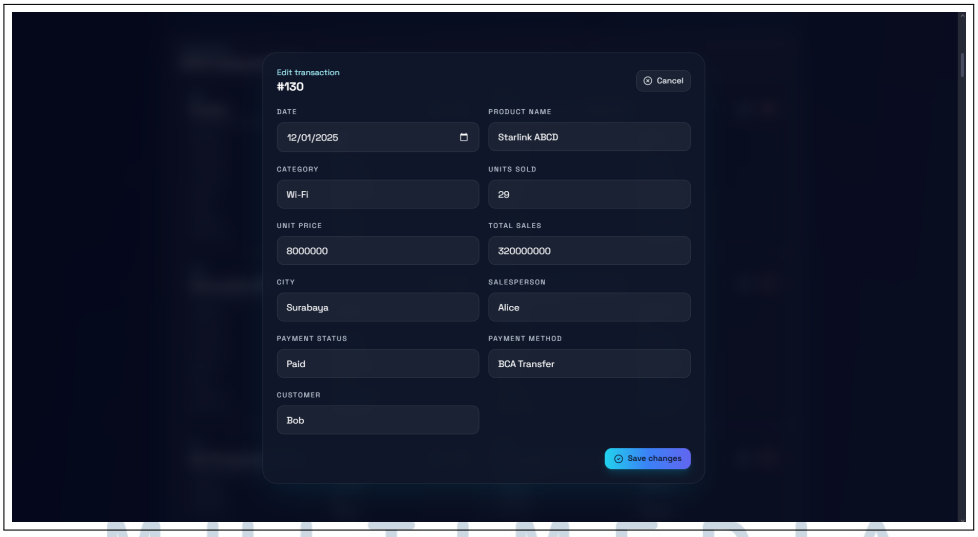
Gambar 3.49. Contoh tampilan data saat sedang diubah

Selain pengeditan data, sistem juga menyediakan fitur penghapusan data yang bertujuan untuk menghilangkan data yang sudah tidak diperlukan, sebagaimana ditunjukkan pada Gambar 3.51 dengan mengambil contoh data berid 130. Ketika pengguna menekan aksi hapus pada data tersebut, sistem akan mengeluarkan sebuah tampilan *pop-up* konfirmasi seperti yang tertera pada Gambar

3.50 untuk memastikan bahwa tindakan penghapusan dilakukan secara sadar dan mencegah kesalahan yang tidak diinginkan, kemudian setelah pengguna menyetujui konfirmasi tersebut sistem akan memproses penghapusan data dari basis data dan data yang bersangkutan tidak lagi muncul pada daftar maupun halaman detail.



Gambar 3.50. Contoh tampilan data sebelum dihapus



Gambar 3.51. Contoh tampilan data saat sedang dihapus

Pengujian terhadap keempat fitur utama pada Platform *AI Reporting* dilakukan menggunakan metode *White Box Testing*. Pendekatan ini dilakukan dengan menelaah kode sumber secara langsung guna memastikan implementasi sistem terbebas dari kesalahan logika maupun kesalahan struktural. Jika keluaran

(*output*) yang dihasilkan belum sesuai dengan hasil yang diharapkan, maka dilakukan evaluasi ulang terhadap kode program disertai proses kompilasi ulang hingga diperoleh hasil yang sesuai. Melalui pengujian ini, pengguna platform dapat memahami alur kerja sistem secara menyeluruh [85]. Seluruh hasil pengujian tersebut telah melalui tahap verifikasi dan memperoleh persetujuan dari supervisor, sebagaimana tercantum pada Lampiran 10.

3.6 Kendala dan Solusi yang Ditemukan

3.6.1 Kendala

Dalam prosedur perancangan Platform AI Reporting yang memiliki empat fitur utama, yaitu *AI Conversational Analytics*, *Automated Data Visualization*, *Dynamic Excel Export*, dan *OCR Data Intake*, terdapat beberapa kendala teknis dan non-teknis yang muncul. Kendala-kendala utama yang diidentifikasi selama pelaksanaan magang adalah sebagai berikut.

1. Kualitas dan akurasi hasil OCR kadang menghasilkan kesalahan pengenalan angka, tabel, dan format sehingga data kotor masuk *pipeline*.
2. LLM menghasilkan kueri SQL yang tidak valid, tidak efisien, atau berisiko terhadap integritas data.
3. Ketidakcocokan versi dependensi seperti Python, Docker, dan driver sistem basis data antar lingkungan sistem operasi antar tim.
4. Permasalahan dalam pengelolaan versi dependensi pada pustaka AI, seperti LangChain, Plot Agent, maupun modul OCR, yang pada kondisi tertentu mengalami ketidakcocokan satu sama lain.
5. Penyesuaian DBMS dan normalisasi tabel saat merancang Platform AI Reporting yang membutuhkan waktu lama untuk menghadapi berbagai kemungkinan buruk yang akan muncul.

3.6.2 Solusi

Pada penjabaran setiap kendala di atas yang telah disebutkan sebelumnya, solusi untuk menanggulangi tiap kendala dirancang dan diimplementasikan diarahkan agar dapat langsung menanggulangi akar masalah serta meningkatkan

keandalan dan pengalaman pengguna Platform AI Reporting. Solusi terlampir disusun sedemikian rupa sehingga setiap poin solusi menjawab kendala yang bersesuaian.

1. Meningkatkan kualitas OCR dengan menambahkan pengecekan otomatis dan menyediakan halaman koreksi agar data yang masuk selalu bersih dan akurat.
2. Membuat *system prompt* yang lebih jelas dan mudah dipahami LLM agar kueri SQL yang dihasilkan lebih valid dan sesuai kebutuhan oleh SQLAgent, serta mengganti ke model LLM yang lebih mahal.
3. Menyatukan versi dependensi dengan *environment* yang seragam untuk menghindari perbedaan perilaku antar sistem operasi atau melakukan *dependency locking*.
4. Menetapkan versi pustaka AI yang stabil dan mengecek kompatibilitasnya secara berkala agar sistem tetap berjalan tanpa konflik.
5. Mendesain ulang struktur sistem basis data secara bertahap dan terencana agar proses penyesuaian tabel dan normalisasi lebih cepat serta mudah dipelihara.

3.7 Hasil Pelaksanaan Magang

Pelaksanaan magang menghasilkan kerangka kerja fungsional Platform AI Reporting yang mencakup perancangan arsitektur, implementasi modul inti, dokumentasi teknis, dan prototipe antarmuka yang siap diuji lanjutan. Aktivitas magang difokuskan pada orkestrasi alur data dari inputan data secara manual atau OCR, hingga penyajian jawaban analitik, visualisasi, dan ekspor laporan dalam bentuk Excel, dengan penekanan pada keamanan (JWT), manajemen berkas menggunakan Cloudinary dan MySQL, serta integrasi LangChain sebagai penghubung antara LLM dan modul-modul pendukung. Adapun fitur-fitur yang telah berhasil diimplementasikan adalah sebagai berikut.

- **AI Conversational Analytics:** Percakapan yang memungkinkan pengguna mengajukan pertanyaan analitik; Permintaan yang digabungkan dengan *system prompt template* dan diproses oleh LLM untuk menghasilkan jawaban naratif atau instruksi kueri.

- **Automated Data Visualization:** Integrasi *Plot Agent* untuk menghasilkan visualisasi otomatis sesuai permintaan pengguna dalam berbagai bentuk grafik.
- **Dynamic Excel Export:** Ekspor hasil kueri ke format Excel yang terstruktur dengan opsi format yang diminta pengguna.
- **OCR Data Intake:** Alur pengunggahan PDF ke Cloudinary, proses ekstraksi OCR, penyimpanan raw dan terverifikasi, serta antarmuka admin untuk koreksi CRUD.

Selain implementasi fitur-fitur inti di atas, hasil magang mencakup dokumentasi yang lengkap, termasuk diagram yang menjelaskan Platform AI Reporting, spesifikasi perangkat lunak, spesifikasi perangkat keras, spesifikasi *library* dan *framework* yang digunakan, penyusunan halaman dan komponen yang diperlukan, penjelasan *endpoint* API yang digunakan, dan tampilan antarmuka pengguna yang mengarahkan alur kerja minimal dari akses ke Platform AI Reporting hingga penerimaan *output* sebagai hasil respon dari *chatbot* oleh permintaan user atau admin. Selain itu, selama proses perancangan dilakukan pengujian menggunakan metode *whitebox testing* untuk memastikan setiap fungsi internal berjalan sesuai logika yang dirancang dan hal ini telah disetujui oleh supervisor sebagaimana yang telah tercantum dalam Lampiran 10.

Secara keseluruhan, pelaksanaan magang ini berhasil menghasilkan kerangka kerja fungsional Platform AI Reporting yang mencakup keempat fitur unggulan dan pondasi teknis untuk pengembangan lebih lanjut. Meskipun beberapa area masih memerlukan penyempurnaan, terutama terkait optimasi performa, validasi kueri yang dihasilkan LLM, dan pengelolaan OCR pada dokumen bermacam-macam format, hasil yang dicapai menunjukkan kesiapan konsep untuk tahap uji coba pengguna dan integrasi ke lingkungan produksi pada sistem TI yang dimiliki oleh perusahaan klien dari perusahaan AI.DECE.