ASISTEN *GENAI* UNTUK REKOMENDASI TUJUAN MAGANG



Disusun Oleh:

N a m a : Ryan Pradipta Wicaksono
NIM : 22523107

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
2026

HALAMAN PENGESAHAN DOSEN PEMBIMBING

ASISTEN *GENAI* UNTUK REKOMENDASI TUJUAN MAGANG

TUGAS AKHIR

Disusun Oleh:

N a m a : Ryan Pradipta Wicaksono
NIM : 22523107

Yogyakarta, 4 Juli 2026

Pembimbing,

(Dr. Nur Wijayaning Rahayu, S.Kom., M.Cs.)

HALAMAN PENGESAHAN DOSEN PENGUJI

ASISTEN *GENAI* UNTUK REKOMENDASI TUJUAN MAGANG

TUGAS AKHIR

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

Yogyakarta, 30 Juli 2026

Tim Penguji

Dr. Nur Wijayaning Rahayu, S.Kom., M.Cs.

Anggota 1

Ir. Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D., IPM., ASEAN.Eng

Anggota 2

Sri Mulyati, S.Kom., M.Kom.

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

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

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Ryan Pradipta Wicaksono

NIM : 22523107

Tugas akhir dengan judul:

ASISTEN *GENAI* UNTUK REKOMENDASI TUJUAN MAGANG

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

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

Yogyakarta, 4 Juli 2026





(Ryan Pradipta Wicaksono)

HALAMAN PERSEMBAHAN

Segala puji dan syukur saya panjatkan kepada Allah SWT atas segala rahmat, kekuatan, dan kemudahan yang senantiasa diberikan sepanjang perjalanan menyelesaikan penelitian ini.

Karya ini saya persembahkan terutama kepada kedua orang tua saya, yang tak pernah berhenti mendoakan, mendukung, dan mengingatkan saya untuk terus berjuang hingga titik ini. Setiap pengorbanan, kasih sayang, dan kepercayaan yang diberikan menjadi alasan utama saya untuk terus melangkah dan menyelesaikan tugas akhir ini dengan sebaik-baiknya. Kakak saya, yang selalu memberikan dukungan dan nasihat di setiap langkah perjalanan saya. Keluarga besar saya, atas dukungan semangat yang selalu diberikan.

Ibu Dr. Nur Wijayaning Rahayu, S.Kom., M.Cs., selaku dosen pembimbing, atas bimbingan, arahan, kesabaran, dan waktu yang telah diberikan selama proses penyusunan tugas akhir ini. Seluruh dosen program studi informatika Universitas Islam Indonesia, atas ilmu dan bekal yang telah diberikan selama masa perkuliahan. Sahabat dan rekan-rekan seperjuangan, yang telah berbagi diskusi, semangat, dan bantuan di masa-masa sulit penyusunan tugas akhir ini. Serta seluruh pihak yang tidak dapat disebutkan satu per satu, yang telah turut membantu dan mendoakan kelancaran penelitian ini.

Semoga karya ini dapat memberikan manfaat, tidak hanya bagi saya pribadi, tetapi juga bagi siapa pun yang membacanya.

HALAMAN MOTO

Jeda juga merupakan bagian dari sebuah perjalanan.

KATA PENGANTAR

Puji dan syukur penulis panjatkan kehadirat Allah SWT atas segala rahmat, karunia, dan kemudahan yang telah diberikan, sehingga penulis dapat menyelesaikan penyusunan laporan tugas akhir yang berjudul "Asisten GenAI untuk Rekomendasi Tujuan Magang" ini dengan baik. Dalam proses penyusunan tugas akhir ini, penulis menghadapi berbagai tantangan, mulai dari keterbatasan spesifikasi perangkat keras dalam menjalankan model bahasa secara lokal, penyesuaian arsitektur sistem agar dapat bekerja optimal dengan sumber data eksternal, hingga proses iteratif dalam merancang rekayasa *prompt* yang tepat agar sistem dapat menghasilkan rekomendasi yang akurat dan minim kesalahan. Seluruh tantangan tersebut menjadi pembelajaran berharga bagi penulis dalam memahami penerapan kecerdasan buatan secara lebih mendalam.

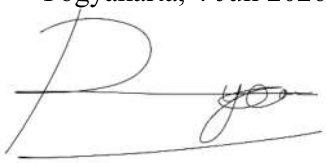
Pada kesempatan ini, penulis ingin menyampaikan ucapan terima kasih yang sebesar-besarnya kepada:

1. Allah SWT, atas segala kemudahan dan kelancaran yang diberikan sepanjang proses penelitian ini.
2. Kedua orang tua dan keluarga penulis, atas doa, dukungan, dan kesabaran yang tiada henti.
3. Ibu Dr. Nur Wijayaning Rahayu, S.Kom., M.Cs., selaku dosen pembimbing, atas segala bimbingan, arahan, masukan, dan waktu yang telah diberikan selama proses penyusunan tugas akhir ini.
4. Seluruh dosen dan staf Program Studi Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia, atas ilmu dan bekal yang telah diberikan selama masa perkuliahan.
5. Rekan-rekan mahasiswa yang telah bersedia menjadi responden survei maupun partisipan pengujian aplikasi dalam penelitian ini.
6. Teman-teman Hore Team, atas kebersamaan, canda tawa, dan dukungannya
7. Sahabat dan rekan-rekan seperjuangan lainnya, atas diskusi, semangat, dan bantuan yang diberikan selama masa penyusunan tugas akhir ini.
8. Serta seluruh pihak yang tidak dapat disebutkan satu per satu, yang telah turut membantu kelancaran penelitian ini.

Penulis menyadari bahwa laporan tugas akhir ini masih jauh dari kesempurnaan. Oleh karena itu, kritik dan saran yang membangun sangat penulis harapkan demi perbaikan di masa mendatang. Penulis berharap hasil penelitian ini dapat memberikan manfaat, baik sebagai

referensi akademik maupun sebagai kontribusi nyata bagi pengembangan sistem rekomendasi magang berbasis kecerdasan buatan selanjutnya.

Yogyakarta, 4 Juli 2026



(Ryan Pradipta Wicaksono)

SARI

Magang merupakan bentuk pembelajaran berbasis pengalaman yang banyak diterapkan di pendidikan tinggi untuk menjembatani kesenjangan antara kompetensi lulusan dan kebutuhan industri. Namun, mahasiswa sering mengalami kesulitan menemukan tempat magang yang sesuai dengan preferensi dan keahliannya, hal ini dikonfirmasi oleh survei terhadap responden. Penelitian ini bertujuan untuk mengembangkan asisten berbasis *Generative AI* (GenAI) yang mampu menganalisis profil mahasiswa dari propmt yang dimasukkan dan memberikan rekomendasi magang yang sesuai preferensi.

Sistem ini dikembangkan menggunakan metode *Rapid Application Development* (RAD) dengan arsitektur *Retrieval-Augmented Generation* (RAG), proses pengambilan data dilakukan melalui *Internship API* dari fantastic.jobs. Model LLM Qwen2.5:7B dijalankan secara lokal via Ollama, dengan rekayasa *prompt* dua tahap dan klasifikasi intensi. *Backend* dibangun dengan python dan flask, antarmuka dengan HTML5, JavaScript, dan TailwindCSS. Evaluasi dilakukan melalui *black-box testing* (18 skenario) dan pengujian penerimaan pengguna terhadap lima partisipan.

Hasil pengujian fungsional mencapai 77,8 & keberhasilan. Penerimaan pengguna mendapat rata-rata positif, terutama pada kualitas bahasa dan antarmuka, sementara waktu respons menjadi kendala akibat *trade-off* dari menggunakan LLM lokal. Penelitian ini juga mendokumentasikan fenomena ketidakcocokan kata kunci antara LLM dan API eksternal sebagai kontribusi bagi pengembangsistem sejenis.

Kata kunci: *Generative AI*, RAG, *Large Language Model*, rekomendasi magang.

GLOSARIUM

Environment variable	variabel yang disimpan di luar kode sumber untuk menampung konfigurasi atau data sensitif, seperti kunci API.
Intent classification	proses mengelompokkan maksud dari masukan pengguna ke dalam kategori tertentu agar sistem dapat merespons secara tepat.
Offloading	mekanisme pemindahan sebagian beban komputasi dari VRAM ke RAM utama ketika kapasitas memori GPU tidak mencukupi.
On-premise	model penerapan sistem yang dijalankan secara lokal pada perangkat keras milik sendiri, bukan pada layanan *cloud* pihak ketiga.
Open weight	istilah untuk model yang hasil pelatihannya dipublikasikan secara terbuka, sehingga dapat diunduh dan dijalankan secara lokal tanpa harus melalui layanan API berbayar dari penyedia model.
Parametric memory	pengetahuan yang tertanam dalam parameter model bahasa hasil dari proses pelatihan.
Non-parametric memory	basis pengetahuan eksternal yang dapat diperbarui secara terpisah tanpa perlu melatih ulang model.
Server-side session	metode penyimpanan data sesi pengguna pada sisi server, bukan pada perangkat klien.
VRAM (Video Random Access Memory)	memori khusus pada kartu grafis (GPU) yang digunakan untuk menyimpan data selama proses komputasi.
Zero-shot prompting	teknik memberi instruksi kepada model bahasa tanpa menyertakan contoh demonstrasi.
Few-shot prompting	teknik menyertakan beberapa contoh demonstrasi dalam instruksi agar model memahami pola tugas yang diinginkan.

DAFTAR ISI

HALAMAN JUDUL i
HALAMAN PENGESAHAN DOSEN PEMBIMBING ii
HALAMAN PENGESAHAN DOSEN PENGUJI iii
HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR iv
HALAMAN PERSEMBAHAN v
HALAMAN MOTO vi
KATA PENGANTAR vii
SARI ix
GLOSARIUM x
DAFTAR ISI xi
DAFTAR TABEL xiii
DAFTAR GAMBAR xiv
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan masalah dan pertanyaan penelitian 2
1.3 Batasan masalah dan lingkup penelitian 2
1.4 Tujuan dan manfaat penelitian 2
1.4.1 Tujuan penelitian 3
1.4.2 Manfaat penelitian 3
1.5 Sistematika penulisan 3
BAB II TINJAUAN PUSTAKA 5
2.1 Landasan teori 5
2.1.1 Program Magang 5
2.1.2 *Generative* AI dan *large language model* 6
2.1.3 Retrieval-augmented generation (RAG) 7
2.1.4 Prompt engineering pada aplikasi LLM 8
2.1.5 Pengujian dan evaluasi aplikasi 9
2.2 Penelitian terdahulu 9
BAB III METODOLOGI PENELITIAN 11
3.1 Tahap perencanaan kebutuhan (*Requirenment planning*) 11
3.1.1 Desain survei dan kriteria responden 11
3.1.2 Instrumen pengumpulan kebutuhan 12
3.1.3 Hasil pengumpulan kebutuhan 13
3.2 Tahap perancangan pengguna 16
3.2.1 Prototipe 16
3.2.2 *Test* (rencana pengujian prototipe) 20
3.2.3 *Refine* 21
3.3 Tahap konstruksi (*Construction*) 21
3.3.1 Lingkungan dan teknologi implementasi 21
3.3.2 Alur rekayasa prompt 23
3.3.3 Kriteria evaluasi pemilihan model LLM 23
3.3.4 Pendekatan iteratif pada pengembangan 23
3.3.5 Potensi kendala data eksternal 24
3.4 Tahap peralihan (*cutover*) 24
3.4.1 Desain pengujian fungsional (*black-box testing*) 24
3.4.2 Desain pengujian penerimaan pengguna 25
BAB IV HASIL DAN PEMBAHASAN 27

4.1 Hasil perancangan pengguna (User design) 27
4.1.1 Prototipe yang diujikan (*prototype*) 27
4.1.2 Hasil pengujian prototipe (*test*) 27
4.1.3 Hasil penyempurnaan rancangan (*refine*) 28
4.2 Hasil konstruksi (*Construction*) 28
4.2.1 Realisasi lingkungan dan *session handling* 28
4.2.2 Implementasi RAG dan rekayasa prompt pada sistem 30
4.2.3 Hasil eksperimen pemilihan model LLM 39
4.2.4 Hasil iterasi pengembangan dan final aplikasi 39
4.2.5 Hasil analisis kendala data API eksternal 42
4.3 Hasil tahap cutover / peralihan 45
4.3.1 Hasil pengujian (*black-box testing*) 45
4.3.2 Hasil pengujian penerimaan pengguna 47
BAB V KESIMPULAN DAN SARAN 49
5.1 Kesimpulan 49
5.1.1 Aspek perancangan dan pembangunan aplikasi 49
5.1.2 Aspek efektivitas fungsional dan penerimaan pengguna 49
5.1.3 Kontribusi penelitian 50
5.2 Saran 50
DAFTAR PUSTAKA 51
LAMPIRAN 54

DAFTAR TABEL

Tabel 2.1 Perbandingan dengan penelitian terdahulu 9
Tabel 3.1 Distribusi penilaian tingkat kepentingan fitur aplikasi 15
Tabel 3.2 Matriks uji *black-box testing* 25
Tabel 4.1 Perbandingan rancangan sebelum dan sesudah penyempurnaan 28
Tabel 4.2 Hasil pengujian dan perbandingan model LLM pada sistem 39
Tabel 4.3 Hasil iterasi pengembangan aplikasi 39
Tabel 4.4 Hasil pengujian variasi penamaan lokasi pada API 43
Tabel 4.5 Hasil rekomendasi dan analisis 43
Tabel 4.6 Hasil *black-box testing* aplikasi 45
Tabel 4.7 Konsistensi output multi-eksekusi (n=3) 47
Tabel 4.8 Hasil pengujian penerimaan pengguna 47

DAFTAR GAMBAR

Gambar 3.1 Diagram RAD 11
Gambar 3.2 Analisis platform pencarian informasi magang 13
Gambar 3.3 Analisis tantangan responden dalam mencari magang. 14
Gambar 3.4 Analisis faktor penyebab kesulitan magang..... 14
Gambar 3.5 Use case diagram 17
Gambar 3.6 Alur RAG sistem..... 18
Gambar 3.7 Desain halaman awal aplikasi 19
Gambar 3.8 Desain halaman hasil rekomendasi 20
Gambar 4.1 Konfigurasi *session* 29
Gambar 4.2 Potongan kode penerimaan masukan pengguna 30
Gambar 4.3 Alur klasifikasi intensi dua jalur 31
Gambar 4.4 Fungsi classify_intent()..... 32
Gambar 4.5 Potongan kode perutean dua jalur..... 32
Gambar 4.6 Instruksi (*prompt*) konteks pada jalur CHAT_BIASA 33
Gambar 4.7 Instruksi ekstraksi parameter profil pengguna..... 34
Gambar 4.8 Kode extract_params() dan penaganan hasil 35
Gambar 4.9 Fungsi call_api() dan location_fix()..... 36
Gambar 4.10 Potongan kode penanganan hasil pencarian jika kosong 36
Gambar 4.11 Potongan kode penyimpanan sesi dan penyusunan konteks lowongan 36
Gambar 4.12 Data lowongan mentah hasil dari API 37
Gambar 4.13 Instruksi untuk rekomendasi (final_prompt)..... 38
Gambar 4.14 Potongan kode *generate respons* akhir 38
Gambar 4.15 Tampilan awal final aplikasi 41
Gambar 4.16 Tampilan hasil rekomendasi magang..... 42

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dunia saat ini berada pada revolusi industri yang bergantung pada otomatisasi dan kecerdasan buatan, sehingga memicu kebutuhan global akan tenaga kerja berliterasi teknologi tinggi. Terdapat tantangan nyata berupa kesenjangan talenta di mana lulusan perguruan tinggi sering kali belum siap menghadapi dinamika industri yang berubah cepat. Guna menjembatani kesenjangan tersebut, perguruan tinggi umumnya menempatkan magang sebagai bentuk pembelajaran berbasis pengalaman yang dirancang untuk memberikan pengalaman nyata, keterampilan relevan, dan jaringan profesional demi meningkatkan daya saing lulusan(Putri dkk., 2025), serta memfasilitasi mahasiswa dalam menerapkan teori perkuliahan ke lingkungan kerja (Ufia dkk., 2024). Namun, banyaknya jumlah peserta melahirkan tantangan logistik yang besar dalam memetakan mahasiswa ke posisi yang tepat agar potensi SDM ini dapat terserap optimal oleh industri. Jumlah peserta yang besar ini juga membuat pendampingan secara personal sulit dilakukan, sebab dosen pembimbing tidak mungkin memahami kualifikasi setiap mahasiswa secara cepat dan menyeluruh.

Tantangan pemetaan tersebut diperparah oleh fenomena *information overload* akibat tersebarnya lowongan magang di berbagai platform digital, yang membuat mahasiswa bingung dan tidak efisien dalam mencari secara manual (Kristiawan dkk., 2025). Ditambah keterbatasan dalam melakukan *self-assessment (hard* dan *soft skills*), mahasiswa cenderung memilih tempat magang hanya berdasarkan popularitas perusahaan atau ikut-ikutan teman, sehingga memicu ketidakpuasan akibat tugas yang tidak relevan atau kualifikasi yang tidak sesuai ekspektasi industri. Sebagai solusi untuk masalah ini, diperlukan teknologi yang mampu bertindak sebagai asisten karir yang dapat menerjemahkan keinginan dari pengguna. Oleh karena itu, penelitian ini mengintegrasikan teknologi GenAI, sebab perkembangan Generative *Artificial Intelligence* (GenAI) khususnya *Large Language Models* (LLM) membawa revolusi besar dalam pemrosesan bahasa manusia (Parker dkk., 2025). Melampaui metode pencocokan kata kunci *(keyword matching*) tradisional, GenAI mampu menganalisa *prompt* naratif mahasiswa untuk bertindak sebagai penasihat karir yang mengevaluasi dan memberikan rekomendasi magang yang paling relevan, serta menyertakan alasan logis demi meningkatkan kepercayaan diri mahasiswa dalam mengambil keputusan.

1.2 Rumusan masalah dan pertanyaan penelitian

Berdasarkan latar belakang tersebut, permasalahan utama penelitian ini adalah mahasiswa kesulitan menemukan posisi magang yang sesuai dengan kompetensinya, baik karena kurang memahami keterkaitan antara hard skills yang dimiliki dengan kualifikasi lowongan, maupun karena banyaknya lowongan yang tersebar di berbagai platform (information overload). Kondisi ini diperberat oleh terbatasnya pendampingan yang bersifat personal, sehingga dibutuhkan solusi teknologi yang mampu memberikan rekomendasi magang secara individual dan sesuai dengan kualifikasi masing-masing mahasiswa. Berdasarkan rumusan masalah tersebut, pertanyaan penelitian ini adalah:

1. Bagaimana cara mengembangkan asisten berbasis GenAI yang mampu menganalisis kualifikasi mahasiswa untuk memberikan rekomendasi tempat magang yang relevan?
2. Sejauh mana asisten berbasis GenAI tersebut dapat membantu mahasiswa mengurangi kebingungan dalam memilih tempat magang?

1.3 Batasan masalah dan lingkup penelitian

Untuk menjaga berjalannya penelitian, pengembangan aplikasi asisten cerdas ini dibatasi pada empat hal berikut:

1. Sumber data. Rekomendasi dihasilkan berdasarkan dua sumber, yaitu data kualifikasi yang dimasukkan mahasiswa (*prompt*), dan dataset lowongan magang dari internship API fantastic.jobs. Kualitas rekomendasi bergantung langsung pada kelengkapan input pengguna dan data lowongan yang ada.
2. LLM yang digunakan. Sistem menggunakan LLM lokal untuk proses rekomendasi dan pencocokan antara profil mahasiswa dan lowongan yang tersedia.
3. Batasan fungsionalitas. Penelitian ini hanya mencakup mekanisme analisis kualifikasi dan pemberian rekomendasi magang.
4. Cakupan program magang**.** Sistem dirancang untuk pencarian lowongan magang secara umum dan tidak terikat pada program magang tertentu. Lowongan yang direkomendasikan berasal sepenuhnya dari API eksternal.

1.4 Tujuan dan manfaat penelitian

Penetapan arah penelitian ini didasarkan untuk menghadirkan solusi teknologi nyata bagi mahasiswa sekaligus memberikan kontribusi ilmiah. Target pencapaian serta dampak positif yang diharapkan dari hasil pengembangan sistem ini dirinci sebagai berikut.

1.4.1 Tujuan penelitian

Tujuan dari penelitian ini yaitu:

1. Mengembangkan sistem asisten cerdas, membangun sebuah prototipe asisten berbasis *Generative* AI yang mampu melakukan ekstraksi informasi mahasiswa secara otomatis guna memetakan sesuai kualifikasi.
2. Integrasi data lowongan yang terstruktur, mengintegrasikan sistem dengan internship API dari fantastic.jobs untuk menyediakan akses ke dataset lowongan magang yang luas dan terkini secara efisien.
3. Memberikan rekomendasi yang personal, menghasilkan sebuah model untuk pemberian rekomendasi tujuan magang yang memiliki relevan dengan profil mahasiswa dan persyaratan industri.

1.4.2 Manfaat penelitian

Hasil dari penelitian ini diharapkan dapat memberikan kontribusi nyata, yaitu :

1. Efisiensi pencarian, menghemat waktu mahasiswa dalam menyaring lowongan magang yang tidak relevan.
2. Peningkatan kesesuaian kualifikasi, membantu mahasiswa menemukan posisi yang sesuai dengan keahliannya, sehingga proses magang menjadi optimal.
3. Penyediaan alat bantu konsultasi, menjadi alat pendukung bagi dosen pembimbing dalam memberikan saran karier bagi setiap mahasiswa.
4. Referensi implementasi GenAI, memberikan bukti nyata mengenai efektivitas penggunaan *Large Language Models* dalam bidang sistem rekomendasi.

1.5 Sistematika penulisan

Laporan penelitian ini disusun dalam lima bab, Bab I menguraikan latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, serta sistematika penulisan sebagai dasar penelitian. Bab II membahas landasan teori yang meliputi program magang, perkembangan *generative* AI dan *large language model*, arsitektur *retrieval-augmented generation* (RAG), teknik prompt engineering, serta pendekatan pengujian dan evaluasi aplikasi seperti *black-box testing*, dilengkapi tinjauan penelitian terdahulu. Bab III menjelaskan metode pengembangan sistem menggunakan *rapid application development* (RAD) melalui empat tahap, yaitu perencanaan kebutuhan, desain pengguna, konstruksi, dan *cutover* yang mencakup survei pengguna, pemodelan sistem, rancangan antarmuka, rekayasa *prompt*, pemilihan model LLM, hingga desain pengujian fungsional dan penerimaan pengguna. Bab IV menyajikan hasil dari tahapan RAD yang telah dirancang pada Bab III, meliputi, validasi pemodelan sistem dan

antarmuka, realisasi implementasi teknis, hasil eksperimen pemilihan LLM, serta hasil pengujian fungsional dan penerimaan pengguna. Terakhir, Bab V memuat kesimpulan penelitian berdasarkan rumusan masalah yang telah dijawab, serta saran untuk pengembang selanjutnya.

BAB II
TINJAUAN PUSTAKA

Bab ini membahas landasan teoritis yang menjadi pondasi teori penelitian, tinjauan terhadap penelitian terdahulu yang relevan dengan sistem yang dikembangkan. Landasan teori mencakup konteks program magang, perkembangan *Generative AI* dan *Large Language Model*, arsitektur *Retrieval-Augmented Generation* (RAG), teknik *prompt engineering*, metode pengembangan *Rapid Application Development* (RAD), serta pendekatan pengujian dan evaluasi aplikasi. Selanjutnya, tinjauan penelitian terdahulu menyajikan ulasan ringkas terhadap karya-karya yang membahas sistem rekomendasi karir dan magang

2.1 Landasan teori

Landasan teori ini disusun untuk memperkuat alur pengembangan aplikasi dari tiga sudut pandang, masalah yang diangkat (program magang), solusi teknologi (*Generative AI*, RAG, dan *prompt engineering*), serta pengujian kesistem. Hubungan antarkomponen tersebut diuraikan pada sub-bab berikut.

2.1.1 Program Magang

Magang merupakan salah satu bentuk pembelajaran berbasis pengalaman yang dirancang untuk menjembatani teori akademik dengan praktik nyata di dunia industri. Tujuan utama dari program magang adalah memberikan pengalaman kerja yang relevan, keterampilan praktis, serta jaringan profesional kepada mahasiswa sehingga lulusan perguruan tinggi memiliki daya saing yang lebih tinggi saat memasuki pasar kerja (Putri dkk., 2025). Selain itu, magang dipandang sebagai sarana efektif untuk meningkatkan kompetensi *hard skill* dan *soft skill* mahasiswa melalui penerapan langsung teori perkuliahan di lingkungan kerja (Ufia dkk., 2024). Kompetensi yang dibangun melalui pengalaman magang ini mencakup kemampuan teknis sesuai bidang studi maupun kemampuan non-teknis seperti komunikasi profesional, kerja tim, manajemen waktu, dan adaptasi terhadap budaya organisasi.

Pentingnya magang, juga tercermin dari adanya peraturan yang mengaturnya. Salah satu bentuknya adalah kebijakan Merdeka Belajar Kampus Merdeka (MBKM) yang diinisiasi oleh Kementerian Pendidikan dan Kebudayaan melalui Peraturan Standar Nasional Pendidikan Tinggi Nomor 3 Tahun 2020 yang menempatkan magang sebagai salah satu kegiatan pembelajaran di luar program studi yang diakui sebagai bagian dari beban studi mahasiswa

(Putri dkk., 2025). Meskipun demikian, penelitian ini tidak membatasi cakupannya pada program magang tertentu, melainkan magang secara umum sebagai kebutuhan mahasiswa.

Namun, perkembangan program magang yang masif tidak serta-merta diiringi oleh kemudahan akses mahasiswa terhadap informasi lowongan yang relevan. Mahasiswa sering kali menghadapi tantangan mendasar berupa fenomena *information overload* akibat tersebarnya informasi lowongan magang di berbagai platform digital, yang menyebabkan proses pencarian menjadi tidak efisien dan tidak terstruktur (Kristiawan dkk., 2025). Kondisi ini menjadi salah satu motivasi utama dikembangkannya sistem yang dapat membantu mahasiswa menavigasi pencarian magang berdasarkan profil dan preferensi.

2.1.2 *Generative* AI dan *large language model*

Generative Artificial Intelligence (GenAI) merupakan jenis kecerdasan buatan yang mampu menciptakan konten baru, baik teks, gambar, kode pemrograman, maupun media lainnya, dengan mempelajari pola dari data yang sudah ada (Parker dkk., 2025). Berbeda dengan pendekatan kecerdasan buatan klasik yang berfokus pada klasifikasi atau prediksi, GenAI menghasilkan luaran yang bersifat kreatif dan kontekstual sesuai dengan instruksi yang diberikan kepada model. Perkembangan GenAI dalam beberapa tahun terakhir membawa transformasi signifikan pada berbagai sektor industri, termasuk personalisasi layanan komersial yang sebelumnya bergantung pada *rule-based system* atau pencocokan kata kunci sederhana (Parker dkk., 2025).

Perkembangan LLM ini ditandai dengan rilisnya berbagai model *open weight* yang memungkinkan *deployment* secara lokal tanpa harus bergantung pada layanan *cloud* berbayar. Namun, LLM memiliki dua kelemahan utama. Pertama, pengetahuan model ini bersifat statis karena hanya terpaku pada data yang digunakan saat pelatihan, sehingga tidak mampu mengakses informasi yang baru atau berubah dinamis. Kedua, LLM rentan terhadap fenomena halusinasi, yaitu kondisi ketika model menghasilkan informasi yang terdengar meyakinkan namun secara faktual salah atau fiktif, karena keterbatasan mekanisme verifikasi terhadap sumber pengetahuan (Lewis dkk., 2020). Kedua keterbatasan inilah yang memicu lahirnya pendekatan baru yang mengintegrasikan kemampuan generatif LLM dengan basis pengetahuan eksternal yang dapat diperbarui.

2.1.3 Retrieval-augmented generation (RAG)

Retrieval-Augmented Generation (RAG) adalah pendekatan arsitektur yang diperkenalkan oleh Lewis et al. (Lewis dkk., 2020). untuk mengatasi kelemahan LLM dalam mengakses informasi eksternal yang terus berubah. Dalam formulasi awalnya, RAG mengombinasikan dua komponen utama, yaitu *parametric memory* berupa pengetahuan yang sudah tertanam dalam parameter LLM, dan *non-parametric memory* berupa kumpulan dokumen eksternal yang dapat terus diperbarui tanpa perlu melatih ulang model. Pada penelitian yang dilakukan Lewis et al., pengimplementasian RAG menggunakan *dense vector retrieval* dengan indeks Wikipedia sebagai sumber pengetahuan eksternal (Lewis dkk., 2020).

Sejak pertama kali diperkenalkan, penelitian tentang RAG berkembang pesat dan melahirkan berbagai variasi arsitektur. Gao et al. (Gao dkk., 2024) dalam studi survei komprehensifnya mengelompokkan RAG ke dalam tiga pendekatan utama. Pertama, *Naive RAG* yang menggunakan metode pencarian-pembacaan sederhana. Kedua, *Advanced RAG* yang memberikan optimasi tambahan sebelum dan sesudah pencarian, seperti penulisan ulang kueri (*query rewriting*) dan pengurutan ulang hasil. Ketiga, *Modular RAG* yang memecah seluruh alur proses menjadi modul-modul yang dapat diatur secara fleksibel sesuai kebutuhan aplikasi (Gao dkk., 2024). Berdasarkan mekanisme pencariannya, RAG juga diklasifikasikan menjadi *Dense RAG* yang mengandalkan kesamaan semantik berbasis vektor, *Sparse RAG* yang menggunakan metode pencarian tradisional seperti BM25, serta *Hybrid RAG* yang mengkombinasikan keduanya (Gao dkk., 2024).

Pendekatan yang lebih baru memperluas kemampuan RAG ke arah *tool-augmented generation* (Schick dkk., 2023). Schick et al. (Schick dkk., 2023) mendemonstrasikan bahwa LLM dapat dilatih untuk menggunakan *external tools* secara *self-supervised*, termasuk pemanggilan API kalkulator, mesin pencari, dan sistem penerjemahan. Konsep *tool-augmented generation* ini menjadi landasan teoritis penting bagi aplikasi yang dikembangkan dalam penelitian ini.

Penerapan RAG pada ranah berbahasa Indonesia juga terus berkembang. Salah satu studi yang relevan adalah KemenkeuGPT yang dikembangkan oleh Febrian dan Figueredo (Febrian & Figueredo, 2024). Studi tersebut membangun asisten LLM untuk menjawab pertanyaan terkait data dan regulasi keuangan di Kementerian Keuangan Republik Indonesia menggunakan kombinasi *framework* LangChain, RAG, *prompt engineering*, dan *fine-tuning*. Keberhasilan KemenkeuGPT menunjukkan bahwa pendekatan RAG dapat diterapkan secara

efektif pada konteks domain spesifik berbahasa Indonesia, sekaligus memvalidasi pendekatan serupa yang diadopsi pada penelitian ini.

2.1.4 Prompt engineering pada aplikasi LLM

Prompt engineering merupakan teknik perancangan instruksi natural yang mengarahkan LLM untuk menghasilkan luaran sesuai kebutuhan tugas tertentu tanpa memerlukan fine-tuning parameter model (Sahoo dkk., 2025). Pendekatan ini menjadi penting dalam penggunaan LLM, karena model yang sudah dilatih sebelumnya dapat disesuaikan untuk beragam tugas hanya dengan mengubah instruksinya. Keunggulan dari *prompt engineering* dibandingkan *fine-tuning* terletak pada fleksibilitas dalam adaptasi tugas baru dan kemudahan iterasi instruksi tanpa harus melatih ulang model.

Sahoo et al. (Sahoo dkk., 2025) dalam *systematic survey* mengkategorikan teknik *prompt engineering* ke dalam beberapa pendekatan utama. *Zero-shot prompting* memberikan instruksi tanpa contoh demonstrasi dan mengandalkan kapabilitas generalisasi model. *Few-shot prompting* menyertakan beberapa contoh demonstrasi di dalam *prompt* untuk membantu model memahami pola tugas yang diinginkan. Selain itu, terdapat teknik-teknik lanjutan seperti *Chain-of-Thought prompting* yang memandu model untuk menjabarkan langkah penalaran sebelum memberikan jawaban akhir, serta *Self-Consistency prompting* yang mengeksekusi beberapa jalur penalaran dan memilih jawaban yang paling konsisten (Sahoo dkk., 2025). Vatsal dan Dubey (Vatsal & Dubey, 2024)memperdalam analisis ini dengan berfokus pada efektivitas berbagai teknik *prompt engineering* untuk beragam tugas pemrosesan bahasa alami (NLP).

Penerapan teknik *prompt engineering* pada sistem berbasis RAG kini tidak lagi terbatas pada satu instruksi tunggal, melainkan telah bergeser ke arah struktur yang lebih kompleks. Dimana LLM digunakan secara berurutan untuk tugas yang berbeda, sering diterapkan pada sistem RAG untuk memisahkan tanggung jawab logis dan meningkatkan kontrol terhadap kualitas luaran (Gao dkk., 2024). Pada pola ini, tahap pertama umumnya berfokus pada ekstraksi parameter terstruktur dari masukan bahasa alami pengguna, sementara tahap kedua berfokus pada sintesis respons akhir setelah data eksternal berhasil diambil melalui mekanisme *retrieval*. Pola dua tahap inilah yang menjadi salah satu desain inti pada sistem yang dikembangkan dalam penelitian ini.

2.1.5 Pengujian dan evaluasi aplikasi

Pengujian perangkat lunak merupakan tahap kritis untuk memastikan bahwa aplikasi yang dikembangkan memenuhi spesifikasi fungsional yang dirancang dan diterima dengan baik oleh pengguna target. Pada penelitian ini, pengujian dilakukan menggunakan kombinasi dua pendekatan, yaitu pengujian fungsional berbasis *black-box testing* dan pengujian penerimaan pengguna menggunakan skala Likert.

Black-box Testing

Black-box testing adalah teknik pengujian perangkat lunak yang berfokus pada eksekusi fungsionalitas sistem tanpa memeriksa struktur kode internalnya, sehingga juga dikenal sebagai *behavioral testing* (Pressman, 2010). Pengujian dilakukan dengan memberikan masukan tertentu pada sistem dan mengamati apakah luaran yang dihasilkan sesuai dengan hasil yang diharapkan. Pendekatan ini umumnya menggunakan matriks uji yang berisi serangkaian skenario pengujian, masing-masing disertai dengan kondisi masukan, hasil yang diharapkan, dan status akhir.

Skala Likert

Skala Likert adalah salah satu instrumen pengukuran yang umum digunakan untuk mengukur sikap, opini, dan persepsi seseorang terhadap suatu objek atau fenomena. Pranatawijaya dkk (Pranatawijaya dkk., 2019) menjelaskan bahwa skala Likert menggunakan rentang lima tingkat (1 hingga 5). Dalam penelitian pengembangan perangkat lunak, skala Likert sering digunakan untuk mengukur tingkat penerimaan pengguna terhadap berbagai aspek aplikasi, seperti kemudahan penggunaan, kualitas luaran, dan niat untuk menggunakan kembali.

2.2 Penelitian terdahulu

Sistem rekomendasi karir dan pekerjaan telah menjadi area riset aktif untuk mengatasi *information overload* pada platform rekrutmen digital. Pendekatan yang umum digunakan dalam literatur mencakup *content-based filtering* (CBF), *collaborative filtering* (CF), serta pendekatan *hybrid* yang mengkombinasikan keduanya. Setiap pendekatan memiliki karakteristik, jenis data yang diproses, serta batasan tertentu dalam penerapannya. Sub-bab ini mengulas empat penelitian terdahulu yang relevan dengan ranah rekomendasi karir dan magang. Perbandingan keempat penelitian terdahulu ini dirangkum pada Tabel 2.1.

Tabel 2.1 Perbandingan dengan penelitian terdahulu

Penulis	Pendekatan Teknis	Limitasi / *Gap*

Kristiawan dkk., 2025	Algoritma *A-Star* (*graph search*) dengan data internal kampus	Tidak menggunakan LLM; tidak menangani masukan bahasa alami; data terbatas pada satu institusi
Poncio, 2024	*Systematic review* berbagai teknik rekomendasi	Bersifat tinjauan, tidak menghasilkan implementasi sistem
Mulay dkk., 2022	*Content-based filtering*, *collaborative filtering*, dan *hybrid filtering* dengan dataset statis	Belum mengeksplorasi LLM modern; menggunakan dataset statis
Rashid dkk., 2022	*Content-based filtering* dengan *web scraping*	Memilih *career path* secara manual; tidak menangani profil naratif
Penelitian ini	LLM lokal (Qwen2.5:7B) + RAG dengan *retrieval* berbasis API + klasifikasi intensi dua jalur + API fantastic.jobs (*live data*)	Memasukkan *prompt* untuk mencari lowongan magang

Berdasarkan analisis pada Tabel 2.1, teridentifikasi tiga posisi pengembangan yang menjadi kontribusi penelitian ini:

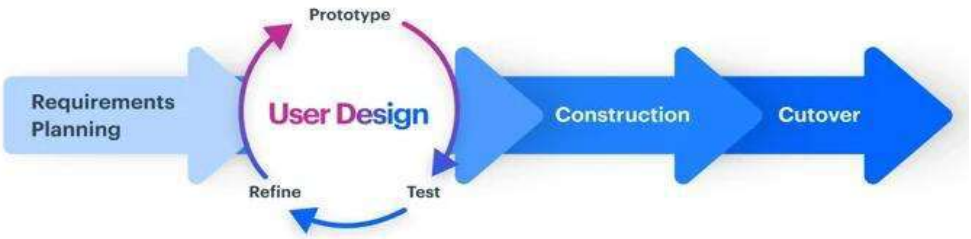
Pertama, pemanfaatan LLM modern untuk rekomendasi magang berbahasa Indonesia. Mayoritas penelitian terdahulu menggunakan pendekatan tradisional (CBF, CF, *hybrid*) yang mengandalkan kategorisasi *skill* terstruktur dan tidak mampu menangani masukan naratif berbahasa alami. Penelitian ini mengisi celah tersebut dengan memanfaatkan LLM modern (Qwen2.5:7B) yang dijalankan secara lokal untuk memproses profil pengguna dalam bentuk teks naratif Bahasa Indonesia.

Kedua, arsitektur RAG dengan *retrieval* berbasis API eksternal. Implementasi RAG yang dilaporkan pada penelitian terdahulu umumnya menggunakan indeks dokumen lokal dengan *dense vector retrieval* (Lewis dkk., 2020). Dalam penelitian ini, arsitektur RAG dirancang agar proses *retrieval* dapat didelegasikan ke API eksternal, mengikuti prinsip kerja *tool-augmented generation* (Schick dkk., 2023). Pendekatan ini memungkinkan sistem selalu memperoleh data lowongan yang *up-to-date* tanpa perlu memelihara basis data lokal.

Ketiga, dokumentasi faktual limitasi pengembangan. Melalui hasil pengujian, penelitian ini berhasil mengidentifikasi adanya ketidakcocokan antara pendekatan berbasis makna pada LLM dan pendekatan kata kaku pada API. Dokumentasi terhadap fenomena ketidakcocokan ini memberikan kontribusi penting bagi pengembangan desain sistem rekomendasi serupa di masa mendatang yang mengadopsi pola arsitektur RAG dengan API komersial sebagai *retrieval layer*.

BAB III
METODOLOGI PENELITIAN

Metode yang digunakan untuk pengembangan ini adalah Rapid Application Development (RAD). Pemilihan metode ini didasarkan pada karakteristiknya yang memungkinkan integrasi masukan atau umpan balik pengguna dilakukan secara berkelanjutan sepanjang siklus pengembangan. Tahapan utama dalam metode RAD yang digunakan meliputi empat fase utama: Perencanaan Kebutuhan (*Requirements Planning*), Desain Pengguna (*User* Design), Konstruksi (*Construction*), dan Peralihan (*Cutover*).



Gambar 3.1 Diagram RAD

Faktor yang krusial dari alur RAD ini adalah keberadaan iterasi (*loop*) pada fase Desain Pengguna. Iterasi ini memastikan bahwa setiap prototipe yang dihasilkan di fase Desain Pengguna diuji terlebih dahulu, kemudian dikembangkan kodenya pada fase Konstruksi. Apabila ditemukan umpan balik yang signifikan dari pengguna, proses desain akan diulang (*Refine*) sebelum kembali memasuki fase implementasi. Karakteristik iteratif inilah yang menjadi alasan utama RAD dipilih sebagai metode pengembangan aplikasi pada penelitian ini.

3.1 Tahap perencanaan kebutuhan (*Requirenment planning*)

Pada tahap awal metodologi RAD ini, fokus utama dilakukan pada penggalian data untuk memahami kebutuhan riil dari calon pengguna. Proses perancangan strategi pengumpulan data, instrumen yang digunakan, serta hasil yang diperoleh dijabarkan pada sub-bab berikut.

3.1.1 Desain survei dan kriteria responden

Proses pengumpulan kebutuhan pengguna diawali dengan merancang dan menyebarkan survei secara daring. Pengumpulan data ini dilaksanakan pada bulan Desember 2025 dengan

menggunakan kuesioner digital (Google Form) untuk mengidentifikasi kendala nyata yang dihadapi pengguna.

Kriteria responden (partisipan) untuk survei ini difokuskan pada pengguna yang relevan dengan ekosistem program magang. Teknik pengambilan sampel yang digunakan adalah *convenience sampling*, yaitu responden diperoleh berdasarkan kesediaan mengisi kuesioner yang disebarkan secara terbuka. Teknik ini dipilih karena survei bersifat eksploratif untuk menggali kebutuhan pengguna sebagai dasar perancangan sistem. Adapun kriteria responden yang dituju dalam pengumpulan data ini meliputi:

1. Mahasiswa aktif lintas jurusan yang sedang berada dalam tahapan mencari atau mempersiapkan diri untuk mengikuti magang.
2. Mahasiswa atau alumni lintas jurusan yang telah menyelesaikan magang, sehingga dapat memberikan perspektif berdasarkan pengalaman nyata.

3.1.2 Instrumen pengumpulan kebutuhan

Instrumen kuesioner yang disebarkan dibagi menjadi dua bagian untuk menangkap kebutuhan pengguna:

1. Identifikasi tantangan pengguna (*multiple-response checklist*): Bagian pertama kuesioner bertujuan untuk memetakan masalah dan tantangan yang dihadapi oleh mahasiswa ketika mencari tempat magang.
2. Pengukuran urgensi fitur (skala Likert): Bagian kedua kuesioner digunakan untuk mengukur tingkat urgensi dari fitur-fitur potensial yang akan dikembangkan dalam sistem. Pengukuran preferensi responden dilakukan dengan menggunakan skala Likert. Melalui skala ini, responden diminta untuk memberikan penilaian terhadap masing-masing fitur berdasarkan tingkat kepentingan (misalnya dari skala 1 yang berarti "tidak penting" hingga skala 5 yang berarti "sangat penting"). Terdapat lima kandidat fitur awal yang dinilai penting dalam proses pencarian lowongan magang untuk dievaluasi oleh responden. Kelima kandidat fitur ini dirumuskan berdasarkan observasi terhadap beberapa lowongan magang. Kelima fitur ini. yaitu:
    a. Informasi gaji/insentif: Fitur yang menampilkan gaji atau uang saku diberikan oleh pihak perusahaan.
    b. Mode kerja (WFH/WFO): Fitur yang menampilkan kejelasan sistem kerja, apakah bersifat *work from home* (*remote*), *work from office* (*on-site*), atau *hybrid*.

c. Filter lokasi spesifik: Fitur penyaringan yang memungkinkan pengguna mempersempit pencarian lowongan berdasarkan wilayah (kota atau provinsi).
d. Terjemahan bahasa: Fitur untuk menerjemahkan atau memahami detail deskripsi pekerjaan yang ditulis dalam bahasa asing.
e. Reputasi perusahaan: Fitur yang menampilkan reputasi perusahaan penyedia magang guna membantu mahasiswa menilai kredibilitas tempat kerja.

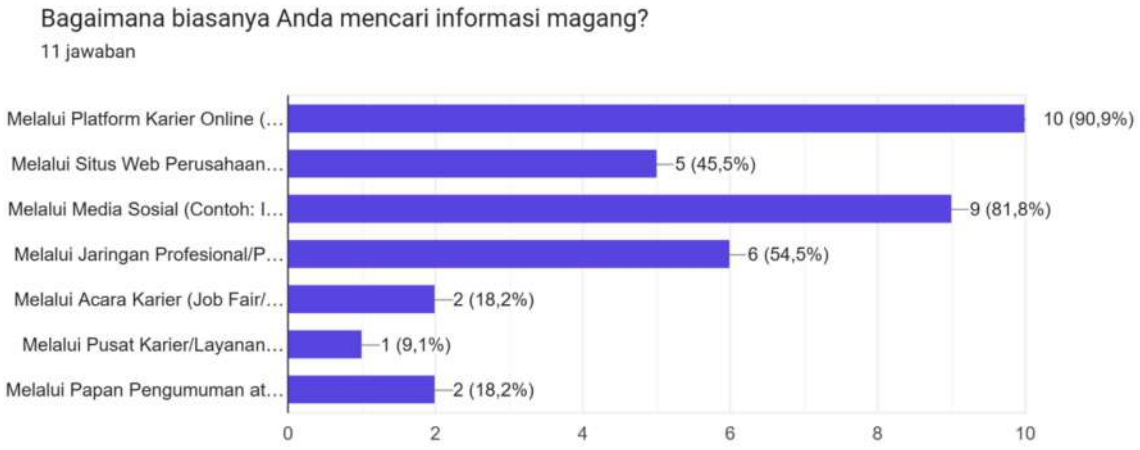
Hasil penilaian berbasis skala Likert ini selanjutnya akan dianalisis pada sub-bab 3.1.3. Fitur-fitur yang mendapatkan akumulasi nilai dan masuk ke dalam kategori "penting" dan “sangat penting” akan mendapatkan perhatian khusus dan dipertimbangkan secara mendalam untuk diimplementasikan pada aplikasi seiring berjalannya proses pengembangan aplikasi.

3.1.3 Hasil pengumpulan kebutuhan

Kuesioner terkumpul dari 11 responden. Hasil pengolahan data dijelaskan melalui analisis kendala dan analisis urgensi fitur.

Analisis hasil survei kendala dan perilaku pengguna

Proses identifikasi kebutuhan diawali dengan analisis terhadap perilaku mahasiswa dalam mencari informasi lowongan magang serta kendala-kendala utama yang dihadapi.



Gambar 3.2 Analisis platform pencarian informasi magang

Berdasarkan data hasil survei pada gambar 3.2, mayoritas responden mengandalkan platform karier daring (90,9%) dan media sosial (81,8%) sebagai instrumen utama dalam mencari informasi lowongan magang. Sebaliknya, pemanfaatan jaringan profesional (54,5%) dan situs web resmi perusahaan (45,5%) berada di tingkat menengah, sementara pusat karier universitas mencatat angka paling rendah, yaitu hanya sebesar 9,1%. Hal ini mengindikasikan

bahwa mahasiswa memiliki kecenderungan yang sangat kuat untuk mengeksplorasi platform digital eksternal dibandingkan fasilitas internal kampus.

Selanjutnya, kuesioner merangkum tantangan dan masalah yang dihadapi oleh mahasiswa, hal ini disajikan secara berurutan pada gambar 3.3 dan gambar 3.4.



Gambar 3.3 Analisis tantangan responden dalam mencari magang.



Gambar 3.4 Analisis faktor penyebab kesulitan magang.

Dari kedua gambar ini kendala utama responden dalam memperoleh tempat magang dapat diidentifikasikan ke dalam dua faktor, yaitu faktor eksternal dan internal:

1. Faktor internal: Hambatan terbesar bersumber dari tidak adanya koneksi atau jaringan profesional yang dapat memberikan rekomendasi (81,8%). Masalah ini diperparah oleh ketidakpastian terhadap kualitas CV yang dimiliki (63,6%) serta ketidaktahuan mengenai titik awal pencarian lowongan magang (63,6%).

2. Faktor eksternal: Responden mengeluhkan ketatnya persaingan dengan kandidat lain (63,6%), keterbatasan akses terhadap informasi lowongan yang valid (63,6%), serta ketidaksesuaian keterampilan yang diminta oleh industri (63,6%). Selain itu, proses seleksi yang panjang dan memakan waktu juga menjadi hambatan bagi 54,5% responden.

Banyaknya mahasiswa yang bingung mengenai titik awal pencarian, memperkuat alasan untuk adanya aplikasi ini.

Analisis urgensi fitur berbasis skala likert

Setelah memetakan kendala dasar pengguna, bagian kedua dari kuesioner digunakan untuk mengukur tingkat urgensi dari lima kandidat fitur yang direncanakan. Evaluasi ini dihitung menggunakan akumulasi persentase skala likert (skala 1 hingga 5) untuk menentukan fitur mana saja yang masuk ke dalam kategori “sangant penting” untuk langsung dibangun, maupun yang bersifat pendukung. Data distribusi penilaian dirangkum dalam tabel 3.1 berikut.

Tabel 3.1 Distribusi penilaian tingkat kepentingan fitur aplikasi

No	Fitur	1 (%)	2 (%)	3 (%)	4 (%)	5 (%)	Kesimpulan
1	Informasi gaji/insentif	0,0	0,0	9,1	36,4	54,5	Sangat Penting
2	Mode kerja (WFH/WFO)	0,0	0,0	0,0	45,5	54,5	Sangat Penting
3	Filter lokasi spesifik	0,0	0,0	9,1	45,5	45,5	Penting
4	Reputasi perusahaan	0,0	9,1	0,0	36,4	54,5	Penting
5	Terjemahan bahasa	0,0	18,2	36,4	27,3	18,2	Pendukung

Berdasarkan klasifikasi hasil pada Tabel 3.1, ditetapkan keputusan integrasi fitur ke dalam tahapan pengembangan sistem dengan rincian analisis sebagai berikut:

1. Fitur prioritas: Untuk fitur informasi gaji/insentif (90,9% gabungan skor 4 & 5), mode kerja WFH/WFO (100% gabungan skor 4 & 5), filter lokasi spesifik (91% gabungan skor 4 & 5), dan reputasi perusahaan (90,9% gabungan skor 4 & 5) menjadi prioritas utama untuk di intergrasikan ke aplikasi. Dari data ini menunjukkan transparansi gaji, fleksibilitas sistem kerja, kemudahan pencarian lokasi, dan kredibilitas perusahaan adalah pertimbangan utama bagi mahasiswa sebellum melamar magang.
2. Fitur pendukung: Untuk fitur terjemahan bahasa, jawaban responden cukup beragam dengan pilihan terbanyak pada skor 3 (36,4%). Oleh karena itu, fitur ini belum menjadi prioritas dan akan dipertimbangkan kembali seiring berjalannya proses pengembangan.

3.2 Tahap perancangan pengguna

Setelah data kebutuhan pengguna berhasil dikumpulkan pada tahap perencanaan, langkah selanjutnya adalah mentransformasikannya ke dalam prototipe rancangan sistem yang kemudian akan dites user awal. Tahap perancangan pengguna ini mencakup perancangan protipe, *test* dan *refine* yang diuraikan di bawah ini.

3.2.1 Prototipe

Tahap ini dilakukan untuk memberikan gambaran terstruktur mengenai bagaimana komponen-komponen teknologi di dalam aplikasi asisten GenAI ini saling berinteraksi, mengolah data, dan menghasilkan luaran sesuai dengan kebutuhan pengguna. Tahap ini mencakup arsitektur teknologi, alur data, serta rancangan antarmuka sebagai representasi visual yang akan diuji bersama calon pengguna. Penjelasan mengenai desain dan mekanisme kerja sistem dijabarkan pada bagian di bawah ini.

1. Arsitektur sistem

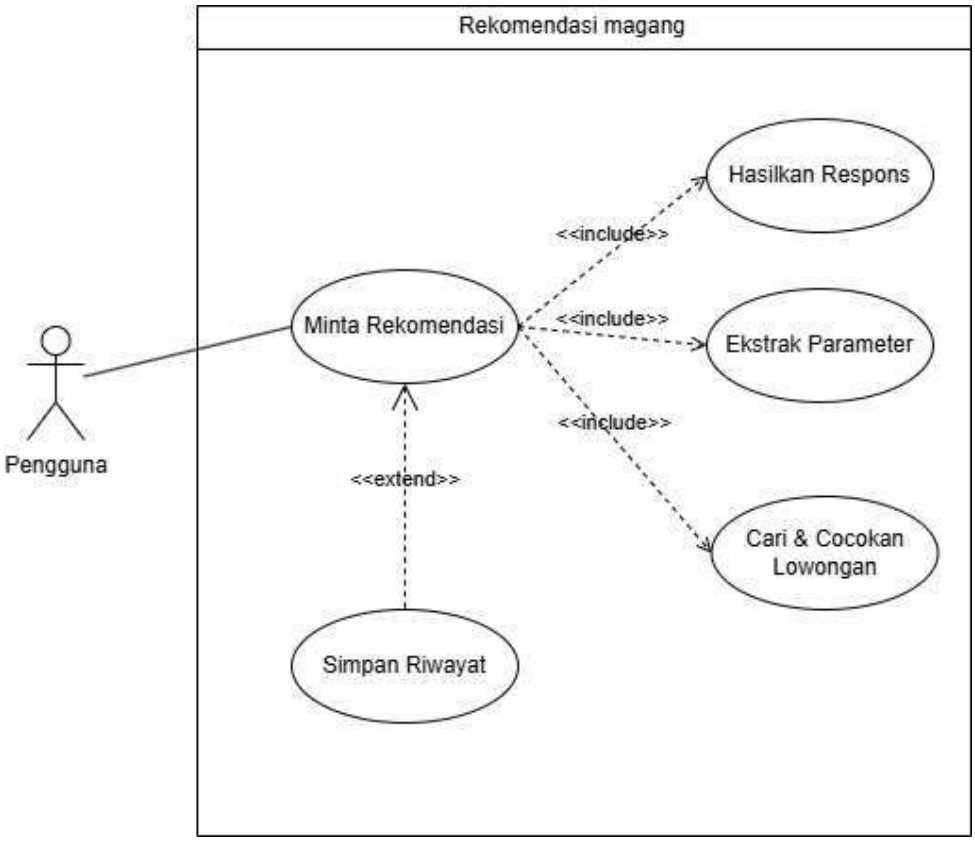
   Aplikasi yang dikembangkan mengadopsi arsitektur *retrieval-augmented generation (*RAG) sebagai pendekatan utama dalam menggabungkan LLM dengan data lowongan magang eksternal yang dinamis. Namun, berbeda dengan implementasi RAG klasik yang umumnya menggunakan *dense vector retrieval* terhadap dokumen lokal, aplikasi ini memanfaatkan API eksternal sebagai sumber *retrieval*.

   Dalam arsitektur ini, LLM menjalankan dua peran sebagai ekstraktor parameter dari input pengguna sekaligus sebagai generator teks rekomendasi setelah data lowongan ditarik melalui API. Pemilihan pendekatan ini didasarkan pada beberapa pertimbangan utama:

   a. Data lowongan magang bersifat dinamis dan terus berubah, sehingga pemanggilan API secara real-time lebih fleksibel.
   b. Penyedia API eksternal telah memiliki *dataset* lowongan yang diperbarui secara berkala, sehingga tidak perlu menyaring maupun merapikan data lowongan dari nol.

2. Diagram *use case*

   Interaksi fungsional antara aktor dan sistem pembuat rekomendasi diuraikan melalui diagram *use case* pada Gambar 3.5.



Gambar 3.5 Use case diagram

Sistem ini melibatkan satu aktor, yaitu pengguna. Pengguna berperan sebagai orang yang memulai semua *use case* karena setiap fungsi dalam sistem dipicu oleh kebutuhan pengguna untuk mencari rekomendasi magang.

Hubungan antarfungsi (*use case*) dijabarkan sebagai berikut:

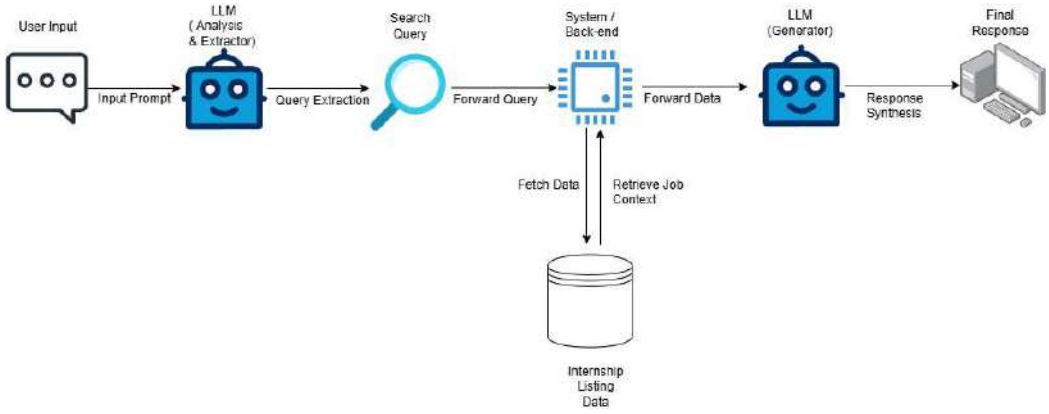
a. Minta rekomendasi magang: Ini merupakan *use case* utama yang dipicu langsung oleh pengguna saat menginput preferensi pencarian magang. *Use case* ini secara otomatis memanggil tiga use case turunan melalui relasi *include*.
b. Ekstrak parameter (*include*): Dipanggil oleh use case "minta rekomendasi magang" untuk menguraikan *input* dari pengguna menjadi parameter pencarian terstruktur. Pada tahap ini sistem berkomunikasi dengan LLM yang bertindak sebagai *analysis & extractor*.
c. Cari & cocokkan lowongan (*include*): Dipanggil oleh use case "minta rekomendasi magang" untuk melakukan pencarian data lowongan berdasarkan

parameter hasil ekstraksi. Pada tahap ini sistem berkomunikasi dengan API eksternal yang menyediakan dataset lowongan magang.

d. Hasilkan respons (*include*): Dipanggil oleh *use case* "minta rekomendasi magang" untuk mengenerate respons akhir berdasarkan profil pengguna dan data lowongan yang berhasil diambil. Pada tahap ini sistem kembali berkomunikasi dengan LLM yang kini bertindak sebagai generator rekomendasi.
e. Simpan riwayat (*extend*): Merupakan *use case* opsional yang dapat dipilih Pengguna setelah *use case* "minta rekomendasi magang" selesai dieksekusi. Hubungan *extend* digunakan karena fungsi ini bersifat opsional dan tidak selalu dijalankan dalam setiap sesi.

3. Mekanisme kerja RAG pada sistem

Mekanisme kerja sistem yang dirancang mengintegrasikan kecerdasan buatan berbasis *large language model* (LLM) untuk memproses permintaan pengguna dan mengenerate rekomendasi dengan sumber data magang eksternal.



Gambar 3.6 Alur RAG sistem

Berdasarkan rancangan pada Gambar 3.6, rancangan awal alur pemrosesan data di dalam sistem ini dijabarkan sebagai berikut:

a. *Input Prompt:* Pengguna memasukkan *input prompt* berupa pertanyaan, profil keahlian, atau instruksi pencarian ke dalam sistem melalui antarmuka aplikasi.
b. *Query extraction* (LLM *analysis & extractor*): Masukan tersebut diproses oleh komponen LLM pertama yang berperan sebagai *analysis & extractor*. Komponen ini bertugas menganalisis keinginan pengguna serta mengekstrak

inti informasi atau kata kunci penting untuk menyusun *search query* yang optimal.

c. *Forward query*: *Search query* yang telah diekstrak kemudian diteruskan menuju komponen *system / back-end*.

d. *Fetch data*: *Back-end* melakukan pencarian dan penarikan data ke basis data eksternal, yaitu Internship Data (fantastic.jobs dari RapidAPI). Basis data kemudian mengembalikan konteks lowongan magang yang relevan (*retrieve job context*) ke *back-end*.

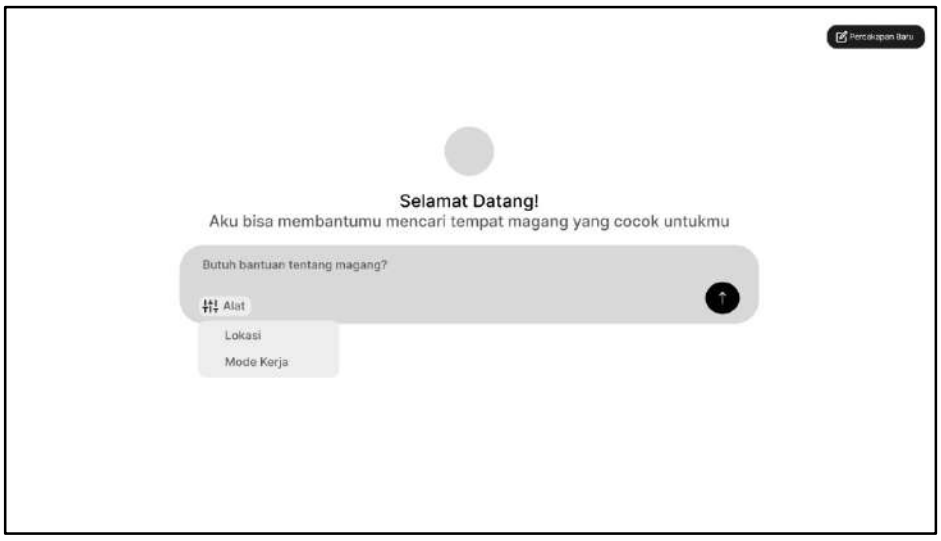
e. *Fetch data* ke generator: *Back-end* menggabungkan data konteks tersebut dengan queri asli pengguna, lalu meneruskannya (*forward data*) ke komponen LLM kedua, yaitu LLM (Generator).

f. *Response synthesis*: LLM (Generator) memadukan seluruh informasi tersebut untuk menghasilkan jawaban yang akurat dan relevan (*response synthesis*), yang kemudian disajikan sebagai *final response* pada layar pengguna.

4. Rancangan antarmuka

Rancangan antarmuka aplikasi dirancang menggunakan Figma dengan konsep minimalis, bersih, dan berorientasi pada kemudahan interaksi berbasis *chat-based interface*. Prinsip minimalis dipilih untuk memprioritaskan elemen-elemen penting dan menghilangkan kerumitan yang tidak perlu (Youvan, 2024). Hasil antarmuka ini dibagi menjadi dua:

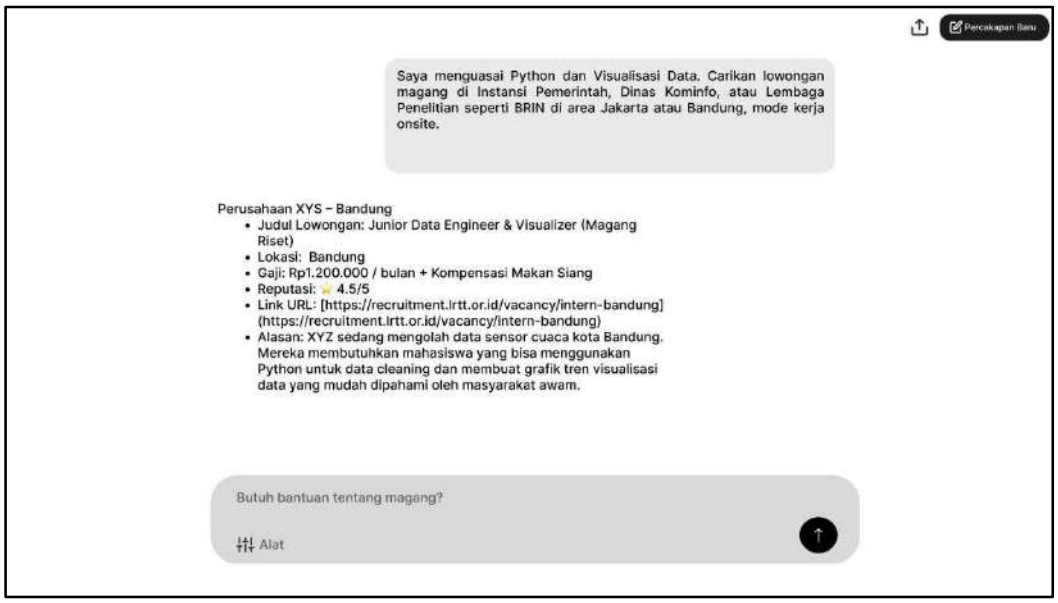
1. Halaman utama dan masukan awal



Gambar 3.7 Desain halaman awal aplikasi

Halaman pembuka aplikasi menyajikan ucapan selamat datang beserta teks panduan fungsional sistem. Fokus utama halaman ini berada pada komponen kotak input di bagian bawah yang bertuliskan "butuh bantuan tentang magang?". Kotak ini dirancang sebagai area utama bagi pengguna untuk memasukkan teks profil, keahlian, dan preferensi secara bebas, yang dilengkapi dengan tombol aksi kirim di sisi kanan. Di pojok kanan atas, disediakan tombol aksi "percakapan baru" untuk membuka sesi pencarian yang baru.

2. Halaman hasil rekomendasi



Gambar 3.8 Desain halaman hasil rekomendasi

Setelah pengguna memasukkan *input*, halaman akan menampilkan visualisasi riwayat percakapan secara runtut. Teks input pengguna terdapat dalam kolom *chat* di bagian sisi kanan, sementara respons dari sistem yang memuat daftar poin-poin hasil analisis dan rekomendasi lowongan magang akan muncul dalam gelembung pesan di bawahnya pada sisi kiri. Kotak *input* teks di bagian bawah layar tetap dipertahankan untuk mendukung interaksi pencarian lanjutan.

3.2.2 *Test* (rencana pengujian prototipe)

Prototipe yang telah disusun diuji terlebih dahulu sebelum memasuki fase konstruksi. Pengujian dilakukan dalam format *peer review* berupa uji coba interaktif terhadap rancangan Figma bersama dua orang partisipan yang merepresentasikan calon pengguna. Format ini dipilih berdasarkan dua pertimbangan: (1) tujuan sesi ini adalah mengidentifikasi masalah desain awal sebelum implementasi kode, sehingga cakupannya terbatas pada aspek rancangan;

dan (2) rekan sejawat yang familiar dengan bidang ini mampu memberikan umpan balik esensial dalam waktu singkat.

3.2.3 *Refine*

Setiap temuan pada tahap *test* sebelumnya ditinjau untuk menentukan apakah perlu dilakukan penyempurnaan rancangan sebelum fase konstruksi dimulai. Seluruh perubahan yang dihasilkan dijelaskan dalam bentuk perbandingan rancangan sebelum dan sesudah perbaikan, yang disajikan pada Bab IV. Rancangan hasil penyempurnaan inilah yang menjadi acuan pada tahap konstruksi.

3.3 Tahap konstruksi (*Construction*)

Setelah rancangan antarmuka dan alur sistem divalidasi oleh calon pengguna, proses dilanjutkan ke tahap konstruksi untuk mengimplementasikan model desain menjadi aplikasi yang fungsional. Proses pengodean dan implementasi kecerdasan buatan pada tahap ini didasarkan pada pertimbangan lingkungan implementasi, pendekatan rekayasa *prompt*, kriteria pemilihan model LLM, pendekatan iteratif, serta antisipasi terhadap kendala data eksternal sebagai berikut.

3.3.1 Lingkungan dan teknologi implementasi

Aplikasi ini dirancang untuk dieksekusi secara lokal (*on-premise*) tanpa bergantung pada layanan LLM berbasis *cloud*. Pemilihan *deployment on-premise* didasarkan pada pertimbangan utama:

1. Privasi data pengguna: Profil keahlian dan preferensi mahasiswa diproses sepenuhnya di sisi lokal tanpa transmisi ke server pihak ketiga.
2. Bebas biaya per-token: Sistem ini mengeliminasi biaya operasional yang biasa diterapkan oleh penyedia LLM berbasis *cloud*, sehingga lebih sesuai untuk konteks pengembangan akademis.

Konsekuensi dari pendekatan ini adalah tingginya ketergantungan sistem pada spesifikasi perangkat keras lokal. Eksekusi model berukuran besar pada GPU dengan kapasitas VRAM terbatas berpotensi memicu mekanisme *offloading* memori dari VRAM ke RAM, yang berdampak pada peningkatan waktu respons sistem. Konsekuensi ini diakui sebagai *trade-off* dari pemilihan *deployment on-premise*.

Untuk membangun aplikasi rekomendasi lowongan magang ini, perangkat lunak dan lingkungan pengembangan yang digunakan dibagi menjadi tiga komponen utama:

1. Komponen *backend* (Python & Flask): Bahasa pemrograman python dipilih sebagai pondasi utama di sisi *backend*. Pemilihan python didasarkan pada faktor familiaritas terhadap sintaks dan ekosistem bahasa tersebut, sehingga memangkas waktu adaptasi teknologi. Eksekusi LLM dilakukan secara lokal menggunakan Ollama Engine (0.15.2).
2. Komponen API eksternal: API eksternal digunakan sebagai basis data utama dari sistem ini. API yang digunakan adalah Internship API dari fantastic.jobs yang tersedia melalui platform RapidAPI. Paket API yang digunakan adalah paket yang free. Adapun kekurangan dari paket ini, yaitu (1) limit lowongan per bulan hanya 200 panggilan dengan maksimal 10 lowongan per panggilan; dan (2) Lowongan yang didapat hanya dari seminggu terakhir. Walaupun begitu dengan pemanfaatan API eksternal ini memungkinkan sistem untuk mendapatkan data lowongan magang yang *up-to-date* dan valid dari sumbernya. Informasi lebih lanjut dapat dilihat di lampiran bagian Dokumentasi lainnya.
3. Komponen *frontend* (HTML5 + JavaScript + TailwindCSS): Untuk antarmuka pengguna, struktur dasar aplikasi dibangun menggunakan HTML5 untuk elemen visual. JavaScript untuk menggerakkan fungsi antarmuka agar berjalan secara dinamis dan responsif. TailwindCSS digunakan untuk menjaga responsivitas tampilan, sementara `library jsPDF` dimanfaatkan untuk fungsionalitas ekspor riwayat percakapan ke dalam format dokumen.

Eksekusi LLM dilakukan secara lokal menggunakan Ollama Engine pada perangkat dengan spesifikasi sebagai berikut:

1. Perangkat keras:
   a. *Processor* (CPU): AMD Ryzen 7 5700X (8 Cores, 16 Threads)
   b. *Memory* (RAM): 16 GB DDR4
   c. *Graphics Processing Unit* (GPU): NVIDIA GeForce GTX 1660 Super (6 GB VRAM)
2. Perangkat lunak:
   a. Sistem Operasi: Windows 11 64-bit
   b. LLM *Runtime*: Ollama Engine 0.15.2

3.3.2 Alur rekayasa prompt

Fitur yang memerlukan bantuan AI dalam hal ini LLM ekstraktor dan LLM generator (gambar 3.6), dirancang logika dasar agar LLM dapat mengolah perintah melalui dua fase. Logika ini mengatur bagaimana teks masukan dari pengguna dan deskripsi lowongan yang diperoleh dari API eksternal diproses oleh *Large Language Model*. Secara garis besar, rekayasa *prompt* pada aplikasi ini dibagi menjadi dua fase utama:

1. Fase ekstraksi variabel: Sistem mengarahkan LLM untuk bertindak sebagai ekstraktor data yang bertugas menyaring teks bebas dari pengguna (profil, keahlian, dan preferensi kerja) menjadi parameter pencarian terstruktur berformat JSON. Parameter ini memisahkan komponen seperti kata kunci kueri, lokasi tujuan, dan status metode kerja (*remote* atau *on-site*).
2. Fase *generate* rekomendasi: Sistem menggabungkan profil pengguna dengan kumpulan data lowongan riil yang diperoleh dari API eksternal. Gabungan data tersebut dikirimkan kembali ke LLM bersama dengan serangkaian instruksi agar menghasilkan luaran rekomendasi yang terstruktur.

3.3.3 Kriteria evaluasi pemilihan model LLM

Karena sistem ini memanfaatkan *Large Language Model* (LLM) untuk menghasilkan rekomendasi tempat magang, dilakukan pengujian untuk menentukan model yang sesuai. Pengujian dilakukan secara kualitatif berbasis pengamatan terhadap keluaran tiap model. Pengujian tidak dilakukan secara bersamaan, tapi seiring proses pengembangan aplikasi. Keluaran dinilai berdasarkan empat kriteria: (1) kualitas bahasa Indonesia; (2) kepatuhan instruksi; (3) akurasi data (bebas halusinasi); dan (4) konsistensi (bebas dari pengulangan). Penilaian dinyatakan dalam skala kualitatif (Baik, Cukup, Buruk), dan model dengan performa terbaik dipilih untuk diimplementasikan ke sistem.

3.3.4 Pendekatan iteratif pada pengembangan

Mengingat karakteristik metodologi RAD yang menempatkan iterasi sebagai inti pengembangan, aplikasi ini dirancang untuk menerima dan mengintegrasikan umpan balik dari dua sumber selama proses pengembangan berlangsung:

1. Iterasi berbasis *peer review*: Sumber masukan berasal dari sesi *peer review* informal bersama perwakilan pengguna. Iterasi pada kategori ini dipicu dalam fase desain pengguna.

2. Iterasi berbasis temuan teknis: Sumber masukan berasal dari observasi pengembang selama proses pengodean dan uji coba aplikasi. Iterasi pada kategori ini dapat memicu perubahan pada antarmuka maupun pada arsitektur sistem.

3.3.5 Potensi kendala data eksternal

Karena aplikasi ini mengandalkan API eksternal sebagai *retrieval layer*, kualitas keluaran sistem akan dipengaruhi oleh karakteristik dan keterbatasan data yang disediakan oleh API. Oleh karena itu, beberapa potensi kendala diantisipasi sejak tahap perancangan, dengan strategi penanganan yang akan diterapkan pada level rekayasa *prompt* dan logika *backend*:

1. Potensi data yang tidak lengkap: Tidak semua entri lowongan pada API menyertakan informasi lengkap (misalnya besaran gaji atau mode kerja). Strategi penanganan: Mengatur agar sistem mencantumkan label "Tidak dicantumkan" untuk variabel yang kosong, sehingga aplikasi tetap dapat menampilkan informasi apa adanya tanpa menambahkan data fiktif.
2. Potensi keterbatasan pembaharuan indeks data: Dataset API mungkin tidak menyediakan indikator status aktif per listing, sehingga lowongan yang sudah ditutup tetap muncul.
3. Potensi ketidakcocokan kata kunci: Mekanisme pencarian API umumnya berbasis pencocokan kata kunci yang kaku, sementara LLM bekerja secara konteks makna. Kondisi ini berpotensi menyebabkan kegagalan pencarian.

3.4 Tahap peralihan (*cutover*)

Tahap *cutover* merupakan fase evaluasi akhir untuk memastikan aplikasi yang telah dibuat siap diterapkan dan mampu memenuhi ekspektasi pengguna. Proses pengujian sistem dirancang dengan mengkombinasikan dua metode, yaitu pengujian fungsional menggunakan pendekatan *black-box testing* dan pengujian penerimaan pengguna berbasis skala Likert.

3.4.1 Desain pengujian fungsional (*black-box testing*)

Pengujian fungsional menggunakan pendekatan *black-box testing* yang berfokus pada hasil dari fungsionalitas sistem. Pengujian dilakukan menggunakan matriks uji berisi 18 *test case* yang dikelompokkan ke dalam tiga kategori: *pipeline* RAG (TC-01 s.d. TC-08), antarmuka pengguna (TC-09 s.d. TC-13), dan kasus langka atau *edge case* (TC-14 s.d. TC-18). Pengelompokan ini bertujuan untuk memastikan pengujian mencakup tiga aspek sistem, yaitu validitas alur pemrosesan AI, keberhasilan komponen antarmuka, serta ketahanan sistem terhadap *input* yang tidak ideal.

Setiap *test case* dinilai berdasarkan kondisi *input*, hasil yang diharapkan (*expected output*), serta status akhir berupa *Passed*, *partial*, atau *fail*. Rincian matriks uji disajikan pada tabel 3.2.

Tabel 3.2 Matriks uji *black-box testing*

ID	Kategori	Skenario Pengujian
TC-01	*Pipeline* RAG	Ekstraksi parameter lengkap (skill, bidang, lokasi) dari *prompt* pengguna
TC-02		Ekstraksi dari *prompt* minimalis (lokasi & bidang implisit)
TC-03		Toleransi ekstraktor terhadap kesalahan ketik (*typo*) dan bahasa kasual
TC-04		Keberhasilan pemanggilan API fantastic.jobs untuk mengambil data
TC-05		Penanganan sistem saat hasil pencarian API kosong (*no result*)
TC-06		Pembentukan respons rekomendasi menggunakan bahasa Indonesia natural
TC-07		Pencegahan pembuatan data lowongan palsu di luar hasil API (anti-halusinasi)
TC-08		Pencegahan duplikasi lowongan yang sama dalam satu sesi respons
TC-09	Antarmuka pengguna	Pemuatan elemen antarmuka utama halaman awal secara benar
TC-10		Validasi pencegahan pengiriman *prompt* kosong oleh pengguna
TC-11		Validasi tautan lowongan otomatis aktif dan dapat diklik pengguna
TC-12		Validasi fungsi ekspor hasil rekomendasi ke dalam format PDF
TC-13		Keberhasilan penyimpanan dan pemuatan ulang riwayat percakapan
TC-14	*Edge case*	Variasi penulisan nama lokasi (contoh: DIY, Yogyakarta, Jogja)
TC-15		Kemampuan pemrosesan *prompt* panjang (>500 karakter) tanpa *error*
TC-16		Penanganan masukan campuran bahasa
TC-17		Penanganan permintaan di luar cakupan konteks rekomendasi magang
TC-18		Konsistensi rekomendasi lowongan pada pengujian *prompt* identik (3x *run*)

3.4.2 Desain pengujian penerimaan pengguna

Pengujian penerimaan pengguna dilakukan terhadap partisipan mahasiswa aktif dan nonaktif lintas jurusan yang merepresentasikan calon pengguna. Setiap partisipan diminta menjalankan skenario berupa memasukkan prompt tentang keahlian dan meminta rekomendai magang. Ada lima indikator yang digunakan meliputi:

1. Kesesuaian rekomendasi yang dihasilkan dengan *prompt* pengguna.
2. Kemudahan desain antarmuka aplikasi.
3. Kualitas bahasa natural pada respons sistem.
4. Akseptabilitas waktu respons sistem.
5. Niat menggunakan sistem secara berkelanjutan.

Kelima indikator ini disusun sendiri oleh penulis berdasarkan aspek-aspek yang dianggap penting dalam menilai kualitas sistem rekomendasi, yang mencakup relevansi hasil, kemudahan penggunaan, kualitas keluaran, kinerja, serta penerimaan pengguna. Penilaian dilakukan menggunakan skala Likert lima tingkat (1 = Sangat Tidak Setuju, 5 = Sangat Setuju).

BAB IV
HASIL DAN PEMBAHASAN

Pada bab ini, dipaparkan seluruh hasil dari tahapan-tahapan metodologi *rapid application development* (RAD) yang telah dirancang sebelumnya pada Bab III. Pembahasan di dalam bab ini mencakup analisis hasil kuesioner kebutuhan pengguna, proses validasi prototipe desain, implementasi kode program pada tahap konstruksi, pengujian sistem menggunakan model LLM *final*, hingga evaluasi akhir aplikasi. Hasil tahap perencanaan kebutuhan tidak diulang pada bab ini karena telah disajikan pada sub-bab 3.1.3 sebagai dasar pemodelan sistem.

4.1 Hasil perancangan pengguna (User design)

Sesuai pada sub-bab 3.2, tahap perancangan pengguna dijalankan dalam satu putaran penuh. Hasil setiap langkah diuraikan berikut ini.

4.1.1 Prototipe yang diujikan (*prototype*)

Prototipe yang yang digunakan terdapat pada langkah *prototype* yang berupa rancangan antarmuka pada platform Figma (Gambar 3.7 dan Gambar 3.8), yang merepresentasikan dua halaman utama aplikasi beserta alur interaksinya. Prototipe ini belum memuat logika pemrosesan, sehingga fokus pengujian terbatas pada aspek tata letak, alur interaksi, dan kelengkapan elemen fungsional.

4.1.2 Hasil pengujian prototipe (*test*)

Sesi *peer review* dilaksanakan bersama dua orang partisipan sebagai perwakilan calon pengguna. Sesi *peer review* berhasil menghasilkan empat temuan yang kemudian diimplementasikan ke dalam desain akhir aplikasi:

1. Tata letak dan tipografi: Masukan diperoleh agar format penyajian teks dan bahasa dibuat lebih menarik dan rapi.
2. Teks tautan: Pada rancangan awal, tautan eksternal menuju situs lowongan kerja masih berupa teks mentah biasa.
3. Format ekspor dokumen: Berdasarkan rancangan fungsional awal, fitur penyimpanan riwayat rekomendasi direncanakan menghasilkan berkas teks dalan format .txt. Namun, terdapat saran agar format dokumen diubah menjadi PDF.
4. Penyematan efek visual *hover* pada komponen antarmuka: Terdapat saran untuk meningkatkan *experience* interaksi.Keempat poin perbaikan ini menjadi dasar

pengembangan antarmuka pada tahap konstruksi yang dijelaskan pada sub-bab berikutnya.

4.1.3 Hasil penyempurnaan rancangan (*refine*)

Keempat temuan pada hasil *peer review* ditindaklanjuti sebagai perbaikan rancangan sebelum memasuki tahap konstruksi. Perbandingan rancangan sebelum dan sesudah penyempurnaan disajikan pada Tabel 4.1.

Tabel 4.1 Perbandingan rancangan sebelum dan sesudah penyempurnaan

No	Sebelum	Sesudah	Alasan perubahan
1	Teks rekomendasi disajikan seragam tanpa penekanan	Nama posisi dan perusahaan ditampilkan dalam format tebal, disertai ikon sebagai penanda kategori informasi	Mempercepat pemindaian informasi kunci pada daftar rekomendasi
2	URL lowongan berupa teks mentah	URL dikonversi menjadi tautan aktif yang dapat diklik langsung	Menghilangkan langkah penyalinan manual oleh pengguna
3	Ekspor riwayat ke berkas .txt	Ekspor riwayat ke berkas .pdf	Format dokumen lebih layak disimpan dan dibagikan
4	Elemen interaktif tanpa umpan balik visual	Ditambahkan efek hover dan animasi status pemuatan	Memberikan kepastian bahwa aksi pengguna terdeteksi sistem

Rancangan hasil penyempurnaan inilah yang menjadi acuan pada tahap konstruksi.

4.2 Hasil konstruksi (*Construction*)

Tahap konstruksi dilakukan dengan menerjemahkan seluruh fungsionalitas hasil validasi rancangan pengguna ke dalam kode program yang utuh. Bagian ini menguraikan realisasi lingkungan implementasi dan keputusan teknis tambahan yang muncul, implementasi teknik rekayasa *prompt* dan klasifikasi intensi, hasil eksperimen pemilihan model LLM, hasil iterasi pengembangan, serta analisis kendala data API eksternal.

4.2.1 Realisasi lingkungan dan *session handling*

Stack teknologi yang dirancang pada sub-bab 3.3.1 berhasil direalisasikan tanpa perubahan substansial. *Backend* python dan flask, *runtime* LLM via Ollama, *frontend* berbasis HTML5 + JavaScript + TailwindCSS, serta integrasi API fantastic.jobs melalui RapidAPI seluruhnya berfungsi sesuai rancangan pada lingkungan pengembangan yang telah ditentukan. Selain konfirmasi tersebut, terdapat beberapa keputusan teknis tambahan yang muncul selama tahap konstruksi yang patut didokumentasikan, yaitu pengamanan kredensial API, migrasi mekanisme manajemen sesi, penambahan sistem *logging*, serta perancangan mekanisme *reset* memori percakapan.

1. Pengamanan kredensial API eksternal

Untuk menjamin keamanan kredensial layanan eksternal, kunci API RapidAPI tidak ditanamkan langsung di dalam *backend* sistem, melainkan dimuat melalui *environment variable* menggunakan `library python-dotenv`. Pendekatan ini mencegah kebocoran kredensial saat sistem dibagikan kepada pihak lain, sekaligus mempermudah pengubahan kredensial tanpa memodifikasi *backend* sistem.

2. Migrasi ke manajemen sesi *server-side*

Pada tahap implementasi awal, ditemukan keterbatasan teknis ketika menggunakan mekanisme sesi bawaan flask yang berbasis *cookie*: ukuran data lowongan magang yang dikembalikan oleh API eksternal dapat melebihi batas ukuran *cookie* (4 KB), sehingga memicu galat penyimpanan. Untuk mengatasi keterbatasan ini, sistem dimigrasikan ke pendekatan *server-side session* menggunakan *flask-session* dengan jenis penyimpanan berbasis sistem file (*filesystem*). Konfigurasi yang diterapkan adalah sebagai berikut:

```
app.config['SESSION_TYPE'] = 'filesystem'
app.config['SESSION_FILE_DIR'] = './flask_sessions'
app.config['SESSION_PERMANENT'] = False
app.config['SESSION_USE_SIGNER'] = True
app.config['PERMANENT_SESSION_LIFETIME'] = timedelta(hours=2)
```

Gambar 4.1 Konfigurasi *session*

Pendekatan ini menyimpan seluruh data sesi (termasuk arsip lowongan terakhir) dalam berkas terenkripsi di direktori `./flask_sessions/` pada sisi server. Klien hanya menerima *session ID* terenkripsi melalui *cookie*, sehingga tidak ada lagi batas ukuran data. Waktu sesi diatur selama dua jam untuk menjaga keseimbangan antara kenyamanan pengguna dan kapasitas penyimpanan server.

3. Sistem *logging* untuk *debugging*

Sebagai dukungan tambahan terhadap proses *debugging*, aplikasi dilengkapi sistem *logging* yang menulis catatan aktivitas penting (klasifikasi intensi, hasil ekstraksi parameter, status pemanggilan API, jumlah rekomendasi yang dihasilkan) ke berkas `app.log` sekaligus ke konsol. Catatan ini digunakan untuk melihat alur eksekusi sistem saat terjadi kesalahn/*error* tanpa harus menjalankan ulang skenario uji.

4. Mekanisme *reset* memori percakapan

Optimasi juga diterapkan pada mekanisme penghapusan riwayat percakapan. Pada fungsi JavaScript `resetChat()`, sistem dirancang untuk membersihkan tampilan percakapan di layar sekaligus mengirimkan permintaan `fetch("/")` ke *backend* flask. Permintaan tersebut akan memicu fungsi `session.clear()` pada server, sehingga ketika

tombol "percakapan baru" ditekan, seluruh memori terkait lowongan terakhir akan dihapus dan memulai sesi pencarian yang baru.

4.2.2 Implementasi RAG dan rekayasa prompt pada sistem

Sub-bab ini menjelaskan implemetasi dari mekanisme *retrieval-augmented generation* dan rekayasa prompt. Dalam bab ini ada empat tahap yaitu (1) *Input promt*; (2) *Query extraction*; (3) *Forward query, fecth data, dan retrieve job ;* serta (4) *Generate* dan *final response*.

1. Input prompt

   Tahap ini dimulai ketika *route* `/chat` menerima masukan pengguna beserta riwayat percakapan yang dikirim dari antarmuka aplikasi dalam format JSON.

```
@app.route('/chat', methods=['POST'])
def chat():
    user_input = request.json.get('message')
    chat_history = request.json.get('history', [])

    if not user_input:
        return jsonify({"reply": "Silakan ceritakan latar belakangmu."})

    logger.info(f"User input: '{user_input[:100]}...'")
```

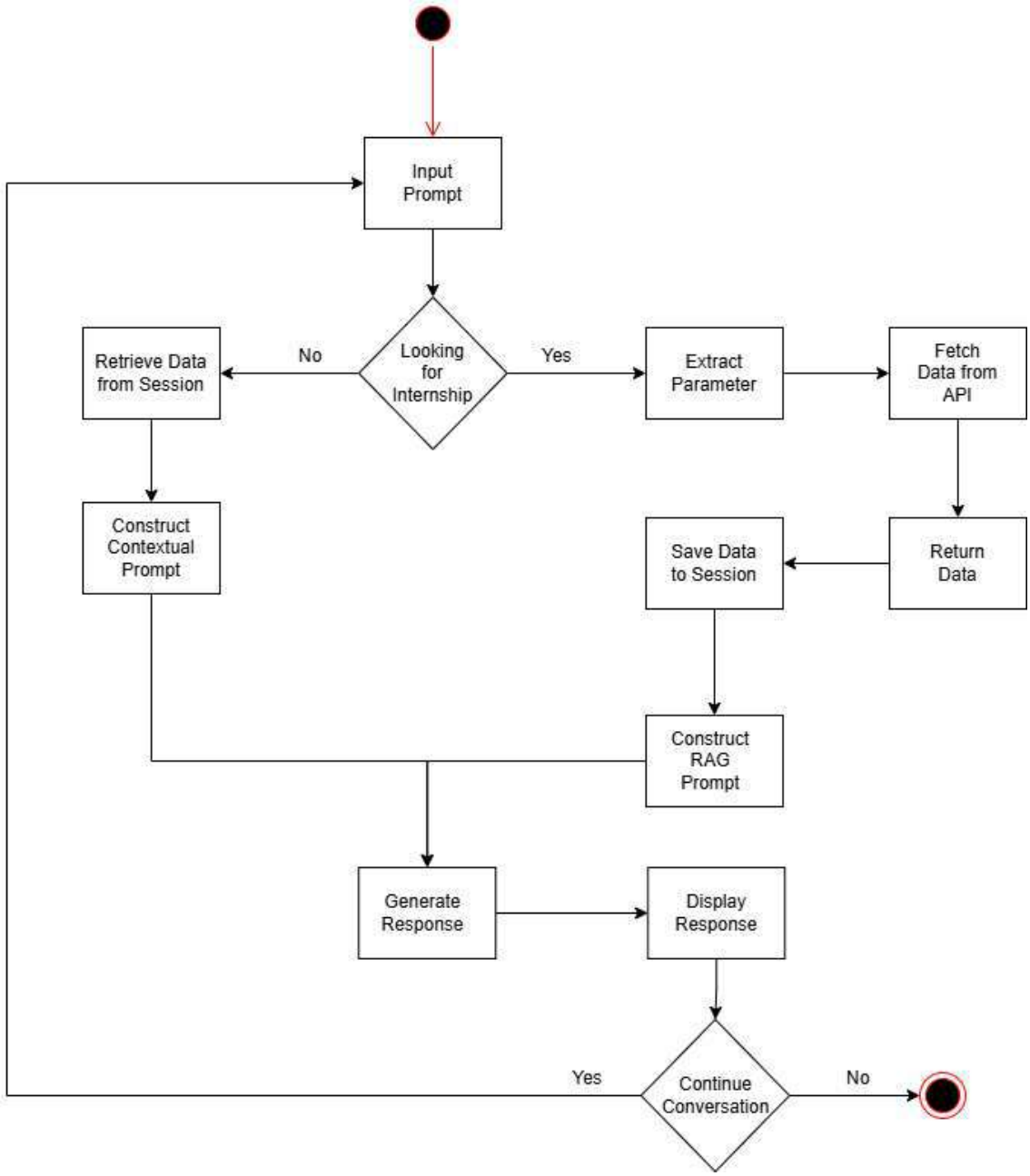
Gambar 4.2 Potongan kode penerimaan masukan pengguna

2. *Query extraction*

   Pada tahap ini awalnya dirancang hanya melakukan satu tugas, yaitu ekstraksi parameter. Kemudian pada saat tahap konstruksi ini teridentifikasi bahwa menjalankan alur RAG penuh untuk setiap input hanya akan memanggil API secara terus menurus walaupun isi input hanya menanyakan pertanyaan lanjutan tentang rekomendasi yang telah di berikan. Karena API yang digunakan hanya menggunkan *free plan* dimana limit lowongan per bulan hanya 200 panggilan. Karena itu pada tahap ini ditambahkan mekanisme klasifikasi intensi yang merutekan masukan ke salah satu dari dua jalur pemrosesan. Sehingga pada tahap ini melakukan dua pemanggilan LLM dengan peran yang berbeda.

   a. Klasifikasi intensi

      Dua jalur pemrosesan hasil penambahan mekanisme ini digambarkan pada Gambar 4.3.



Gambar 4.3 Alur klasifikasi intensi dua jalur

Klasifikasi dijalankan oleh fungsi `classify_intent()` yang memanfaatkan LLM untuk mengategorikan setiap masukan pengguna ke dalam dua label, sebagaimana disajikan pada Gambar 4.4.

```
def classify_intent(user_input):
    prompt = f"""
    Tugas: Klasifikasikan intensi pesan dari pengguna berikut.
    Pesan: "{user_input}"

    Pilihlah satu dari dua kategori berikut:
    1. "CARI_MAGANG": Jika pengguna menceritakan profilnya,
```

```
       mencari lowongan baru, menyebutkan skill baru untuk dicari,
       atau meminta rekomendasi magang secara eksplisit.
    2. "CHAT_BIASA": Jika pengguna menyapa, berterima kasih,
       bertanya tentang tips umum, atau menanyakan
       detail/pertanyaan lanjutan terkait lowongan yang sebelumnya
       sudah diberikan AI.

    Output HANYA kata "CARI_MAGANG" atau "CHAT_BIASA" tanpa
    penjelasan apa pun.
    """
    try:
        response = ollama.chat(
            model=MODEL_NAME,
            messages=[{"role": "user", "content": prompt}]
        )
        intent = response['message']['content'].strip()
        logger.info(f"Intent    classified:    '{user_input[:50]}...'    →
{intent}")

        if "CARI_MAGANG" in intent:
            return "CARI_MAGANG"
        return "CHAT_BIASA"
    except Exception as e:
        logger.error(f"Intent classification failed: {e}")
        return "CARI_MAGANG"
```

Gambar 4.4 Fungsi classify_intent()

Disini LLM hanya akan memberi jawaban atanra `CHAT_BIASA` atau `CARI_MAGANG`. Jika pemanggilan LLM gagal, fungsi secara otomatis akan mengembalikan / menhhasilkan jawaban `CARI_MAGANG`.

b. Perutean berdasarkan hasil klasifikasi

Hasil klasifikasi menentukan jalur pemrosesan yang dieksekusi, sebagaimana disajikan pada Gambar 4.5.

```
intent = classify_intent(user_input)

if intent == "CHAT_BIASA":
    job_sesi = session.get('job_sesi', [])
    logger.info(f"CHAT_BIASA session data:")
    logger.info(f"  Total jobs in session: {len(job_sesi)}")
    for i, job in enumerate(job_sesi):
        logger.info(f"  Job {i}: title={job.get('title')}, "
                    f"org={job.get('organization')}")

    messages = []
    for msg in chat_history[-6:]:
        messages.append({"role": msg['role'], "content": msg['content']})
    messages.insert(0, {"role": "system", "content": context})
    messages.append({"role": "user", "content": user_input})

    try:
        res = ollama.chat(model=MODEL_NAME, messages=messages)
        return jsonify({"reply": res['message']['content']})
    except Exception as e:
        logger.error(f"CHAT_BIASA response failed: {e}")
        return jsonify({"reply": f"Terjadi kesalahan pada AI: {str(e)}"})
else:
    analisis = extract_params(user_input)
    jobs = call_api(analisis)
```

Gambar 4.5 Potongan kode perutean dua jalur

Pada jalur `CHAT_BIASA`, sistem tidak memanggil API eksternal. Sumber data eksternal yang digunakan adalah `job_sesi`, yaitu hasil lowongan yang tersimpan pada sesi yang sedang berjalan. Riwayat percakapan juga digunakan tapi dibatasi enam pesan terakhir (`chat_history[-6:]`) untuk menjaga *prompt* tetap terkendali.

c. Instruksi pada jalur CHAT_BIASA

Karena jalur ini menjawab tanpa menarik data baru, resiko halusinasi dimitigasi melalui instruksi yang dimasukkan kedalam prompt, sebagaimana disajikan pada Gambar 4.6.

```
context = f"""Kamu adalah AI Internship Assistant yang ramah dan
profesional,
khusus membantu mahasiswa Indonesia mencari informasi tentang magang.

KONTEKS DATA LOWONGAN TERAKHIR YANG DIBERIKAN:
{json.dumps(job_sesi, ensure_ascii=False, indent=2)}

ATURAN PENTING (WAJIB DIPATUHI):

1. ATURAN ANTI-HALUSINASI (PALING UTAMA):
   - HANYA gunakan informasi dari "DATA LOWONGAN TERAKHIR" di atas
   - JANGAN PERNAH mengarang nama perusahaan, posisi, gaji, lokasi,
     atau detail lain yang TIDAK ADA dalam data tersebut
   - JANGAN PERNAH menambah lowongan baru dari pengetahuan umummu
   - Jika user menanyakan lowongan yang TIDAK ADA dalam data, jawab
     dengan jujur dan tawarkan pencarian baru

2. ATURAN PENGGUNAAN DATA:
   - Jika user bertanya tentang lowongan spesifik, cari di data
     dan sebutkan detail SESUAI data yang ada
   - Jika user bertanya perbandingan, bandingkan ANTAR LOWONGAN
     YANG ADA, berikan alasan berdasarkan informasi tersedia

3. ATURAN HANDLE DATA KOSONG:
   - Jika data lowongan kosong, jawab dengan menawarkan pencarian baru

4. ATURAN SCOPE:
   - Topik yang BOLEH: pencarian magang, rekomendasi karir, tips
     melamar magang, persiapan CV/wawancara
   - Topik yang HARUS dialihkan: pertanyaan di luar konteks magang/karir
   - Format pengalihan: "Maaf, saya khusus membantu pencarian dan
     persiapan magang. Apakah ada yang bisa saya bantu terkait magang?"

5. ATURAN FORMAT:
   - Bahasa Indonesia yang ramah dan profesional
   - Format markdown yang rapi
   - Jika mereferensikan lowongan, sebutkan nama posisi dan perusahaan
   - Jangan terlalu panjang—to-the-point

PRINSIP UTAMA: LEBIH BAIK MENGAKUI TIDAK TAHU DARIPADA MENGARANG.
```

Gambar 4.6 Instruksi (*prompt*) konteks pada jalur CHAT_BIASA

d. Ekstraksi parameter pencarian

Apabila intensi adalah `CARI_MAGANG`, sistem memanggil fungsi `extract_params()` untuk mengubah teks menjadi parameter pencarian berformat JSON. Untuk baris kodenya seperti ini ketika dipanggil `analisis = extract_params(user_input)` yang

terdapat pada gambar 4.5. Fungsi ini terdiri atas dua bagian, yaitu teks instruksi (prompt) yang dikirim ke LLM (Gambar 4.7) dan kode pemanggilan beserta penanganan hasilnya (Gambar 4.8).

```
Tugas: Analisis profil user dan berikan rekomendasi posisi magang.
Profil User: "{user_story}"

Aturan:
1. 'rekomendasi_posisi': List 3 posisi yang cocok dengan skill user.
2. 'api_query': Buat string pencarian menggunakan operator OR.
   PENTING - Pertimbangkan variasi penamaan di industri Indonesia:
   - Title utama yang paling cocok
   - 2-3 sinonim atau title terkait
   - 1-2 variasi konvensi industri (formal/informal)
   - GUNAKAN 4-6 variasi total
   - JANGAN masukkan nama industri (FMCG, Banking, dll) ke api_query,
     karena ini bukan nama posisi
   Contoh untuk "ingin magang sebagai pembuat konten":
   "Content Creator" OR "Social Media" OR "Digital Marketing" OR
   "Multimedia" OR "Creative"
   Contoh untuk "mahasiswa Teknik Lingkungan":
   "Environmental" OR "Sustainability" OR "HSE" OR "EHS" OR
   "Safety" OR "Environment Officer"
   Contoh untuk "ingin magang analisis data":
   "Data Analyst" OR "Business Analyst" OR "Data Science" OR
   "Analytics" OR "Business Intelligence"
3. 'location': Lokasi pencarian yang dapat berupa nama kota
   atau nama negara, dalam format kanonikal.
   PENTING - Gunakan nama resmi yang dikenali API:
   - Untuk kota:
     * "Yogyakarta" (BUKAN Jogja, DIY)
     * "Jakarta" (BUKAN JKT, DKI)
     * "Surabaya" (BUKAN SBY)
   - Untuk negara:
     * "Indonesia" (BUKAN ID, INA)
     * "United States" (BUKAN US, USA)
     * "United Kingdom" (BUKAN UK)
   - Default null jika user tidak menyebutkan lokasi spesifik
4. 'is_remote': true jika user menyebut remote/WFH/work from home.

Output JSON saja:
{
    "rekomendasi_posisi": [],
    "api_query": "...",
    "location": "...",
    "is_remote": false,
    "alasan": "Penjelasan rekomendasi posisi dan kenapa cocok dengan
    profil pengguna. Jika user menyebut industri spesifik, jelaskan
    bahwa hasil mencakup berbagai industri karena filter spesifik per
    industri tidak tersedia."
}
```

Gambar 4.7 Instruksi ekstraksi parameter profil pengguna

Pada instruksi ini (prompt) dilakukan 2 mitigasi, yaitu pada aturan nomor 2 meminta LLM menuliskan empat sampai enam variasi nama posisi yang bermakna sama, agar lowongan dengan judul berbeda tetap ikut ditemukan. Aturan nomor 3 meminta LLM menuliskan nama lokasi dalam bentuk baku yang dikenali API, misalnya "Yogyakarta" dan bukan "Jogja". Selain itu, aturan nomor 2 juga menyertakan tiga contoh pemetaan dari profil pengguna ke kata kunci pencarian.

```
def extract_params(user_story):
    prompt = f"""...instruksi pada Gambar 4.7..."""

    try:
        response = ollama.chat(
            model=MODEL_NAME,
            messages=[{"role": "user", "content": prompt}]
        )
        content = response['message']['content'].strip()
        output_json = re.search(r'\{.*\}', content, re.DOTALL)
        result = json.loads(output_json.group()) if output_json \
            else json.loads(content)
        logger.info(f"Extracted params: query='{result.get('api_query')}',
"
                    f"location='{result.get('location')}',
                    f"remote={result.get('is_remote')}")
        return result
    except Exception as e:
        logger.error(f"Parameter extraction failed: {e}")
        return {
            "rekomendasi_posisi": ["General Intern"],
            "api_query": "Internship",
            "location": None,
            "is_remote": False,
            "alasan": "Gagal menganalisis secara mendalam, mencari posisi
umum."
        }
```

Gambar 4.8 Kode extract_params() dan penaganan hasil

Hasil dari fungsi ini berupa objek dengan lima kunci. Empat kunci pertama diteruskan ke tahap pemanggilan API, sedangkan kunci alasan disimpan untuk ditampilkan sebagai bagian analisis profil pada respons akhir.

3. *Forward query, fecth data,* dan *retrieve job*

Parameter hasil ekstraksi diteruskan ke fungsi `call_api()` yang menyusun dan mengirim permintaan GET ke Internship API, sebagaimana disajikan pada Gambar 4.9.

```
def call_api(params):
    is_remote = params.get('is_remote', False)
    if isinstance(is_remote, bool):
        remote_str = "true" if is_remote else "false"
    else:
        remote_str = str(is_remote).lower()

    api_params = {
        "title_filter": params.get('api_query'),
        "location_filter": params.get('location'),
        "remote": remote_str,
        "include_ai": "true",
        "limit": 10
    }
    api_params = {k: v for k, v in api_params.items() if v is not None}

    headers = {
        "X-RapidAPI-Key": RAPIDAPI_KEY,
        "X-RapidAPI-Host": RAPIDAPI_HOST
    }

    try:
        logger.info(f"Calling API with params: {api_params}")
        r = requests.get(API_URL, params=api_params,
                         headers=headers, timeout=20)
```

```
        if r.status_code == 200:
            data = r.json()
            jobs = data if isinstance(data, list) else data.get('data', [])
            logger.info(f"Total Jobs Returned: {len(jobs)}")
            return jobs
        else:
            logger.error(f"API error: status {r.status_code}")
            return []
    except requests.Timeout:
        logger.error("API request timeout")
        return []
    except Exception as e:
        logger.error(f"API call failed: {e}")
        return []


def location_fix(job):
    locations = job.get("locations_derived")
    if isinstance(locations, list) and len(locations) > 0:
        return locations[0]
    elif isinstance(locations, str):
        return locations
    return "Tidak dicantumkan"
```

Gambar 4.9 Fungsi call_api() dan location_fix()

Fungsi ini di panggil pada gambar 4.5 , yaitu bagian ini `jobs = call_api(analisis)`. Ketika API tidak menghasilkan satu pun lowongan, sistem menghentikan alur dan memberikan respons, sebagaimana disajikan pada Gambar 4.10.

```
if not jobs:
    logger.info("No jobs found for query")
    return jsonify({
        "reply": f"**Analisis Profil:**\n{analisis['alasan']}\n\n"
                 f"Maaf, tidak ditemukan lowongan yang cocok saat ini. "
                 f"Hal ini dapat terjadi karena keterbatasan ketersediaan "
                 f"data lowongan dalam 7 hari terakhir."
    })
```

Gambar 4.10 Potongan kode penanganan hasil pencarian jika kosong

Apabila pencarian berhasil, data lowongan yang diperoleh dimasukkan dalam dua variabel , sebagaimana disajikan pada Gambar 4.11.

```
session['job_sesi'] = jobs[:10]
logger.info(f"Saved {len(jobs[:10])} jobs to session")

list_jobs = []
for idx, job in enumerate(jobs[:10]):
    list_jobs.append({
        "index": idx,
        "title": job.get("title", ""),
        "company": job.get("organization", ""),
        "location": location_fix(job),
        "description": job.get("ai_core_responsibilities")
                       or job.get("description", "")
    })
```

Gambar 4.11 Potongan kode penyimpanan sesi dan penyusunan konteks lowongan

Variabel `jobs` menampung data mentah lengkap yang kemudian disimpan ke sesi, sekaligus menjadi sumber asli pada tahap *generate respons*. Variabel `list_jobs` berisi empat atribut yang akan dikirimkan ke LLM sebagai konteks. Atribut `index` disertakan

agar LLM dapat memilih lowongan pilihannya melalui nomor urut, bukan dengan menuliskan ulang datanya. Untuk data mentah dari API disajikan pada gambar 4.12 (bentuk ringkas). Hasil lengkap satu lowongan ada pada lampiran bagian Data mentah dari API lengkap.


```
[
  {
    "id": "2286325067",
    "date_posted": "2026-07-30T12:08:26",
    "title": "IT Programmer Intern",
    "organization": "Dynapack Asia",
    "date_validthrough": "2026-08-27T14:22:27",
    "salary_raw": null,
    "employment_type": ["INTERN"],
    "url": "https://id.linkedin.com/jobs/view/it-programmer-intern-at-...",
    "source_type": "jobboard",
    "source": "linkedin",
    "cities_derived": ["Tangerang"],
    "regions_derived": ["Banten"],
    "countries_derived": ["Indonesia"],
    "locations_derived": ["Tangerang, Banten, Indonesia"],
    "remote_derived": false,
    "linkedin_org_employees": 185,
    "linkedin_org_industry": "Plastics Manufacturing",
    "linkedin_org_followers": 7588,
    "linkedin_org_headquarters": "Karawaci, Tangerang, Banten",
    "seniority": "Tidak Berlaku",
    "directapply": false
  }
]
```


Gambar 4.12 Data lowongan mentah hasil dari API

Pada gambar ini memperlihatkan atribut `salary_raw` bernilai kosong, yang mengonfirmasi bahwa tidak seluruh data lowongan menyertakan informasi gaji.

4. *Generate* dan *final response*.

   Pada tahap akhir, sistem menggabungkan profil pengguna dengan konteks lowongan, lalu mengirimkannya kembali ke LLM melalui variabel `final_prompt`, sebagaimana disajikan pada Gambar 4.13.

```
PROFIL USER: {user_input}
DATA LOWONGAN: {json.dumps(list_jobs)}

TUGAS:
Pilih maksimal 5 lowongan dari DATA LOWONGAN yang paling cocok
dengan PROFIL USER.
Terjemahkan informasi lokasi dan ringkasan deskripsi lowongan
pilihan tersebut ke dalam Bahasa Indonesia yang baik dan
profesional.

ATURAN STRUKTUR OUTPUT (WAJIB):
Anda HARUS merespon dalam format JSON bersih dengan struktur
seperti berikut tanpa tambahan teks penjelas apa pun:
{
    "rekomendasi": [
        {
            "index": 0,
            "lokasi_indo": "Hasil terjemahan lokasi ke Bahasa
            Indonesia (Contoh: 'Jakarta, Indonesia' atau 'Remote')",
```

```
            "deskripsi_indo": "Ringkasan pendek deskripsi pekerjaan
            yang sudah diterjemahkan ke Bahasa Indonesia maksimal 2
            kalimat",
            "alasan_cocok": "Penjelasan singkat menggunakan Bahasa
            Indonesia kenapa posisi ini cocok dengan profil user"
        }
    ]
}

Tampilkan maksimal 5 rekomendasi. Jika hanya ada 1 data
lowongan, berikan 1 saja di dalam list.
```

Gambar 4.13 Instruksi untuk rekomendasi (final_prompt)

Pada gambar ini LLM diminta mengembalikan nomor indeks lowongan yang terpilih, dua hasil terjemahan(lokasi_indo dan deskripsi_indo), dan satu alasan kesesuaian. Hal ini dilakukan untuk memitigasi halusinasi / perubahan isi data.

Hasil dari LLM ini kemudian diurai, dan nomor indeksnya digunakan untuk mengambil kembali data asli dari `jobs[idx]`. Penggabungan antara data mentah API dan hasil terjemahan LLM menjadi respons akhir disajikan pada Gambar 4.14.

```
res = ollama.chat(
    model=MODEL_NAME,
    messages=[{"role": "user", "content": final_prompt}]
)
ai_respons = res['message']['content'].strip()
output_json = re.search(r'\{.*\}', ai_respons, re.DOTALL)
hasil_rekomendasi = json.loads(output_json.group()) if output_json \
    else json.loads(ai_respons)

output_final = f"**Analisis :** {analisis['alasan']}\n\n---\n\n" \
               f"Berikut adalah rekomendasi untukmu:\n\n"

rekomendasi_list = hasil_rekomendasi.get("rekomendasi", [])
for i, recc in enumerate(rekomendasi_list):
    idx = int(recc.get("index", 0))
    if idx >= len(jobs):
        continue

    job_api = jobs[idx]
    posisi = job_api.get("title", "Tidak dicantumkan")
    perusahaan = job_api.get("organization", "Tidak dicantumkan")
    gaji = job_api.get("ai_salary_value") or "Tidak dicantumkan"
    reputasi = job_api.get("linkedin_org_followers") or "0"
    link_asli = job_api.get("url", "#")
    lokasi_id = recc.get("lokasi_indo", location_fix(job_api))
    deskripsi_id = recc.get("deskripsi_indo",
                            job_api.get("ai_core_responsibilities",
                                        "Tidak dicantumkan"))
    alasan = recc.get("alasan_cocok", "Cocok dengan kualifikasi Anda.")

    output_final += f"{i+1}. **{posisi}** di **{perusahaan}**\n"
    output_final += f"   Lokasi: {lokasi_id}\n"
    output_final += f"   Gaji: {gaji}\n"
    output_final += f"   Deskripsi: {deskripsi_id}\n"
    output_final += f"   Reputasi: {reputasi} followers\n"
    output_final += f"   Link: {link_asli}\n"
    output_final += f"   *Kenapa cocok?* {alasan}\n\n"

logger.info(f"Generated {len(rekomendasi_list)} recommendations")
return jsonify({"reply": output_final})
```

Gambar 4.14 Potongan kode *generate respons* akhir

Respons akhir dikirim kembali ke antarmuka, diterima oleh fungsi `sendMessage()` pada index.html, lalu ditampilkan oleh fungsi `renderChat()` sebagai gelembung pesan. Pada tahap penampilan inilah fungsi `formatMessage()` mengonversi URL mentah menjadi tautan aktif.

4.2.3 Hasil eksperimen pemilihan model LLM

Mengacu pada empat kriteria yang dirancang pada sub-bab 3.3.3 (Konsistensi, kepatuhan instruksi, tingkat halusinasi, dan kualitas Bahasa Indonesia), pengujian berbasis pengamatan dilakukan pada tiga kandidat model LLM lokal. Hasil perbandingan disajikan pada tabel 4.2.

Tabel 4.2 Hasil pengujian dan perbandingan model LLM pada sistem

Aspek	**Phi-3:mini**	**Qwen2.5:3B**	**Qwen2.5:7B**
Kualitas bahasa Indonesia	Buruk	Baik	Baik
Kepatuhan instruksi	Cukup	Buruk	Baik
Konsistensi	Cukup	Cukup	Cukup
Akurasi data	Cukup	Cukup	Baik

Berdasarkan Tabel 4.2, pemilihan model *final* jatuh pada Qwen2.5:7B karena tingkat kepatuhannya yang tinggi dalam mengeksekusi instruksi. Model ini juga menunjukkan kemampuan terbaik dalam mempertahankan konteks pada *prompt* berukuran besar (kombinasi profil pengguna dengan data JSON sepuluh lowongan dan aturan).

Meskipun eksekusi model 7B pada GPU dengan kapasitas 6 GB VRAM memicu mekanisme *offloading* memori ke RAM menyebabkan waktu tunggu berkisar 1 hingga 2 menit per respons, tingkat akurasi yang disajikan jauh lebih tinggi dibanding kandidat lainnya. Konsekuensi ini diakui sebagai *trade-off* yang dapat diterima.

4.2.4 Hasil iterasi pengembangan dan final aplikasi

Sesuai pendekatan iteratif yang dirancang pada sub-bab 3.3.4, seluruh iterasi yang muncul selama proses pengembangan didokumentasikan secara sistematis. Iterasi ini bersumber dari dua kategori: umpan balik *peer review* pada fase desain pengguna dan temuan teknis pada fase Konstruksi. Ringkasan seluruh iterasi disajikan pada Tabel 4.3.

Tabel 4.3 Hasil iterasi pengembangan aplikasi

Fase	**Sumber**	**Perubahan**
User Design	*Peer review* prototipe	Perbaikan tipografi dan hierarki visual pesan
User Design	*Peer review* prototipe	Konversi tautan menjadi hyperlink aktif
User Design	*Peer review* prototipe	Migrasi format ekspor .txt ke .pdf

User Design	*Peer review* prototipe	Penambahan efek hover pada elemen interaktif
Construction	Temuan teknis	Penyederhanaan UI dengan menghapus tombol filter manual
Construction	Temuan teknis	Penambahan mekanisme intent classification dua jalur

Poin-poin dari hasil iterasi dijelaskan sebagai berikut:

1. Iterasi penyederhanaan antarmuka

   Iterasi pertama pada fase konstruksi dipicu oleh analisis kapabilitas LLM. Model Qwen2.5:7B sudah memiliki kemampuan ekstraksi parameter melalui fungsi `extract_params ()`, sehingga tombol filter manual ("alat" yang berisi pilihan lokasi dan mode kerja) pada rancangan awal Figma dapat dihilangkan. Hasil akhir antarmuka dikunci menjadi sebuah desain dengan satu kolom masukan utama tanpa tombol filter terpisah (gambar 4.15).

2. Iterasi penambahan klasifikasi intensi

   Iterasi kedua pada fase konstruksi berupa penambahan mekanisme klasifikasi intensi dua jalur (sub-bab 4.2.2) Iterasi ini dipicu oleh observasi pada pemanggilan API yang tidak efisien ketika pengguna hanya bertanya lanjutan tentang lowongan yang sudah ditampilkan sebelumnya.

3. Realisasi empat poin *peer review*

   Keempat poin *peer review* pada Tabel 4.3 direalisasikan secara fungsional ke dalam baris kode program sebagai berikut:

   a. Realisasi format tipografi: Diimplementasikan melalui pengaturan langsung pada lapisan instruksi prompt tahap kedua. LLM diminta menampilkan nama posisi dan perusahaan dalam format tebal (*bold*) menggunakan sintaks *markdown* sederhana, dengan ikon visual sebagai penanda kategori.

   b. Realisasi tautan aktif: Pada skrip *frontend,* fungsi `formatMessage(text)` memanfaatkan ekspresi reguler `/(https?:\/\/[^\s()<>\[\]]+)/g` untuk memindai luaran teks dari LLM secara otomatis, mengekstrak teks URL mentah, dan membungkusnya ke dalam elemen *anchor* HTML.

   c. Realisasi ekspor dokumen format PDF: Sistem ekspor riwayat percakapan dibangun menggunakan `library jsPDF` yang dipicu melalui fungsi `saveInteraction()`. Kode ini memproses riwayat objek `chatHistory`, membersihkan teks dari karakter aneh menggunakan regex `/[^\x00-\x7F]/g`, dan

memformat teks ke dalam dokumen PDF formal dengan penataan batas halaman otomatis sebelum diunduh dengan nama berkas hasil_magang.pdf.

d. Efek *hover* dan *loading*: Elemen CSS Tailwind seperti `transition-colors duration-200` dipasang pada tombol, bersama dengan efek transisi *transform* `hover:scale-105`. Selain itu, saat sistem menunggu respons dari *server*, variabel `chatHistory` menyisipkan *string* khusus "`loading_state`" yang dibaca oleh fungsi `renderChat` untuk memunculkan efek animasi tiga titik melompat (`animate-bounce`).

Hasil dari perubahan dapat dilihat pada poin keempat pada sub-bab ini.

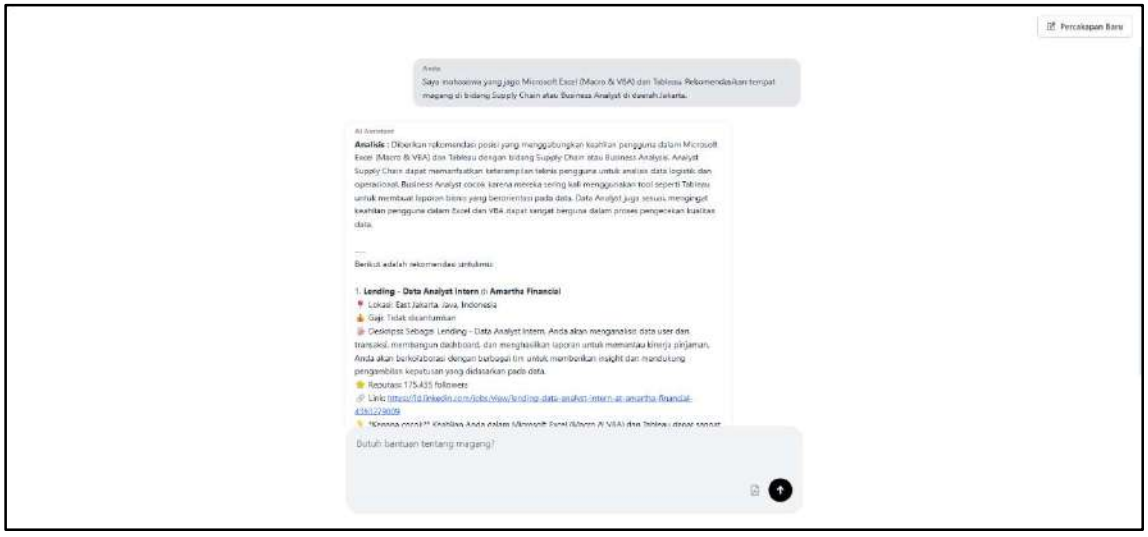
4. Dokumentasi visual aplikasi *final*

Hasil akhir dari fase konstruksi yang mengintegrasikan seluruh komponen disajikan pada Gambar 4.15 dan Gambar 4.16.



Gambar 4.15 Tampilan awal final aplikasi

Halaman awal (*welcome screen*), ketika aplikasi pertama kali dibuka di web, pengguna dihadapkan pada halaman utama minimalis dengan dominasi satu kolom input utama tanpa filter tombol manual.



Gambar 4.16 Tampilan hasil rekomendasi magang

Halaman percakapan (*chat interface),* menampilkan hasil rekomendasi final dari respons pengguna ataupun riwayat percakapan yang sudah dibuat lebih menarik. Respons yang dihasilkan mencakup analisis profil pengguna pada bagian atas, kemudian daftar rekomendasi lowongan yang dilengkapi nama posisi, perusahaan, lokasi terjemahan, gaji, deskripsi terjemahan, jumlah pengikut perusahaan, tautan lowongan yang dapat diklik, serta alasan kesesuaian masing-masing posisi.

4.2.5 Hasil analisis kendala data API eksternal

Mengacu pada potensi kendala pada sub-bab 3.3.5, pengujian dan integrasi sistem dengan API lowongan magang eksternal (Internship Listing dari fantastic.jobs) mengonfirmasi munculnya tiga kategori kendala data sebagai berikut.

1. Konfirmasi masalah data yang tidak lengkap

   Ditemukan adanya data lowongan dari API eksternal yang tidak mencantumkan variabel seperti informasi besaran gaji/insentif (`ai_salary_value`). Strategi mitigasi yang dirancang berhasil diimplementasikan: sistem mengatur agar mencantumkan kata "Tidak dicantumkan" pada komponen visual gaji/mode kerja apabila variabel pendukung pada JSON bernilai kosong, sehingga aplikasi tetap dapat menampilkan informasi apa adanya.

2. Konfirmasi keterbatasan aktualisasi indeks data

   API data lowongan hanya menyediakan listing dalam kurun waktu tujuh hari terakhir karena menggunakan versi gratis API. Selain itu, dataset mentah dari API tidak menyediakan parameter status aktif untuk menandakan apakah lowongan tersebut

masih membuka pendaftaran atau sudah ditutup. Dampaknya adalah munculnya risiko *false-positive information*, di mana sistem tetap merekomendasikan sebuah lowongan padahal kuota kandidat pada perusahaan tersebut telah terpenuhi atau proses rekrutmen telah ditutup, namun *list*ingnya belum dihapus dari indeks API. Karena keterbatasan ini berasal dari sisi penyedia API, mitigasi penuh pada aplikasi tidak dapat dilakukan.

3. Konfirmasi ketidakcocokan kata

   Pengujian menunjukkan adanya fenomena ketidakcocokan kata antara cara kerja LLM dan cara kerja API eksternal. Masalah ini muncul dalam dua bentuk utama yang masing-masing diuraikan sebagai berikut.

   a. Perbedaan penamaan lokasi, pengujian variasi penulisan nama wilayah, khususnya untuk cakupan domestik (Indonesia), mengungkapkan ketidakkonsistenan. Pengujian dilakukan langsung pada *endpoint* API melalui *dashboard* RapidAPI menunjukkan hasil yang disajikan pada Tabel 4.4.

   Tabel 4.4 Hasil pengujian variasi penamaan lokasi pada API

Input Parameter Lokasi	HTTP Status	Jumlah Hasil
Yogyakarta	200 OK	3
Jogja	200 OK	0
Daerah Istimewa Yogyakarta	200 OK	0

   Analisis terhadap atribut lokasi pada respons API mengungkapkan bahwa indeks data disimpan dalam bentuk standar "Yogyakarta dan Sekitarnya". Akibatnya, pencarian hanya berhasil jika kueri memuat "Yogyakarta". Fenomena ini membuktikan karakteristik berbasis kata kunci pada *retrieval* API, di mana input yang secara makna setara (Jogja = Yogyakarta) tetap menghasilkan data kosong.

   b. Variasi penamaan posisi/judul lowongan. Untuk mengidentifikasi lebih lanjut ketidakcocokan kata kunci ini dilakukan pengujian terhadap empat case prompt pengguna. Setiap case diuji dua kali menggunakan prompt yang sama dengan perbedaan pada lokasi, yaitu dengan tidak menggunakan lokasi dan menggunakan lokasi Indonesia. Pengujian dilakukan pada hari yang sama. Untuk dokumentasi lengkapnya, meliputi log eksekusi beserta hasil rekomendasi dalam format PDF, disertakan pada lampiran bagian Dokumentasi lainnya. Hasil pengujian ditampilkan pada tabel berikut.

   Tabel 4.5 Hasil rekomendasi dan analisis

No	**Prompt**	**Hasil**	**Analisis**
1a	Menguasai Microsoft Excel dan Tableau,	10 lowongan; ditampilkan 5	Kueri: *Business Analyst OR Data Analyst OR Excel Specialist*; tanpa

	meminta magang bidang Business Analyst	(Filipina, India, Singapura, Amerika Serikat, Prancis)	filter lokasi. Seluruh hasil berasal dari luar Indonesia
1b	Sama seperti 1a, ditambah "di Indonesia"	2 lowongan; keduanya ditampilkan (Tangerang)	Kueri: *Business Analyst OR Data Analyst OR Analyst*. Kata kunci pendek *Analyst* cocok dengan judul lowongan Indonesia yang tersedia
2a	Mahasiswa Informatika menguasai desain UI/UX	4 lowongan; seluruhnya ditampilkan (Mesir, India, Sri Lanka)	Kueri: *UI/UX Designer OR Desainer Grafis OR User Experience (UX) Specialist*. Satu kata kunci ditulis dalam Bahasa Indonesia
2b	Sama seperti 2a, ditambah "di Indonesia"	0 lowongan	Kueri: *UI/UX Designer OR Desainer Grafis OR Desainer Produk*. Dua dari tiga kata kunci berbahasa Indonesia.
3a	Mahasiswa Teknik Lingkungan, bidang Environmental Engineer	3 lowongan; ditampilkan 2 (Jakarta, Bekasi)	Kueri: *Environmental Engineer OR Sustainability OR HSE OR EHS*. Lokasi terisi Indonesia meskipun tidak disebutkan pada *prompt*. Singkatan *HSE* dan *EHS* cocok dengan judul lowongan di Indonesia
3a'	Sama seperti 3a, ditambah "di mana saja"	10 lowongan; ditampilkan 4 (Singapura, Swedia, Jerman, Filipina)	Kueri: *Environmental Engineer OR Sustainability OR HSE Specialist OR Environment Officer*; lokasi berhasil dikosongkan. Tidak ada hasil dari Indonesia
3b	Sama seperti 3a, ditambah "di Indonesia"	0 lowongan	Kueri: *Environmental Engineer OR Sustainability OR EHS Specialist*. Lowongan HSE Indonesia masih tersedia pada API (terlihat pada 3a), namun tidak tertangkap karena kata kunci berupa dua kata
4a	Mahasiswa Teknik Kimia ahli termodinamika, ditambah "di mana saja"	10 lowongan; ditampilkan 5 (Amerika Serikat, Belanda, Italia)	Kueri: *Thermodynamic Engineer OR Process Engineer OR R&D OR Research AND Development*. Tidak ada hasil dari Indonesia
4b	Sama seperti 4a, ditambah "di Indonesia"	3 lowongan; ditampilkan 2 (Medan, Tangerang)	Kueri: *Process Engineer OR R&D OR Quality Assurance OR QA*. Sebagian hasil bergeser ke bidang QA yang kurang selaras dengan termodinamika

Berdasarkan Tabel 4.5, dapat diamati bahwa, penyebab kegagalan adalah dari bentuk kata kunci. Hal ini terlihat pada pengujian 3a dan 3b. Keduanya sama-sama mencari di Indonesia dan dijalankan hanya berselang beberapa menit, tetapi 3a memperoleh tiga lowongan sedangkan 3b tidak memperoleh satu pun. Yang berbeda hanya kata kuncinya. Pengujian 3a memakai singkatan *HSE* dan *EHS*, yang cocok dengan judul lowongan yang memang ada saat itu, yaitu "HSE Intern" dan

"EHS Intern". Pengujian 3b memakai *EHS Specialist*. Karena API mencari judul yang memuat kedua kata sekaligus, judul "EHS Intern" tidak tampil meskipun mengandung kata *EHS*. Pola yang sama terlihat pada pengujian lain: 1b dan 4b berhasil karena memakai kata kunci pendek seperti *Analyst* dan *QA*, sedangkan 2b gagal karena memakai kata kunci panjang berbahasa Indonesia.

Dari temuan ini dapat disimpulkan bahwa dua strategi mitigasi yang digunakan pada aturan kedua dan ketiga instruksi `extract_params()` (Gambar 4.7), tidak sepenuhnya berhasil. Standardisasi nama lokasi terbukti mencegah kegagalan akibat penulisan yang tidak baku. Sebaliknya, perluasan variasi judul belum efektif, karena variasi yang dihasilkan LLM sering kali tidak sesuai dengan cara perusahaan di Indonesia menuliskan judul lowongannya.

4.3 Hasil tahap cutover / peralihan

Tahap evaluasi akhir dilakukan menggunakan dua metode yang telah dirancang pada Bab III, yaitu pengujian fungsional berbasis *black-box testing* dan pengujian penerimaan pengguna berbasis skala Likert

4.3.1 Hasil pengujian (*black-box testing*)

Pengujian fungsional dilakukan menggunakan matriks uji berisi 18 *test case* yang dikelompokkan ke dalam tiga kategori. Ringkasan hasil disajikan pada Tabel 4.6.

Tabel 4.6 Hasil *black-box testing* aplikasi

ID	Skenario	Status
TC-01	Ekstraksi parameter lengkap (skill, bidang, lokasi) dari *prompt*	*Passed*
TC-02	Ekstraksi dari *prompt* minimalis (lokasi & bidang implisit)	*Passed*
TC-03	Toleransi *ekstrator* terhadap kesalahan ketik *(typo)* dan bahasa kasual	*Partial*
TC-04	Keberhasilan pemanggilan API fantastic.jobs untuk mengambil data	*Passed*
TC-05	Penanganan sistem saat hasil pencarian API kosong *(no result)*	*Passed*
TC-06	Pembentukan respons rekomendasi menggunakan Bahasa Indonesia natural	*Passed*
TC-07	Pencegahan pembuatan data lowongan palsu di luar hasil API	*Passed*
TC-08	Pencegahan duplikasi lowongan yang sama dalam satu sesi respons	*Passed*
TC-09	Pemuatan elemen antarmuka utama halaman awal secara benar	*Passed*
TC-10	Validasi pencegahan pengiriman prompt kosong oleh pengguna	*Passed*
TC-11	Validasi tautan lowongan otomatis aktif dan dapat diklik pengguna	*Passed*
TC-12	Validasi fungsi ekspor hasil rekomendasi ke dalam format PDF	*Passed*
TC-13	Keberhasilan penyimpanan dan pemuatan ulang riwayat percakapan	*Passed*
TC-14	Variasi penulisan nama lokasi (contoh: DIY, Yogyakarta, Jogja)	*Failed*
TC-15	Kemampuan pemrosesan *prompt* panjang (>500 karakter) tanpa error	*Passed*
TC-16	Penanganan input campuran bahasa (Indonesia-Inggris)	*Passed*
TC-17	Penanganan permintaan di luar cakupan konteks rekomendasi magang	*Partial*
TC-18	Konsistensi rekomendasi lowongan pada pengujian *prompt* identik (3x *run*)	*Partial*

Tingkat keberhasilan kumulatif mencapai 77,8% pada penilaian *strictly passed* (14 dari 18 *test case*), atau 86,1% jika *partial passed* dihitung sebagai 0,5 (14 + 3×0,5 = 15,5 dari 18). Analisis temuan per kategori dijabarkan sebagai berikut.

1. Kategori pipeline RAG (TC-01 s.d. TC-08)

   Pengujian pada kategori ini memvalidasi performa Qwen2.5:7B dalam alur RAG aplikasi. Aplikasi berhasil mengekstrak parameter dengan akurat dari prompt lengkap (TC-01) maupun yang berisi parameter terbatas (TC-02). Pada TC-03, masukan dengan *typo* dan bahasa gaul hanya memperoleh status *partial*, karena terbatasnya kemampuan LLM dalam menebak nama lokasi yang ditulis tidak baku.

   Validasi anti-halusinasi (TC-07) dilakukan dengan membandingkan setiap lowongan pada output dengan data mentah dari API. Tidak ditemukan data palsu karena data penting seperti nama perusahaan, gaji, link lowongan, dan reputasi diambil langsung dari API dan di *render* lewat python. Pengujian bebas pengulangan (TC-08) juga lolos sepenuhnya.

2. Kategori antarmuka pengguna (TC-09 s.d. TC-13)

   Seluruh skenario pengujian pada kategori antarmuka memperoleh status *passed*. Halaman awal termuat sesuai rancangan minimalis (TC-09), validasi input kosong berfungsi melalui mekanisme di sisi klien (TC-10), serta fitur hasil iterasi seperti konversi tautan menjadi *hyperlink* (TC-11) dan ekspor PDF (TC-12) terbukti berfungsi sesuai rancangan.

3. Kategori Edge Case (TC-14 s.d. TC-18)

   Pada kategori ini, terdapat tiga temuan signifikan yang diperoleh terkait karakteristik *layer retrieval* API.

   a. TC-14 (variasi penamaan lokasi) memperoleh status *failed*. Detail temuan ini sudah dianalisis pada sub-bab 4.2.5, dan mitigasi melalui standarisasi lokasi di prompt tahap pertama.

   b. TC-17 (permintaan di luar cakupan) memperoleh status *partial*. Sistem berhasil mendeteksi sebagian besar pertanyaan di luar konteks magang dan mengalihkan ke jalur `CHAT_BIASA`, namun pada beberapa skenario tertentu sistem masih memberikan output yang berlebihan (contoh : memhasilkan tutorial di luar konteks magang).

   c. TC-18 (konsistensi *output*) memperoleh status partial. Pengujian dilakukan dengan menjalankan *prompt* identik sebanyak tiga kali. Hasilnya menunjukkan

konsistensi sempurna pada lima kriteria inti, dengan variasi *minor* pada *framing* analisis. Detail disajikan pada Tabel 4.7.

Tabel 4.7 Konsistensi output multi-eksekusi (n=3)

Aspek	**Konsistensi**
Struktur respons	100% (3/3)
Job ID rekomendasi utama	100% (3/3)
Anti-halusinasi	100% (3/3)
Anti-pengulangan	100% (3/3)
Framing analisis	0%

Variasi pada *framing* analisis merupakan karakteristik dari LLM yang beroperasi dengan nilai *temperature* di atas nol (*temperature* > 0), sehingga tidak memengaruhi validitas rekomendasi yang dihasilkan.

4.3.2 Hasil pengujian penerimaan pengguna

Pengujian pada pengguna dilakukan terhadap lima partisipan mahasiswa aktif dan nonaktif lintas jurusan (Teknik lingkungan, kimia, elektro, dan informatika) yang merepresentasikan calon pengguna. Pengujian dilakukan pada bulan Juni 2026. Setiap partisipan diminta memasukkan *prompt* sesuai profil keahlian dan memberikan penilaian pada lima indikator menggunakan skala Likert lima tingkat. Distribusi skor disajikan pada Tabel 4.8

Tabel 4.8 Hasil pengujian penerimaan pengguna

Indikator	**P1**	**P2**	**P3**	**P4**	**P5**
Kesesuaian rekomendasi	4	5	4	3	4
Kemudahan desain	5	4	4	5	5
Kualitas bahasa	5	4	5	5	5
Akseptabilitas waktu respons	4	4	3	4	3
Niat menggunakan	5	4	5	4	5

Pada indikator kesesuaian rekomendasi, sebagian besar partisipan memberikan skor empat, dengan satu partisipan memberikan skor tiga, menunjukkan hasil rekomendasi dinilai cukup sesuai dengan prompt yang dimasukkan. Namun, tiga dari lima partisipan memperoleh hasil lowongan yang relevan ketika lokasi yang dimasukkan berada di luar Indonesia. Temuan ini berkaitan dengan masalah ketidakcocokan pada proses pencocokan antara *output* LLM dengan data yang tersedia melalui API, di mana lowongan magang di Indonesia cenderung memiliki judul yang kurang spesifik dibandingkan lowongan di luar negeri, dari sisi penamaan judul. Hal ini menyebabkan pencocokan antara *prompt* pengguna dan data lowongan menjadi kurang optimal untuk konteks lowongan di Indonesia.

Indikator kemudahan desain antarmuka dan kualitas bahasa natural memperoleh skor yang konsisten (4–5), mengindikasikan antarmuka dinilai mudah digunakan dan respons sistem dinilai alami oleh hampir seluruh partisipan.

Indikator akseptabilitas waktu respons memperoleh skor terendah (3–4), dengan dua partisipan memberikan skor tiga. Hal ini dipengaruhi oleh latensi model Qwen2.5:7B yang berjalan secara lokal sehingga waktu respon relatif lebih lama dibandingkan model berbasis *cloud*.

Indikator niat menggunakan sistem, juga memperoleh skor tinggi (4–5), menunjukkan keinginan partisipan untuk terus menggunakan sistem.

BAB V
KESIMPULAN DAN SARAN

Bab ini memaparkan kesimpulan dari seluruh hasil pengembangan dan evaluasi aplikasi yang telah dilakukan pada penelitian ini, serta saran-saran konkret untuk pengembangan lanjutan berdasarkan keterbatasan dan temuan yang teridentifikasi

5.1 Kesimpulan

Berdasarkan hasil pengembangan dan evaluasi pada Bab IV, penelitian ini menjawab kedua pertanyaan penelitian pada sub-bab 1.2 sebagai berikut.

5.1.1 Aspek perancangan dan pembangunan aplikasi

Menjawab pertanyaan penelitian pertama, asisten GenAI berhasil dikembangkan melalui metode RAD (perencanaan kebutuhan, desain pengguna, konstruksi, *cutover*). Kebutuhan pengguna diidentifikasi dari survei 11 responden yang mengkonfirmasi hambatan dalam mencari tempat magang. Kemampuan analisis profil pengguna diwujudkan melalui rekayasa *prompt* dua tahap (ekstraksi parameter ke *generate* rekomendasi) dan klasifikasi intensi dua jalur (`CARI_MAGANG` dan `CHAT_BIASA`), sehingga sistem mampu memproses secara efisien.

Integrasi LLM lokal (Qwen2.5:7B *via* Ollama) dengan Internship API fantastic.jobs terbukti dapat dijalankan secara *on-premise* melalui pendekatan *tool-augmented generation*. Risiko halusinasi dimitigasi dengan strategi pembagian tugas, LLM hanya menangani pemilihan, penerjemahan, dan penalaran, sementara data krusial (perusahaan, gaji, link) di*render* langsung dari data mentah API. Hal ini terbukti efektif pada pengujian TC-07 dan TC-08. *Trade-off* yang diterima adalah latensi 1–2 menit akibat keterbatasan VRAM.

5.1.2 Aspek efektivitas fungsional dan penerimaan pengguna

Menjawab pertanyaan penelitian kedua, sistem terbukti membantu mengurangi kebingungan mahasiswa dari hasil *black-box testing* mencapai 77,8–86,1% keberhasilan dengan anti-halusinasi konsisten, dan penerimaan pengguna positif terutama pada kualitas bahasa, kemudahan antarmuka, dan niat menggunakan berkelanjutan (skor 4–5). Namun, efektivitas ini dibatasi oleh dua hal, waktu respons yang dinilai kurang cepat (skor 3–4), dan kesesuaian rekomendasi untuk lowongan lokal yang masih lebih rendah dibanding luar negeri akibat kebanyakan judul lowongan Indonesia yang kurang spesifik.

5.1.3 Kontribusi penelitian

Penelitian ini mendokumentasikan masalah ketidakcocokan kata kunci antara LLM dan API eksternal, dimana LLM memahami permintaan secara makna, sementara API mencari secara kaku berbasis kata kunci, baik pada penamaan lokasi maupun posisi pekerjaan. Temuan ini menjadi kontribusi bagi pengembangan sejenis yang mengadopsi RAG dengan API komersial sebagai data eksternal.

5.2 Saran

Berdasarkan keterbatasan dan temuan yang teridentifikasi selama proses pengembangan dan pengujian, disusun saran-saran untuk pengembang selanjutnya yang dikelompokkan sebagai berikut:

1. Membuat *database* sendiri sebagai alternatif atau pelengkap API eksternal, khususnya untuk lowongan-lowongan dari perusahaan Indonesia yang mungkin belum tercatat oleh API komersial internasional untuk memitigasi masalah API.
2. Migrasi ke versi API terbaru dan premium. Versi terbaru dari API ini akan mengupdate data setiap satu hari sehingga masalah *false-positive* karena lowongan sudah ditutup tidak terjadi untuk mengatasi masalah keterbaruan data.
3. Pengujian dengan jumlah partisipan yang lebih banyak. Ini memungkinkan validasi yang lebih kuat dibandingakan pada pengujian skala kecil.
4. Pengujian *usability* lanjutan seperti *System Usability Scale* (SUS) atau *Post-Study System Usability Questionnaire* (PSSUQ), yang memungkinkan perbandingan langsung dengan hasil pengujian aplikasi serupa di literatur.

DAFTAR PUSTAKA

Febrian, G. F., & Figueredo, G. (2024). KemenkeuGPT: Leveraging a Large Language Model on Indonesia's Government Financial Data and Regulations to Enhance Decision Making (arXiv:2407.21459). arXiv. https://doi.org/10.48550/arXiv.2407.21459

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., & Wang, H. (2024). Retrieval-Augmented Generation for Large Language Models: A Survey (arXiv:2312.10997). arXiv. https://doi.org/10.48550/arXiv.2312.10997

Kristiawan, Y. I., Maulindar, J., & Sundari, S. (2025). Sistem Rekomendasi Tempat Magang dengan Metode Algoritma A-Star (Studi Kasus: Fikom Universitas Duta Bangsa Surakarta). JATI (Jurnal Mahasiswa Teknik Informatika), 9(5), 9135–9141.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., & Rocktäschel, T. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in neural information processing systems, 33, 9459–9474.

Mulay, A., Sutar, S., Patel, J., Chhabria, A., & Mumbaikar, S. (2022). Job Recommendation System Using Hybrid Filtering. ITM Web of Conferences, 44, 02002. https://doi.org/10.1051/itmconf/20224402002

Parker, L., Vance, E., Hassan, O., & Davis, C. (2025). The Role of Generative AI in the Future of Commercial Personalization. Journal of Electronic Commerce, 1(1), 1–16.

Poncio, F. P. (2024). Navigating techniques in job recommender systems on internship profile matching: A systematic review. Journal of Research in Innovative Teaching & Learning, 17(2), 352–367. https://doi.org/10.1108/JRIT-01-2024-0016

Pranatawijaya, V. H., Widiatry, W., Priskila, R., & Putra, P. B. A. A. (2019). Penerapan Skala Likert dan Skala Dikotomi Pada Kuesioner Online. Jurnal Sains dan Informatika, 5(2), 128–137. https://doi.org/10.34128/jsi.v5i2.185

Pressman, R. S. (2010). Software engineering: A practitioner's approach (7th ed). McGraw-Hill.

Putri, Y. N. A., Efendi, N., & Roni, M. (2025). Implikasi Program Magang Merdeka Belajar-Kampus Merdeka Terhadap Daya Saing Mahasiswa Lulusan Universitas Lampung. Jurnal Kompetitif Bisnis, 3(3).

Rashid, A. H. A., Mohamad, M., Masrom, S., & Selamat, A. (2022). Student Career Recommendation System Using Content-Based Filtering Method. 2022 3rd International Conference on Artificial Intelligence and Data Sciences (AiDAS), 60–65. https://doi.org/10.1109/AiDAS56890.2022.9918766

Sahoo, P., Singh, A. K., Saha, S., Jain, V., Mondal, S., & Chadha, A. (2025). A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications (arXiv:2402.07927). arXiv. https://doi.org/10.48550/arXiv.2402.07927

Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Hambro, E., Zettlemoyer, L., Cancedda, N., & Scialom, T. (2023). Toolformer: Language models can teach themselves to use tools. Advances in neural information processing systems, 36, 68539–68551.

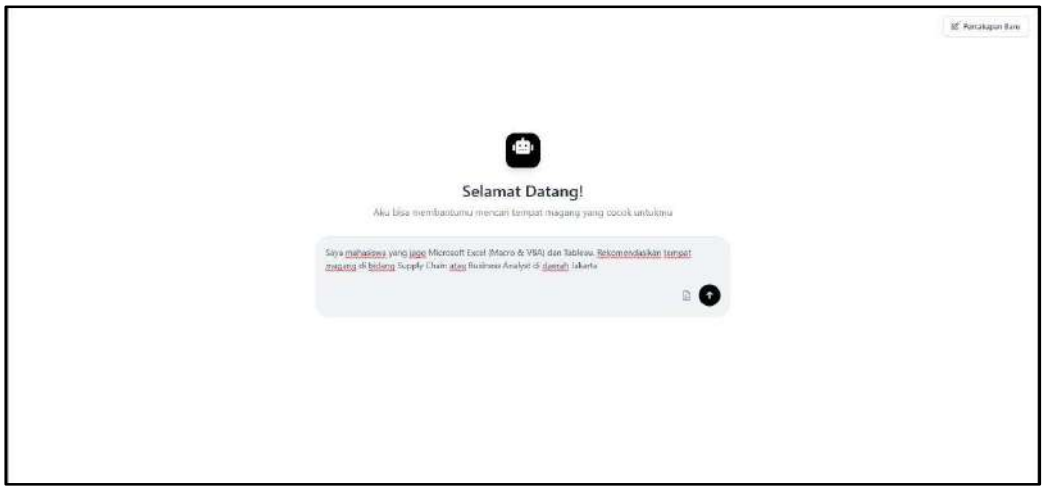
Ufia, S., Nugroho, A. D., & Wahjoedi, T. (2024). Meningkatkan Kompetensi Mahasiswa melalui Program Magang Sebagai Upaya Peningkatan Hard Skill dan Soft Skill. Journal of Knowledge and Collaboration, 1(2), 39–47. https://doi.org/10.59613/97dmmj73

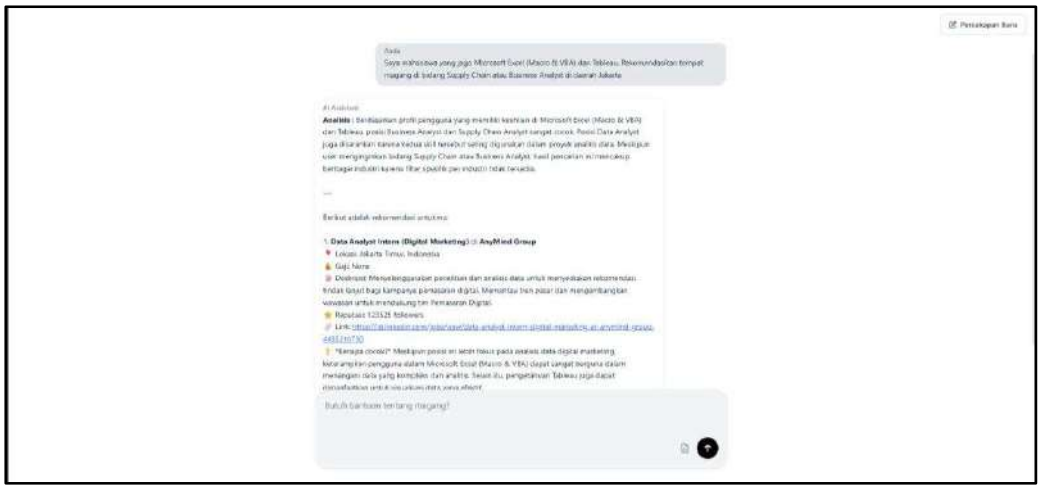
Vatsal, S., & Dubey, H. (2024). A Survey of Prompt Engineering Methods in Large Language Models for Different NLP Tasks (arXiv:2407.12994). arXiv. https://doi.org/10.48550/arXiv.2407.12994

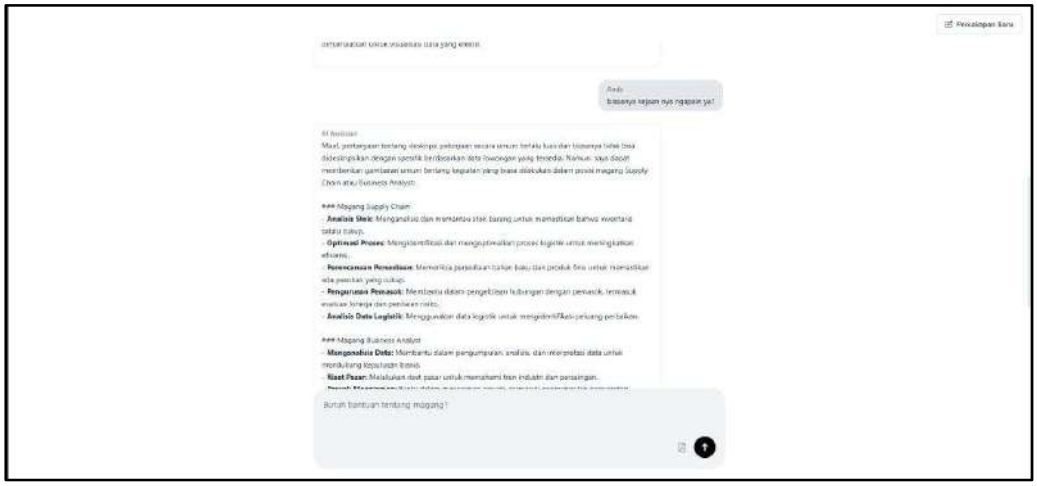
Youvan, D. (2024). The Essence of Less: Exploring Abstract and Minimalist Concepts Across Art, Design, Technology, and Philosophy. https://doi.org/10.13140/RG.2.2.21300.67205

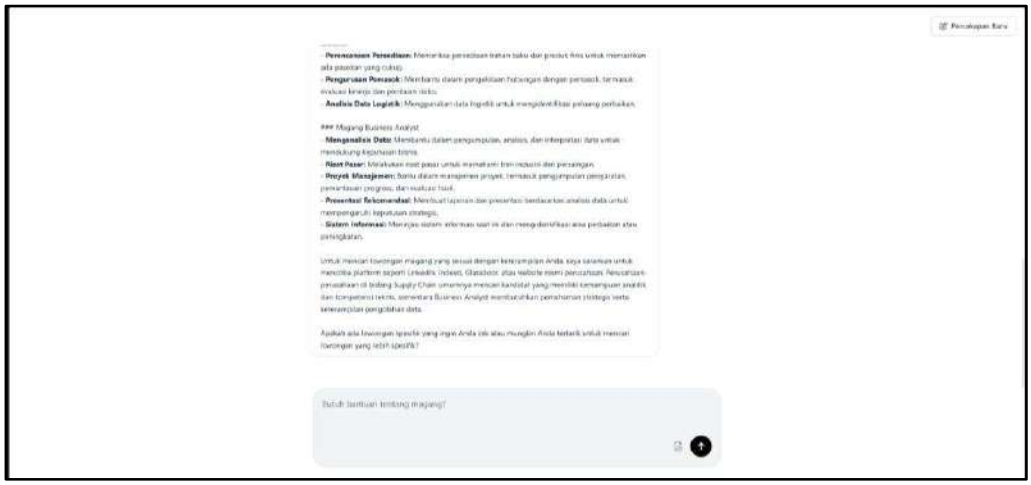
LAMPIRAN

A. Dokumentasi tampilan hasil akhir

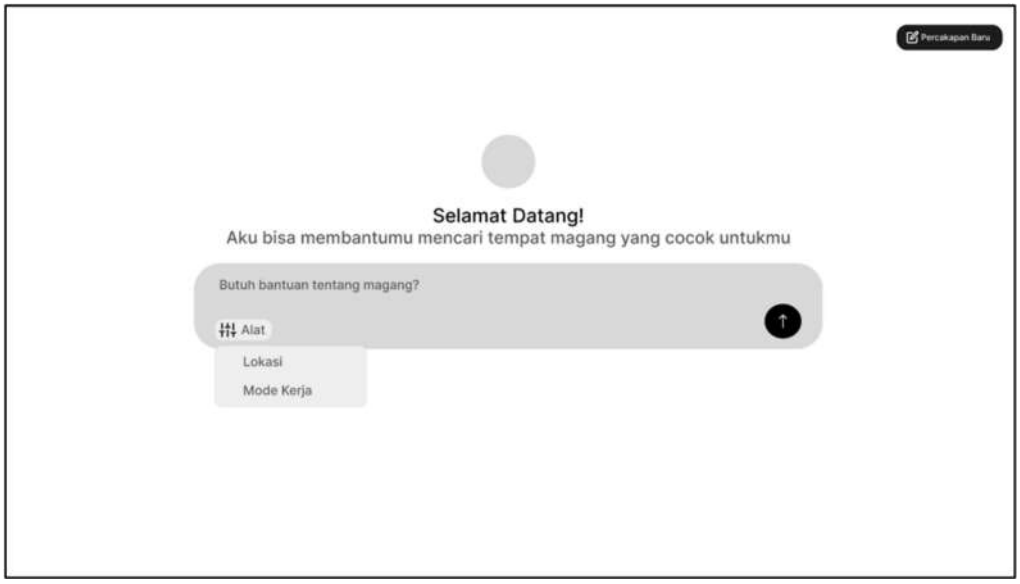








B. Tampilan desain awal



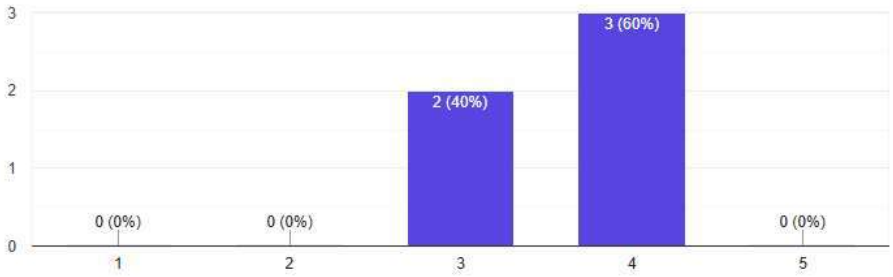


C. Hasil pengujian pengguna

Waktu tunggu respons aplikasi masih dalam batas yang dapat saya terima.

Salin diagram

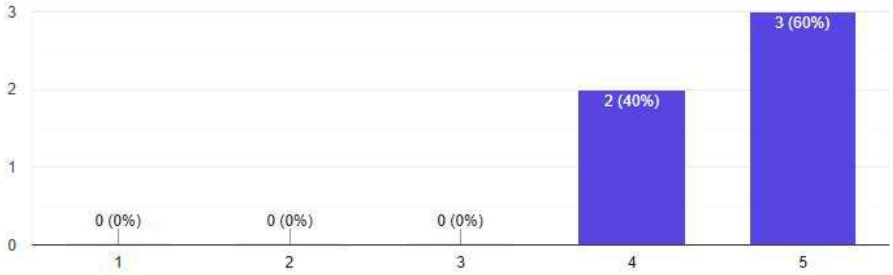
5 jawaban



Saya akan menggunakan aplikasi ini jika tersedia untuk umum.

Salin diagram

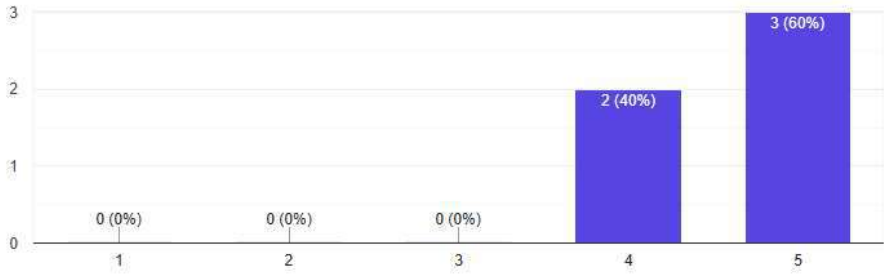
5 jawaban



Tampilan dan desain aplikasi mudah dipahami.

Salin diagram

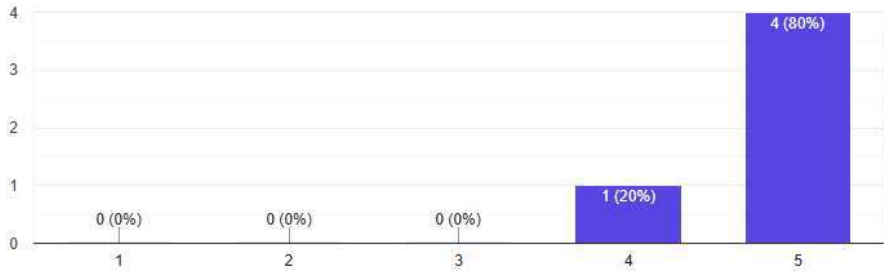
5 jawaban



Bahasa yang digunakan aplikasi terasa natural dan mudah dimengerti.

Salin diagram

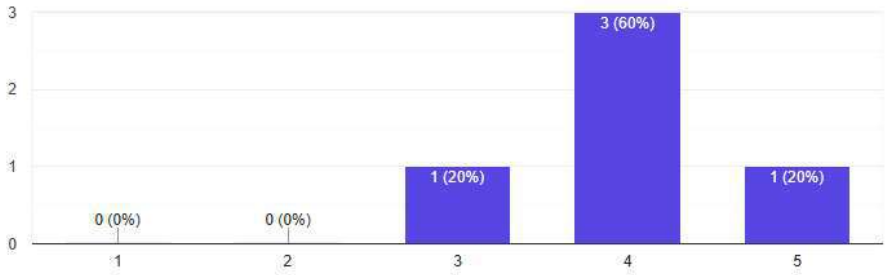
5 jawaban

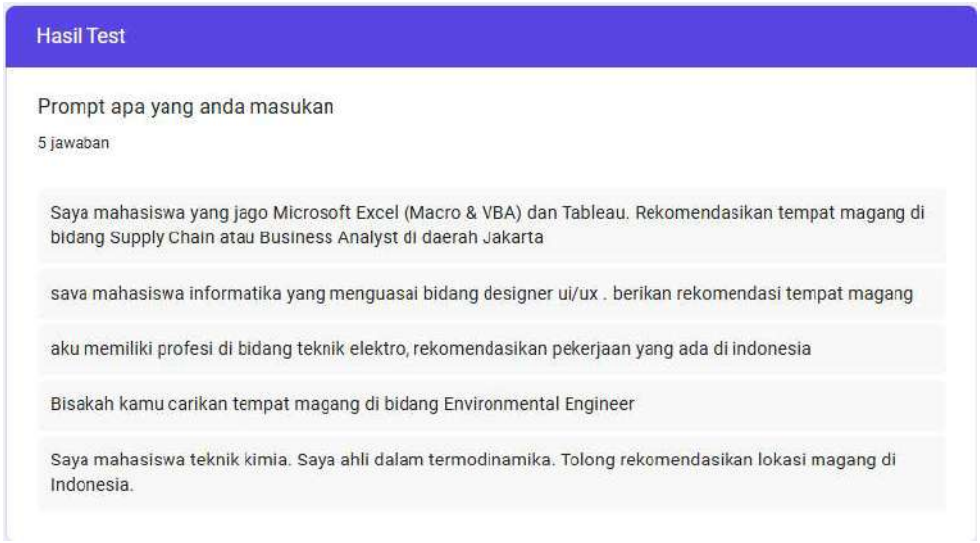


Rekomendasi magang yang diberikan sesuai dengan minat dan profil anda.

Salin diagram

5 jawaban





D. Kode app.py (backend)

```
from flask import Flask, render_template, request, jsonify, session
from flask_session import Session
from dotenv import load_dotenv
load_dotenv()
import ollama
import json
import requests
import re
import os
import logging
from datetime import timedelta



logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s - %(levelname)s - %(message)s',
    handlers=[
        logging.FileHandler('app.log'),
        logging.StreamHandler()
    ]
)
logger = logging.getLogger(__name__)

app = Flask(__name__)

app.config['SECRET_KEY'] = os.urandom(24)
app.config['SESSION_TYPE'] = 'filesystem'
app.config['SESSION_FILE_DIR'] = './flask_sessions'
app.config['SESSION_PERMANENT'] = False
app.config['SESSION_USE_SIGNER'] = True
app.config['PERMANENT_SESSION_LIFETIME'] = timedelta(hours=2)

os.makedirs('./flask_sessions', exist_ok=True)

Session(app)

MODEL_NAME = "qwen2.5:7b"

RAPIDAPI_KEY = os.environ.get('key', '')
RAPIDAPI_HOST = "internships-api.p.rapidapi.com"
```

```
API_URL = f"https://internships-api.p.rapidapi.com/active-jb-7d"

if not RAPIDAPI_KEY:
    logger.warning("KEY tidak ada!")


def classify_intent(user_input):
    prompt = f"""
    Tugas: Klasifikasikan intensi pesan dari pengguna berikut.
    Pesan: "{user_input}"

    Pilihlah satu dari dua kategori berikut:
    1. "CARI_MAGANG": Jika pengguna menceritakan profilnya, mencari lowongan
baru, menyebutkan skill baru untuk dicari, atau meminta rekomendasi magang
secara eksplisit.
    2. "CHAT_BIASA": Jika pengguna menyapa, berterima kasih, bertanya tentang
tips umum, atau menanyakan detail/pertanyaan lanjutan terkait lowongan yang
sebelumnya sudah diberikan AI.

    Output HANYA kata "CARI_MAGANG" atau "CHAT_BIASA" tanpa penjelasan apa
pun.
    """
    try:
        response = ollama.chat(
            model=MODEL_NAME,
            messages=[{"role": "user", "content": prompt}]
        )
        intent = response['message']['content'].strip()
        logger.info(f"Intent classified: '{user_input[:50]}...' → {intent}")

        if "CARI_MAGANG" in intent:
            return "CARI_MAGANG"
        return "CHAT_BIASA"
    except Exception as e:
        logger.error(f"Intent classification failed: {e}")
        return "CARI_MAGANG"


def extract_params(user_story):
    prompt = f"""
    Tugas: Analisis profil user dan berikan rekomendasi posisi magang.
    Profil User: "{user_story}"

    Aturan:
    1. 'rekomendasi_posisi': List 3 posisi yang cocok dengan skill user.
    2. 'api_query': Buat string pencarian menggunakan operator OR.
        PENTING - Pertimbangkan variasi penamaan di industri Indonesia:
        - Title utama yang paling cocok
        - 2-3 sinonim atau title terkait
        - 1-2 variasi konvensi industri (formal/informal)
        - GUNAKAN 4-6 variasi total
        - JANGAN masukkan nama industri (FMCG, Banking, dll) ke api_query,
          karena ini bukan nama posisi

        Contoh untuk "ingin magang sebagai pembuat konten":
        "Content Creator" OR "Social Media" OR "Digital Marketing" OR
        "Multimedia" OR "Creative"

        Contoh untuk "mahasiswa Teknik Lingkungan":
        "Environmental" OR "Sustainability" OR "HSE" OR "EHS" OR
        "Safety" OR "Environment Officer"

        Contoh untuk "ingin magang analisis data":
        "Data Analyst" OR "Business Analyst" OR "Data Science" OR
        "Analytics" OR "Business Intelligence"

    3. 'location': Lokasi pencarian yang dapat berupa nama kota
```

```
       atau nama negara, dalam format kanonikal.
       PENTING - Gunakan nama resmi yang dikenali API:
       - Untuk kota:
         * "Yogyakarta" (BUKAN Jogja, DIY)
         * "Jakarta" (BUKAN JKT, DKI)
         * "Surabaya" (BUKAN SBY)
       - Untuk negara:
         * "Indonesia" (BUKAN ID, INA)
         * "United States" (BUKAN US, USA)
         * "United Kingdom" (BUKAN UK)
       - Default null jika user tidak menyebutkan lokasi spesifik


    4. 'is_remote': true jika user menyebut remote/WFH/work from home.

    Output JSON saja:
    {{
        "rekomendasi_posisi": [],
        "api_query": "...",
        "location": "...",
        "is_remote": false,
        "alasan": "Penjelasan rekomendasi posisi dan kenapa cocok dengan
profil pengguna. Jika user menyebut industri spesifik, jelaskan bahwa hasil
mencakup berbagai industri karena filter spesifik per industri tidak tersedia."
    }}
    """
    try:
        response = ollama.chat(
            model=MODEL_NAME,
            messages=[{"role": "user", "content": prompt}]
        )
        content = response['message']['content'].strip()
        output_json = re.search(r'\{.*\}', content, re.DOTALL)
        result = json.loads(output_json.group()) if output_json else
json.loads(content)

        logger.info(f"Extracted params: query='{result.get('api_query')}', "
                    f"location='{result.get('location')}',
remote={result.get('is_remote')}")

        return result
    except Exception as e:
        logger.error(f"Parameter extraction failed: {e}")
        return {
            "rekomendasi_posisi": ["General Intern"],
            "api_query": "Internship",
            "location": None,
            "is_remote": False,
            "alasan": "Gagal menganalisis secara mendalam, mencari posisi
umum."
        }




def call_api(params):
    is_remote = params.get('is_remote', False)
    if isinstance(is_remote, bool):
        remote_str = "true" if is_remote else "false"
    else:
        remote_str = str(is_remote).lower()

    api_params = {
        "title_filter": params.get('api_query'),
        "location_filter": params.get('location'),
        "remote": remote_str,
        "include_ai": "true",
```

```
        "limit": 10
    }


    api_params = {k: v for k, v in api_params.items() if v is not None}

    headers = {
        "X-RapidAPI-Key": RAPIDAPI_KEY,
        "X-RapidAPI-Host": RAPIDAPI_HOST
    }

    try:
        logger.info(f"Calling API with params: {api_params}")
        r   =   requests.get(API_URL,   params=api_params,   headers=headers,
timeout=20)

        if r.status_code == 200:
            data = r.json()
            jobs = data if isinstance(data, list) else data.get('data', [])

            # ==================== MODIFIKASI: LOGGING HASIL PENCARIAN API
====================
            logger.info("=== SEARCH QUERY RESULT FROM API ===")
            logger.info(f"Total Jobs Returned: {len(jobs)}")
            if jobs:
                for idx, job in enumerate(jobs):
                    logger.info(f" [{idx+1}] Title: {job.get('title')} | Org:
{job.get('organization')} | Location: {job.get('locations_derived')}")
            else:
                logger.warning("Tidak ada lowongan yang cocok dari API.")
            logger.info("====================================")
            #
================================================================================
=====

            return jobs
        else:
            logger.error(f"API   error:   status   {r.status_code},   response:
{r.text[:200]}")
            return []
    except requests.Timeout:
        logger.error("API request timeout")
        return []
    except Exception as e:
        logger.error(f"API call failed: {e}")
        return []

def location_fix(job):
    locations = job.get("locations_derived")
    if isinstance(locations, list) and len(locations) > 0:
        return locations[0]
    elif isinstance(locations, str):
        return locations
    return "Tidak dicantumkan"


@app.route('/')
def index():
    session.clear()
    logger.info("Session cleared, new session started")
    return render_template('index.html')


@app.route('/chat', methods=['POST'])
def chat():
    user_input = request.json.get('message')
    chat_history = request.json.get('history', [])
```

```
    if not user_input:
        return jsonify({"reply": "Silakan ceritakan latar belakangmu."})

    logger.info(f"User input: '{user_input[:100]}...'")


    intent = classify_intent(user_input)


    if intent == "CHAT_BIASA":
        job_sesi = session.get('job_sesi', [])
        logger.info(f"🔍 CHAT_BIASA session data:")
        logger.info(f"   Total jobs in session: {len(job_sesi)}")
        for i, job in enumerate(job_sesi):
            logger.info(f"   Job {i}: title={job.get('title')}, "
                        f"org={job.get('organization')}")

        messages = []
        for msg in chat_history[-6:]:
            messages.append({"role": msg['role'], "content": msg['content']})

        context = f"""Kamu adalah AI Internship Assistant yang ramah dan
profesional,
            khusus membantu mahasiswa Indonesia mencari informasi tentang
magang.

            KONTEKS DATA LOWONGAN TERAKHIR YANG DIBERIKAN:
            {json.dumps(job_sesi, ensure_ascii=False, indent=2)}

            ATURAN PENTING (WAJIB DIPATUHI):

            1. ATURAN ANTI-HALUSINASI (PALING UTAMA):
            - HANYA gunakan informasi dari "DATA LOWONGAN TERAKHIR" di atas
            - JANGAN PERNAH mengarang nama perusahaan, posisi, gaji, lokasi,
                atau detail lain yang TIDAK ADA dalam data tersebut
            - JANGAN PERNAH menambah lowongan baru dari pengetahuan umummu
            - Jika user menanyakan lowongan yang TIDAK ADA dalam data,
                jawab dengan jujur dan tawarkan pencarian baru

            2. ATURAN PENGGUNAAN DATA:
            - Jika user bertanya tentang lowongan spesifik, cari di data
                dan sebutkan detail SESUAI data yang ada
            - Jika user bertanya perbandingan, bandingkan ANTAR LOWONGAN
                YANG ADA, berikan alasan berdasarkan informasi tersedia

            3. ATURAN HANDLE DATA KOSONG:
            - Jika data lowongan kosong, jawab dengan menawarkan pencarian
baru

            4. ATURAN SCOPE:
            - Topik yang BOLEH: pencarian magang, rekomendasi karir, tips
                melamar magang, persiapan CV/wawancara
            - Topik yang HARUS dialihkan: pertanyaan di luar konteks
magang/karir
            - Format pengalihan: "Maaf, saya khusus membantu pencarian dan
                persiapan magang. Apakah ada yang bisa saya bantu terkait
magang?"

            5. ATURAN FORMAT:
            - Bahasa Indonesia yang ramah dan profesional
            - Format markdown yang rapi
            - Jika mereferensikan lowongan, sebutkan nama posisi dan
perusahaan
            - Jangan terlalu panjang—to-the-point
```

```
            PRINSIP   UTAMA:   LEBIH   BAIK   MENGAKUI   TIDAK   TAHU   DARIPADA
MENGARANG."""

        messages.insert(0, {"role": "system", "content": context})
        messages.append({"role": "user", "content": user_input})

        try:
            res = ollama.chat(model=MODEL_NAME, messages=messages)
            return jsonify({"reply": res['message']['content']})
        except Exception as e:
            logger.error(f"CHAT_BIASA response failed: {e}")
            return jsonify({"reply": f"Terjadi kesalahan pada AI: {str(e)}"})

    else:
        analisis = extract_params(user_input)
        jobs = call_api(analisis)

        if not jobs:
            logger.info("No jobs found for query")
            return jsonify({
                "reply": f"**Analisis Profil:**\n{analisis['alasan']}\n\n"
                         f"Maaf, tidak ditemukan lowongan yang cocok saat ini.
"
                         f"Hal   ini   dapat   terjadi   karena   keterbatasan
ketersediaan "
                         f"data lowongan dalam 7 hari terakhir."
            })

        session['job_sesi'] = jobs[:10]
        logger.info(f"Saved {len(jobs[:10])} jobs to session")

        list_jobs = []
        for idx, job in enumerate(jobs[:10]):
            list_jobs.append({
                "index": idx,
                "title": job.get("title", ""),
                "company": job.get("organization", ""),
                "location": location_fix(job),
                "description":     job.get("ai_core_responsibilities")     or
job.get("description", "")
            })

        final_prompt = f"""
        PROFIL USER: {user_input}
        DATA LOWONGAN: {json.dumps(list_jobs)}

        TUGAS:
        Pilih maksimal 5 lowongan dari DATA LOWONGAN yang paling cocok dengan
PROFIL USER.
        Terjemahkan informasi lokasi dan ringkasan deskripsi lowongan pilihan
tersebut ke dalam Bahasa Indonesia yang baik dan profesional.

        ATURAN STRUKTUR OUTPUT (WAJIB):
        Anda HARUS merespon dalam format JSON bersih dengan struktur seperti
berikut tanpa tambahan teks penjelas apa pun:
        {{
            "rekomendasi": [
                {{
                    "index": 0,
                    "lokasi_indo":   "Hasil   terjemahan   lokasi   ke   Bahasa
Indonesia (Contoh: 'Jakarta, Indonesia' atau 'Remote')",
                    "deskripsi_indo": "Ringkasan pendek deskripsi pekerjaan
yang sudah diterjemahkan ke Bahasa Indonesia maksimal 2 kalimat",
                    "alasan_cocok": "Penjelasan singkat menggunakan Bahasa
Indonesia kenapa posisi ini cocok dengan profil user"
                }}
            ]
```

```
        }}
        Tampilkan maksimal 5 rekomendasi. Jika hanya ada 1 data lowongan,
berikan 1 saja di dalam list.
        """

        try:
            res = ollama.chat(
                model=MODEL_NAME,
                messages=[{"role": "user", "content": final_prompt}]
            )
            ai_respons = res['message']['content'].strip()

            output_json = re.search(r'\{.*\}', ai_respons, re.DOTALL)
            hasil_rekomendasi     =     json.loads(output_json.group())     if
output_json else json.loads(ai_respons)

            output_final   =   f"**Analisis   :**   {analisis['alasan']}\n\n---
\n\nBerikut adalah rekomendasi untukmu:\n\n"

            rekomendasi_list = hasil_rekomendasi.get("rekomendasi", [])
            for i, recc in enumerate(rekomendasi_list):
                idx = int(recc.get("index", 0))

                if idx >= len(jobs):
                    continue

                job_api = jobs[idx]

                posisi = job_api.get("title", "Tidak dicantumkan")
                perusahaan = job_api.get("organization", "Tidak dicantumkan")
                gaji = job_api.get("ai_salary_value") or "Tidak dicantumkan"
                reputasi = job_api.get("linkedin_org_followers") or "0"
                link_asli = job_api.get("url", "#")

                lokasi_id = recc.get("lokasi_indo", location_fix(job_api))
                deskripsi_id          =          recc.get("deskripsi_indo",
job_api.get("ai_core_responsibilities", "Tidak dicantumkan"))
                alasan  =  recc.get("alasan_cocok",  "Cocok  dengan  kualifikasi
Anda.")

                output_final += f"{i+1}. **{posisi}** di **{perusahaan}**\n"
                output_final += f"    Lokasi: {lokasi_id}\n"
                output_final += f"    Gaji: {gaji}\n"
                output_final += f"    Deskripsi: {deskripsi_id}\n"
                output_final += f"    Reputasi: {reputasi} followers\n"
                output_final += f"    Link: {link_asli}\n"
                output_final += f"    *Kenapa cocok?* {alasan}\n\n"

            logger.info(f"Generated {len(rekomendasi_list)} recommendations")
            return jsonify({"reply": output_final})

        except json.JSONDecodeError as e:
            logger.error(f"JSON parse error: {e}")
            return jsonify({"reply": f"Terjadi kesalahan format AI, silakan
coba lagi."})
        except Exception as e:
            logger.error(f"CARI_MAGANG processing failed: {e}")
            return jsonify({"reply": f"Terjadi kesalahan format AI, silakan
coba lagi. (Error: {str(e)})"})



if __name__ == '__main__':
    app.run(debug=True)
```

E. Kode tampilan (index.html)

```
<!doctype html>
<html lang="id">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>AI Internship Search</title>
    <script src="https://cdn.tailwindcss.com"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/jspdf/2.5.1/jspdf.umd.min.js"></
script>
    <link
      rel="stylesheet"
      href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.0.0/css/all.min.css"
    />
    <style>
      #scroll-area {
        flex: 1;
        overflow-y: auto;
        scroll-behavior: smooth;
        width: 100%;
      }

      #scroll-area::-webkit-scrollbar {
        width: 6px;
      }
      #scroll-area::-webkit-scrollbar-thumb {
        background: #e5e7eb;
        border-radius: 10px;
      }

      .chat-container {
        width: 100%;
        max-width: 48rem;
        margin: 0 auto;
        padding: 20px 0;
      }

      .chat-link {
        color: #2563eb;
        text-decoration: underline;
      }

      #user-input {
        min-height: 56px;
        max-height: 150px;
        overflow-y: hidden;
      }

      .delay-100 {
        animation-delay: 0.1s;
      }
      .delay-200 {
        animation-delay: 0.2s;
      }
    </style>
  </head>
  <body
    class="bg-white text-gray-800 font-sans h-screen flex flex-col overflow-
hidden"
  >
    <header class="p-4 flex justify-end shrink-0">
      <button
        onclick="resetChat()"
```

```
        class="flex items-center gap-2 border border-gray-300 rounded-lg px-4
py-2  text-sm  font-medium  text-gray-700  hover:bg-black  hover:text-white
hover:border-black transition-colors duration-200"
      >
        <i class="fa-regular fa-pen-to-square"></i>
        <span>Percakapan Baru</span>
      </button>
    </header>

    <main class="flex-1 flex flex-col overflow-hidden relative">
      <div id="scroll-area">
        <div
          id="dynamic-content-wrapper"
          class="w-full h-full flex flex-col items-center justify-center px-4
pb-32"
        >
          <div id="welcome-section" class="text-center mb-8 shrink-0">
            <div
              class="w-16 h-16 bg-black text-white rounded-2xl mx-auto mb-6
flex  items-center  justify-center  shadow-lg  transform  hover:scale-105
transition-transform duration-300"
            >
              <i
                class="fa-solid fa-robot text-3xl animate-bounce"
                style="animation-duration: 3s"
              ></i>
            </div>
            <h1 class="text-3xl font-semibold mb-2">Selamat Datang!</h1>
            <p class="text-gray-500 text-lg">
              Aku bisa membantumu mencari tempat magang yang cocok untukmu
            </p>
          </div>

          <div  id="chat-display"  class="chat-container  hidden  space-y-
6"></div>

          <div id="input-initial-wrapper" class="w-full max-w-3xl">
            <div id="input-area" class="w-full pb-8">
              <div
                class="bg-gray-100 rounded-3xl p-4 shadow-sm border border-
gray-200"
              >
                <textarea
                  id="user-input"
                  rows="2"
                  oninput="autoResize(this)"
                  onkeydown="handleKeyPress(event)"
                  class="w-full  bg-transparent  border-none  focus:ring-0
focus:outline-none resize-none text-gray-700 placeholder-gray-500"
                  placeholder="Butuh bantuan tentang magang?"
                ></textarea>

                <div class="flex justify-end items-center mt-2 gap-2">
                  <button
                    onclick="saveInteraction()"
                    title="Simpan PDF"
                    class="text-gray-400 hover:text-red-500 p-2"
                  >
                    <i class="fa-regular fa-file-pdf text-lg"></i>
                  </button>
                  <button
                    onclick="sendMessage()"
                    class="bg-black text-white rounded-full w-10 h-10 flex
items-center justify-center hover:bg-gray-800 transition"
                  >
                    <i class="fa-solid fa-arrow-up"></i>
                  </button>
```

```
              </div>
            </div>
          </div>
        </div>

        <div id="scroll-spacer" class="h-10 hidden shrink-0"></div>
      </div>
    </div>

    <div
      id="sticky-input-container"
      class="w-full flex justify-center px-4 bg-white shrink-0 hidden"
    >
      <div class="w-full max-w-3xl"></div>
    </div>
  </main>

  <script>
    let chatHistory = [];

    function formatMessage(text) {
      let cleanText = text.replace(/\[(https?:\/\/[^\]]+)\]/g, "$1");
      const urlRegex = /(https?:\/\/[^\s()<>\[\]]+)/g;

      return cleanText
        .replace(urlRegex, (url) => {
          const cleanUrl = url.replace(/[\.,\?!\)\]]+$/, "");
          return `<a href="${cleanUrl}" target="_blank" class="chat-
link">${cleanUrl}</a>`;
        })
        .replace(/\*\*(.*?)\*\*/g, "<strong>$1</strong>");
    }

    function handleKeyPress(event) {
      if (event.key === "Enter" && !event.shiftKey) {
        event.preventDefault();
        sendMessage();
      }
    }

    function autoResize(textarea) {
      textarea.style.height = "auto";
      textarea.style.height = textarea.scrollHeight + "px";
      textarea.style.overflowY =
        textarea.scrollHeight > 150 ? "auto" : "hidden";
    }

    async function sendMessage() {
      const inputField = document.getElementById("user-input");
      const message = inputField.value.trim();

      if (message) {
        const welcome = document.getElementById("welcome-section");
        const chatDisplay = document.getElementById("chat-display");
        const dynamicWrapper = document.getElementById(
          "dynamic-content-wrapper",
        );
        const spacer = document.getElementById("scroll-spacer");
        const inputArea = document.getElementById("input-area");
        const stickyContainer = document.getElementById(
          "sticky-input-container",
        );
        const stickyTarget = stickyContainer.querySelector("div");

        if (!welcome.classList.contains("hidden")) {
          welcome.classList.add("hidden");
          chatDisplay.classList.remove("hidden");
```

```
            spacer.classList.remove("hidden");
            dynamicWrapper.classList.remove(
              "justify-center",
              "h-full",
              "pb-32",
            );
            stickyContainer.classList.remove("hidden");
            stickyTarget.appendChild(inputArea);
          }

          chatHistory.push({ role: "user", content: message });
          renderChat();

          inputField.value = "";
          inputField.style.height = "auto";

          const loadingIndex = chatHistory.length;
          chatHistory.push({
            role: "ai",
            content: "loading_state",
          });
          renderChat();

          try {
            const response = await fetch("/chat", {
              method: "POST",
              headers: { "Content-Type": "application/json" },
              body: JSON.stringify({
                message: message,
                history: chatHistory.slice(0, -1),
              }),
            });
            const data = await response.json();
            chatHistory[loadingIndex].content = data.reply;
            renderChat();
          } catch (error) {
            chatHistory[loadingIndex].content =
              "Maaf, terjadi kesalahan saat menghubungi server.";
            renderChat();
          }
        }
      }

      function renderChat() {
        const display = document.getElementById("chat-display");
        const scrollArea = document.getElementById("scroll-area");

        display.innerHTML = chatHistory
          .map((msg) => {
            let messageContent = "";
            if (msg.content === "loading_state") {
              messageContent = `
                  <div class="flex items-center gap-1 py-1 px-2">
                    <div  class="w-2  h-2  bg-gray-500  rounded-full  animate-
bounce"></div>
                    <div  class="w-2  h-2  bg-gray-500  rounded-full  animate-
bounce delay-100"></div>
                    <div  class="w-2  h-2  bg-gray-500  rounded-full  animate-
bounce delay-200"></div>
                  </div>
                `;
            } else {
              messageContent = `<p style="white-space: pre-line;" class="text-
sm leading-relaxed">${formatMessage(msg.content)}</p>`;
            }

            let roleLabel = msg.role === "user" ? "Anda" : "AI Assistant";
```

```
            return `
                <div class="flex ${msg.role === "user" ? "justify-end" :
"justify-start"}">
                    <div class="${msg.role === "user" ? "bg-gray-200 text-
gray-800" : "bg-white text-gray-800 border border-gray-100 shadow-sm"}
rounded-2xl px-4 py-2 max-w-[85%]">
                        <p class="text-xs font-bold mb-1 opacity-
50">${roleLabel}</p>
                        ${messageContent}
                    </div>
                </div>
            `;
          })
          .join("");

        setTimeout(() => {
          scrollArea.scrollTop = scrollArea.scrollHeight;
        }, 50);
      }

      function saveInteraction() {
        if (chatHistory.length === 0)
          return alert("Belum ada percakapan untuk disimpan.");

        const { jsPDF } = window.jspdf;
        const doc = new jsPDF();

        let y = 20;
        const margin = 15;
        const pageWidth = 180;
        const lineHeight = 7;

        doc.setFont("helvetica", "bold");
        doc.setFontSize(16);
        doc.text("Laporan Rekomendasi Magang", margin, y);
        y += 12;

        chatHistory.forEach((msg) => {
          if (msg.content === "loading_state") return;
          let cleanContent = msg.content
            .replace(/\*\*/g, "")
            .replace(/[^\x00-\x7F]/g, "")
            .replace(/\n\s*\n/g, "\n");

          const roleLabel = msg.role === "user" ? "ANDA:" : "AI ASSISTANT:";

          doc.setFont("helvetica", "bold");
          doc.setFontSize(10);
          doc.text(roleLabel, margin, y);
          y += 6;

          doc.setFont("helvetica", "normal");
          const lines = doc.splitTextToSize(cleanContent, pageWidth);

          if (y + lines.length * lineHeight > 275) {
            doc.addPage();
            y = 20;
          }

          doc.text(lines, margin, y);
          y += lines.length * lineHeight + 8;
        });

        doc.save("hasil_magang.pdf");
      }
```

```
      function resetChat() {
        chatHistory = [];
        const welcome = document.getElementById("welcome-section");
        const chatDisplay = document.getElementById("chat-display");
        const dynamicWrapper = document.getElementById(
          "dynamic-content-wrapper",
        );
        const spacer = document.getElementById("scroll-spacer");
        const inputArea = document.getElementById("input-area");
        const   initialWrapper   =   document.getElementById("input-initial-
wrapper");
        const stickyContainer = document.getElementById(
          "sticky-input-container",
        );
        const inputField = document.getElementById("user-input");

        welcome.classList.remove("hidden");
        chatDisplay.classList.add("hidden");
        spacer.classList.add("hidden");
        chatDisplay.innerHTML = "";

        dynamicWrapper.classList.add("justify-center", "h-full", "pb-32");
        stickyContainer.classList.add("hidden");
        initialWrapper.appendChild(inputArea);

        inputField.value = "";
        inputField.style.height = "auto";

        fetch("/");
      }
    </script>
  </body>
</html>
```

F. Data mentah dari API lengkap


```
[
  {
    "id": "2286325067",
    "date_posted": "2026-07-30T12:08:26",
    "date_created": "2026-07-30T14:25:28.639179",
    "title": "IT Programmer Intern",
    "organization": "Dynapack Asia",
    "organization_url": "https://www.linkedin.com/company/dynapack-asia",
    "date_validthrough": "2026-08-27T14:22:27",
    "locations_raw": [
      {
        "@type": "Place",
        "address": {
          "@type": "PostalAddress",
          "addressCountry": "ID",
          "addressLocality": "Kota Tangerang",
          "addressRegion": null,
          "streetAddress": null
        },
        "latitude": -6.189195,
        "longitude": 106.64075
      }
    ],
    "location_type": null,
    "location_requirements_raw": null,
    "salary_raw": null,
    "employment_type": [
      "INTERN"
    ],
```

```
    "url":        "https://id.linkedin.com/jobs/view/it-programmer-intern-at-
dynapack-asia-4444598040",
    "source_type": "jobboard",
    "source": "linkedin",
    "source_domain": "id.linkedin.com",
    "organization_logo":
"https://media.licdn.com/dms/image/v2/C560BAQEdRTkp71oj_w/company-
logo_200_200/company-
logo_200_200/0/1630655174208/dynapack_asia_logo?e=2147483647&v=beta&t=zTTreG
Ug0WIgBrckN-A__bKKmYOUp-RRA78u6ItorKc",
    "cities_derived": [
      "Tangerang"
    ],
    "counties_derived": null,
    "regions_derived": [
      "Banten"
    ],
    "countries_derived": [
      "Indonesia"
    ],
    "locations_derived": [
      "Tangerang, Banten, Indonesia"
    ],
    "timezones_derived": [
      "Asia/Jakarta"
    ],
    "lats_derived": [
      -6.16996365
    ],
    "lngs_derived": [
      106.64029651490846
    ],
    "remote_derived": false,
    "recruiter_name": null,
    "recruiter_title": null,
    "recruiter_url": null,
    "linkedin_org_employees": 185,
    "linkedin_org_url": "http://www.dynapackasia.com",
    "linkedin_org_size": "5,001-10,000 employees",
    "linkedin_org_slogan": "#dynapackasia",
    "linkedin_org_industry": "Plastics Manufacturing",
    "linkedin_org_followers": 7588,
    "linkedin_org_headquarters": "Karawaci, Tangerang, Banten",
    "linkedin_org_type": "Partnership",
    "linkedin_org_foundeddate": "1959",
    "linkedin_org_specialties": [
      ""
    ],
    "linkedin_org_locations": [
      "Dynaplast Tower 9-10th Floor Jl. M.H. Thamrin No. 1 Lippo Karawaci,
Karawaci, Tangerang, Banten 15811, ID"
    ],
    "linkedin_org_description": "DYNAPACK ASIA group of companies is Asia’s
leading dynamic packaging and plastic total solutions provider. We are growing
regionally through a solid foundation of organic growth and acquisition of
leading companies. We continuously develop our capabilities to supply our
products to local and regional customers, including leading multinationals
operating throughout Asia. We have the ‘best of breed’ management disciplines
and expertise that propels us to become one of the most recognized and respected
plastic manufacturer in Asia. We have 10 companies within the DYNAPACK ASIA
group, across 5 countries (Indonesia, Thailand, Vietnam, Malaysia, and China),
with a total of 33 manufacturing plants. We are also committed to lead through
innovation with a dedicated innovation center driving the competitive advantage
for our group of companies.\n\nConnect with us in our Social Medias:\nLinkedIn
:  DYNAPACK  ASIA\nInstagram    :  DYNAPACK  ASIA\nFacebook      :  DYNAPACK
ASIA\n\nRECRUITMENT EMAIL:\nrecruitment@dynapackasia.com",
    "linkedin_org_recruitment_agency_derived": false,
```

```
    "seniority": "Tidak Berlaku",
    "directapply": false,
    "linkedin_org_slug": "dynapack-asia",
    "no_jb_schema": null,
    "external_apply_url": null,
    "ats_duplicate": false,
    "linkedin_id": 4444598040
  }
]
```


G. Dokumentasi lainnya (Dokumentasi API, Dokumentasi dari uji coba, Dan full kode program)

Github: https://github.com/Vonn-27/Asisten-GenAI-untuk-Rekomendasi-Tujuan-Magang