IMPLEMENTASI *RETRIEVAL AUGMENTED GENERATION* (RAG) *MULTIMODAL* SEBAGAI SISTEM REKOMENDASI BERBASIS TEKS DAN GAMBAR DI PLATFORM *E-COMMERCE*



Disusun Oleh:

N a m a : Bagas Wahyu Herdiansyah
NIM : 21523013

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
2025

HALAMAN PENGESAHAN DOSEN PEMBIMBING

IMPLEMENTASI *RETRIEVAL AUGMENTED GENERATION* (RAG) *MULTIMODAL* SEBAGAI SISTEM REKOMENDASI BERBASIS TEKS DAN GAMBAR DI PLATFORM *E-COMMERCE*

TUGAS AKHIR

Disusun Oleh:

N a m a : Bagas Wahyu Herdiansyah

NIM : 21523013

Yogyakarta, 23 Juli 2025

Pembimbing,

(Mukhammad Andri Setiawan, S.T., M.Sc., Ph.D.)

HALAMAN PENGESAHAN DOSEN PENGUJI

IMPLEMENTASI *RETRIEVAL AUGMENTED GENERATION* (RAG) *MULTIMODAL* SEBAGAI SISTEM REKOMENDASI BERBASIS TEKS DAN GAMBAR DI PLATFORM *E-COMMERCE*

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, 8 September 2025

Tim Penguji

Mukhammad Andri Setiawan, S.T., M.Sc., Ph.D.

Anggota 1
Ir. Chandra Kusuma Dewa, S.Kom., M.Cs., Ph.D.

Anggota 2

Sheila Nurul Huda, S.Kom., M.Cs.

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

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

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Bagas Wahyu Herdiansyah

NIM : 21523013

Tugas akhir dengan judul:

IMPLEMENTASI *RETRIEVAL AUGMENTED GENERATION* (RAG) *MULTIMODAL* SEBAGAI SISTEM REKOMENDASI BERBASIS TEKS DAN GAMBAR DI PLATFORM *E-COMMERCE*

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, 23 Juli 2025



(Bagas Wahyu Herdiansyah)

Dipindai dengan CamScanner

HALAMAN PERSEMBAHAN

Segala puji dan syukur dipanjatkan ke hadirat Allah SWT atas rahmat, karunia, dan inayah-Nya yang memungkinkan terselenggaranya penelitian dan penyusunan tugas akhir ini hingga tahap akhir. Hanya dengan pertolongan dan kemudahan dari-Nya, seluruh proses yang panjang dan menantang dapat dilalui dengan baik. Shalawat serta salam senantiasa tercurah kepada Nabi Muhammad SAW, teladan utama umat manusia, dengan harapan memperoleh syafaat beliau di hari kemudian.

Tugas akhir ini dipersembahkan kepada Ayah dan Ibu tercinta, yang telah menjadi sumber inspirasi dan kekuatan dalam menempuh pendidikan. Keteladanan, keikhlasan, dan kerja keras yang diperlihatkan oleh Ayah, serta kasih sayang, doa yang tiada henti, dan dukungan tulus dari Ibu, menjadi fondasi utama dalam perjalanan akademik ini. Segala pengorbanan dan upaya yang telah diberikan menjadi cahaya penerang setiap langkah.

Ucapan terima kasih dan penghargaan juga ditujukan kepada seluruh anggota keluarga. Kehadiran adik-adik yang selalu menghadirkan keceriaan dan kehangatan, turut memberikan semangat dan motivasi tambahan dalam menghadapi setiap tantangan akademik. Keluarga senantiasa menjadi tempat kembali yang penuh dengan ketenangan, rasa aman, dan kasih sayang.

Apresiasi setinggi-tingginya diberikan kepada seluruh dosen Program Studi Informatika, Fakultas Teknologi Industri Universitas Islam Indonesia, atas ilmu, bimbingan, dan arahan yang diberikan selama masa studi. Penghargaan khusus ditujukan kepada Bapak Mukhammad Andri Setiawan, S.T., M.Sc., Ph.D., selaku dosen pembimbing, atas dedikasi, perhatian, dan bimbingan yang sangat berarti dalam proses penyusunan tugas akhir ini.

Terima kasih yang tulus juga disampaikan kepada rekan-rekan dan sahabat seperjuangan atas segala dukungan, doa, dan kebersamaan yang telah diberikan. Semangat, bantuan, serta kerja sama yang terjalin selama ini menjadi kekuatan dalam menghadapi berbagai tantangan, baik dalam lingkup akademik maupun kehidupan sehari-hari.

Penghargaan terakhir diberikan kepada diri sendiri atas komitmen, konsistensi, dan ketekunan dalam menempuh proses penyusunan tugas akhir ini. Setiap usaha, pengorbanan, dan perjalanan yang telah dilewati membuktikan bahwa keberhasilan merupakan buah dari kerja keras dan dedikasi yang sungguh-sungguh.

HALAMAN MOTO

"The best way to predict the future is to invent it."

— Alan Kay

KATA PENGANTAR

Segala puji dan syukur dipanjatkan ke hadirat Allah SWT, Tuhan semesta alam, atas limpahan rahmat, karunia, dan petunjuk-Nya yang telah memudahkan seluruh proses penelitian dan penyusunan skripsi berjudul **"**Implementasi *Retrieval Augmented Generation* (RAG) *Multimodal* sebagai Sistem Rekomendasi Berbasis Teks dan Gambar di Platform *E-commerce***"**. Dengan izin dan kasih sayang-Nya, seluruh tahapan penyusunan karya ilmiah ini dapat dilalui dengan baik. Shalawat serta salam semoga senantiasa tercurah kepada Nabi Muhammad SAW, suri teladan utama bagi seluruh umat manusia.

Skripsi ini disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana pada Program Studi Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia. Proses penyelesaian karya ilmiah ini tidak terlepas dari dukungan, doa, serta bantuan dari berbagai pihak yang telah memberikan kontribusi dan motivasi sepanjang perjalanan akademik. Oleh karena itu, apresiasi dan ucapan terima kasih yang sebesar-besarnya disampaikan kepada:

1. Bapak Fathul Wahid, S.T., M.Sc., Ph.D., selaku Rektor Universitas Islam Indonesia, beserta jajaran pimpinan universitas atas kesempatan dan fasilitas yang telah diberikan dalam menciptakan lingkungan akademik yang kondusif.
2. Prof. Dr. Ir. Hari Purnomo, M.T., IPU, ASEAN.Eng., selaku Dekan Fakultas Teknologi Industri, beserta seluruh jajaran, atas dukungan dan fasilitas yang diberikan selama masa studi.
3. Dr. Ir. Raden Teduh Dirgahayu, S.T., M.Sc., selaku Ketua Jurusan Informatika, atas arahan dan bimbingan yang sangat berharga dalam pengembangan keilmuan dan proses akademik.
4. Ir. Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D., selaku Ketua Program Studi Informatika, atas motivasi dan semangat yang senantiasa diberikan selama menjalani proses pendidikan.
5. Bapak Mukhammad Andri Setiawan, S.T., M.Sc., Ph.D., selaku dosen pembimbing, atas bimbingan, arahan, kesabaran, serta perhatian yang telah diberikan sepanjang proses penyusunan skripsi ini. Setiap masukan, waktu, dan dukungan yang diberikan sangat berarti dalam menyelesaikan karya ilmiah ini. Semoga Allah SWT membalas segala kebaikan dan ketulusan dengan keberkahan dan rahmat yang berlimpah.

6 Bapak Andhik Budi Cahyono, S.T., M.T., selaku dosen pembimbing akademik, atas bimbingan, perhatian, serta nasihat yang telah membantu dalam perjalanan studi selama di Program Studi Informatika.

7 Seluruh dosen Program Studi Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia, atas ilmu pengetahuan, dedikasi, dan perhatian yang telah menjadi bekal berharga selama masa perkuliahan.

8 Kedua orang tua tercinta, yang menjadi sumber kekuatan, doa, dan pengorbanan yang tiada henti. Ucapan terima kasih setinggi-tingginya atas segala perjuangan, kasih sayang, dan dukungan yang diberikan sepanjang perjalanan pendidikan. Semoga Allah SWT membalas dengan pahala terbaik dan menempatkan dalam kemuliaan di sisi-Nya.

9 Kakak-Kakak tercinta, yang selalu memberikan semangat dan keceriaan dalam menjalani berbagai proses kehidupan dan akademik.

10 Rekan-rekan mahasiswa Informatika serta sahabat seperjuangan, atas kebersamaan, semangat, dan pengalaman berharga yang dibagikan selama masa studi.

11 Tiara Nabilla Sandriana, terima kasih atas ketulusan dukungan, perhatian, dan semangat yang senantiasa menyertai setiap langkah dalam proses penyusunan skripsi ini. Kehadiran serta kebersamaan yang terjalin telah menjadi penopang moral dan motivasi yang berharga dalam menghadapi berbagai rintangan, sekaligus menambah warna dan kebahagiaan dalam perjalanan akademik ini.

Yogyakarta, 23 Juli 2025



(Bagas Wahyu Herdiansyah)

SARI

Penelitian ini menjawab tantangan pada platform e-commerce, di mana sistem pencarian tradisional seringkali gagal memahami niat pengguna yang kompleks dan mengabaikan aspek visual produk, sehingga menciptakan friksi. Untuk mengatasinya, dirancang dan dievaluasi sebuah sistem rekomendasi end-to-end berbasis arsitektur Retrieval-Augmented Generation (RAG) Multimodal yang bertujuan menciptakan pengalaman belanja konsultatif. Sistem ini diimplementasikan sebagai aplikasi web dengan empat mode pencarian, menggunakan model SigLIP untuk embedding, Gemini 2.5 Pro untuk generasi respons, dan database vektor PostgreSQL+pgvector.

Evaluasi mixed-methods digunakan untuk mengukur performa sistem. Secara kuantitatif, fitur RAG Multimodal terbukti unggul pada seluruh metrik retrieval (misalnya, nDCG@10 mencapai 0.86) terhadap 60 kueri ground truth. Kualitas jawaban RAG juga sangat tinggi, dengan skor Faithfulness 0.90 dan Answer Relevance 0.80, yang mengindikasikan respons yang faktual dan relevan. Secara kualitatif, melalui metode Think-Aloud Protocol dan wawancara, partisipan secara universal memilih asisten RAG sebagai fitur yang paling bermanfaat.

Temuan kunci menunjukkan bahwa RAG berhasil mengubah paradigma interaksi dari "mencari" menjadi "berkonsultasi," terutama pada skenario pencarian yang ambigu dan eksploratif. Kemampuan sistem untuk memberikan rekomendasi yang terjustifikasi secara signifikan mengurangi beban kognitif dan membangun kepercayaan pengguna. Penelitian ini mengonfirmasi bahwa RAG Multimodal tidak hanya merupakan solusi teknis yang lebih unggul, tetapi juga mampu menghadirkan pengalaman pengguna yang superior dan terpersonalisasi dalam ekosistem e-commerce modern.

Kata kunci: Retrieval-Augmented Generation, RAG Multimodal, Sistem Rekomendasi, SigLIP, Gemini 2.5 Pro, Evaluasi Mixed-Methods, E-commerce.

GLOSARIUM

Augmentation	Proses penggabungan informasi hasil *retrieval* dengan kueri pengguna untuk membentuk prompt yang lebih kaya sebelum dikirim ke model generatif, agar respons lebih relevan dan kontekstual.
Cosine Similarity	Rumus matematika yang digunakan untuk mengukur tingkat kemiripan antara dua vektor dalam ruang multidimensi berdasarkan sudut di antara vektor tersebut. Nilai yang mendekati 1 menunjukkan kemiripan semantik yang tinggi.
Embedding	Representasi numerik dari data teks atau gambar dalam bentuk vektor berdimensi tinggi, yang dihasilkan oleh model seperti SigLIP, sehingga data dapat dibandingkan dan diproses secara matematis oleh sistem rekomendasi.
Gemini 2.5 Pro	*Large Language Model* (LLM) generatif *multimodal* dari Google yang mampu memahami dan menghasilkan respons berdasarkan input teks dan gambar secara simultan.
HNSW (Hierarchical Navigable Small World)	Algoritma pembuatan indeks dalam vector database yang mempercepat pencarian kemiripan vektor secara efisien pada data berskala besar.
Incremental Model	Metodologi pengembangan perangkat lunak yang membagi proses pembangunan aplikasi menjadi tahapan bertahap (*increment*), sehingga setiap fitur dapat diimplementasikan, diuji, dan divalidasi secara terpisah sebelum integrasi penuh.
Multimodal	Pendekatan yang menggabungkan lebih dari satu jenis data (misal: teks dan gambar) dalam pemrosesan dan pencarian, agar hasil rekomendasi lebih akurat dan relevan dengan kebutuhan pengguna.
pgvector	Ekstensi pada PostgreSQL yang memungkinkan penyimpanan dan pencarian data vektor (embedding) secara efisien di database relasional.

RAG (*Retrieval Augmented Generation*)	Arsitektur AI yang menggabungkan proses pencarian informasi eksternal (*retrieval*) dengan kemampuan generatif model bahasa besar (LLM) untuk menghasilkan respons berbasis fakta terkini.
Recommender System (Sistem Rekomendasi)	Sistem yang memberikan rekomendasi produk atau konten kepada pengguna berdasarkan preferensi, perilaku, dan data historis mereka.
Retrieval	Tahap dalam arsitektur RAG yang berfungsi mencari dan mengambil data atau informasi yang paling relevan dari basis data eksternal untuk dijadikan konteks bagi model generatif.
SigLIP	Model embedding multimodal yang mampu menghasilkan representasi vektor untuk teks dan gambar dalam ruang vektor bersama, sehingga pencarian kemiripan dapat dilakukan lintas modalitas.
Streamlit	Framework open-source untuk membangun antarmuka aplikasi web berbasis Python secara cepat, yang digunakan sebagai front-end pada penelitian ini.
Think-Aloud Protocol	Metode evaluasi kualitatif di mana partisipan diminta mengungkapkan pikiran dan pertimbangannya secara verbal saat menggunakan sistem, untuk memahami proses kognitif dan persepsi pengguna.
Vector Database	Basis data yang dirancang untuk menyimpan dan melakukan pencarian secara efisien pada data vektor berdimensi tinggi (embedding), mendukung proses similarity search dalam sistem rekomendasi.

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 xii
DAFTAR TABEL xiv
DAFTAR GAMBAR xv
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 7
1.3 Batasan Masalah 7
1.4 Tujuan Penelitian 8
1.5 Manfaat Penelitian 8
1.6 Sistematika Penulisan 8
BAB II LANDASAN TEORI 2
2.1 Dasar Teori 2
2.1.1 Sistem Rekomendasi di *E-commerce* 3
2.1.2 Sistem Rekomendasi di *E-commerce* 5
2.1.3 *Vector Embeddings* 6
2.1.4 Model *Multimodal (Vision-Language Models)* 8
2.1.5 *Retrieval-Augmented Generation (RAG)* 10
2.1.6 *Vector Database* 13
2.2 Tinjauan Pustaka 15
BAB III METODOLOGI PENELITIAN 18
3.1 Fase 1: *Requirement Analysis* 19
3.1.1 Analisis Kebutuhan Sistem 19
3.2 Fase 2: *Design & Development* 21
3.2.1 Desain Arsitektur Sistem 21
3.2.2 *Use Case Diagram* 23
3.2.3 Desain Skema *Database* 25
3.3 Desain dan Prosedur Evaluasi Sistem 26
3.3.1 Persiapan Perangkat Evaluasi 26
3.3.2 Prosedur Evaluasi Kuantitatif 27
3.3.3 Prosedur Evaluasi Kualitatif 31
BAB IV HASIL DAN PEMBAHASAN 34
4.1 Arsitektur dan Alur Kerja Sistem 34
4.2 Fase 3: Implementation 35
4.2.1 Increment 1: Implementasi Fondasi Data 36
4.2.2 Increment 2: Implementasi Fitur Pencarian Baseline 46
4.2.3 Increment 3: Implementasi Fitur RAG Multimodal 54
4.2.4 Increment 4: Implementasi Antarmuka Pengguna 59
4.3 Fase 4: Testing dan Hasil Penelitian 64

4.3.1 Hasil Evaluasi Kuantitatif 65
4.3.2 Hasil Evaluasi Kualitatif 69
BAB V KESIMPULAN DAN SARAN 73
5.1 Kesimpulan 73
5.2 Saran 74
DAFTAR PUSTAKA 76
LAMPIRAN 80

DAFTAR TABEL

Tabel 2. 1 Tinjauan Pustaka 15
Tabel 3. 1 Penjelasan *Use Case Diagram* 24
Tabel 4. 1 Data Mentah 36
Tabel 4. 2 Hasil Pengujian *Precision@10* 65
Tabel 4. 3 Hasil Pengujian *Recall@10* 66
Tabel 4. 4 Hasil Pengujian *MAP@10* 67
Tabel 4. 5 Hasil Pengujian *nDCG@10* 68
Tabel 4. 6 Hasil Pengujian *Faithfulnes* dan *Answer Relevance* 68
Tabel 4. 7 Hasil Pengujian Kualitatif 69

DAFTAR GAMBAR

Gambar 2.1 Arsitektur RAG 11
Gambar 3.1 Alur metode *Incremental Model* 18
Gambar 3. 2 *Flow* Arsitektur Sistem. 21
Gambar 3. 3 *Use case diagram* 24
Gambar 3.4 *Entity Relationship Diagram* 25
Gambar 4.1 Realisasi Arsitektur Sistem 35
Gambar 4. 2 Kode *Preprocessing* Data 40
Gambar 4. 3 Kode *Embedding* Teks 43
Gambar 4. 4 Kode *Embedding* Gambar 44
Gambar 4. 5 Skema *Database* 46
Gambar 4. 6 Kode Pencarian Kata Kunci 48
Gambar 4. 7 Kode Pencarian Semantik Teks 49
Gambar 4. 8 Kode Pencarian Semantik Gambar 51
Gambar 4. 9 Kode Pencarian *Multimodal* Teks dan Gambar 54
Gambar 4.10 Kode Logika *Retrieval* 55
Gambar 4.11 Kode Logika *Augmentation* 57
Gambar 4.12 Kode Logika *Generation* 59
Gambar 4.13 Antarmuka Pencarian Kata Kunci 60
Gambar 4.14 Antarmuka Pencarian Semantik Teks 61
Gambar 4.15 Antarmuka Pencarian Semantik Gambar 62
Gambar 4. 16 Antarmuka Pencarian Multimodal 63
Gambar 4.17 Antarmuka Asisten Chat RAG 64

BAB I
PENDAHULUAN

1.1 Latar Belakang

Ekonomi digital Indonesia telah menunjukkan pertumbuhan yang luar biasa, didorong oleh peningkatan penetrasi internet yang masif. Jumlah pengguna internet di Indonesia melonjak signifikan, dari tingkat penetrasi 73,7% pada periode 2019-2020, menjadi 79,5% pada awal tahun 2024 (Erlina F. Santika, 2024) . Angka ini setara dengan 221,5 juta jiwa yang telah terhubung ke internet, menjadikan Indonesia salah satu pasar digital terbesar di Asia Tenggara. Pertumbuhan ini secara langsung mengakselerasi ekspansi sektor *e-commerce*, yang secara konsisten menjadi pilar utama dan pendorong pertumbuhan ekonomi digital nasional (Pratiwi, 2022).

Peningkatan basis pengguna ini berkorelasi langsung dengan lonjakan nilai transaksi *e-commerce*. Nilai *Gross Merchandise Value* (GMV) yang pada tahun 2020 tercatat sebesar $44 miliar terus menunjukkan pertumbuhan yang pesat (Google, 2025) .Berbagai studi memproyeksikan nilai pasar *e-commerce* Indonesia akan mencapai antara $82 miliar hingga $124 miliar pada tahun 2025, menegaskan posisinya sebagai salah satu pasar terbesar di kawasan ini (Google, 2025). Pertumbuhan eksponensial ini menandakan pergeseran fundamental dalam perilaku konsumen dan lanskap ritel di Indonesia, di mana e-commerce menjadi enabler penting bagi inklusi ekonomi, terutama bagi UMKM.

Seiring dengan matangnya pasar yang ditandai dengan melambatnya laju akuisisi pengguna baru, fokus platform *e-commerce* bergeser dari akuisisi ke retensi dan peningkatan nilai seumur hidup pelanggan. Dalam lingkungan yang semakin kompetitif, kualitas interaksi dan kemudahan dalam menemukan produk menjadi diferensiator utama untuk mempertahankan loyalitas. Pengalaman pencarian produk yang buruk, yang menimbulkan friksi dan frustrasi, secara langsung berkontribusi pada tingkat konversi yang lebih rendah dan perpindahan pelanggan ke platform pesaing (Bai et al., 2020). Oleh karena itu, optimalisasi teknologi pencarian bukan lagi sekadar peningkatan teknis, melainkan sebuah keharusan strategis untuk pertumbuhan bisnis yang berkelanjutan.

Secara historis, teknologi pencarian produk di *e-commerce* telah berevolusi melalui beberapa paradigma utama. Tahap awal didominasi oleh pencarian berbasis kata kunci (keyword-based search), sebuah metode yang bekerja dengan mencocokkan secara harfiah istilah yang dimasukkan pengguna dengan data produk yang ada di dalam indeks (Rokon et al.,

2024). Meskipun cepat dan efisien untuk kueri yang sangat spesifik seperti nomor seri atau nama produk yang pasti, pendekatan ini pada dasarnya bersifat kaku dan rapuh (Rokon et al., 2024). Keterbatasan ini mendorong pengembangan teknologi yang lebih canggih untuk memahami maksud pengguna dengan lebih baik.

Paradigma berikutnya adalah pencarian semantik (*semantic search*), yang menandai lompatan signifikan dalam kemampuan sistem untuk memahami niat pengguna. Berbeda dengan pencocokan literal, pencarian semantik memanfaatkan *Natural Language Processing* (NLP) dan *vector embeddings* untuk memahami makna dan konteks di balik sebuah kueri (Rokon et al., 2024). Teknologi ini mampu menginterpretasikan hubungan antar konsep, mengenali sinonim, dan memahami maksud pengguna bahkan ketika kata-kata yang digunakan tidak sama persis dengan yang ada di deskripsi produk. Dengan demikian, pencarian semantik mulai menjembatani kesenjangan antara bahasa alami pengguna dan data terstruktur di dalam basis data *e-commerce*.

Meskipun merupakan kemajuan, kedua paradigma pencarian sebelumnya memiliki keterbatasan fundamental dalam konteks *e-commerce* modern. Pencarian berbasis kata kunci sering kali gagal karena dua masalah utama: masalah sinonim, di mana sistem tidak dapat menghubungkan istilah yang berbeda tetapi bermakna sama seperti "pakaian" dan "busana"; dan masalah ambiguitas atau polisemi, di mana satu kata dapat memiliki banyak arti seperti "Apple" (merek vs. buah) (Rokon et al., 2024). Kegagalan ini menghasilkan interaksi yang membuat frustrasi, dengan hasil yang tidak relevan (false positives) atau tidak ditemukannya produk yang dicari (*false negatives*) (Rokon et al., 2024).

Pencarian semantik berbasis teks, meskipun mampu mengatasi masalah sinonim dan ambiguitas, memiliki kelemahan kritis yang disebut "kebutaan visual" (visual blindspot) (W. Liang et al., 2022). Dalam domain e-commerce, penampilan visual suatu produk seringkali menjadi faktor penentu utama dalam keputusan pembelian, bahkan lebih penting daripada deskripsi tekstualnya (W. Liang et al., 2022). Sistem pencarian yang hanya mengandalkan teks tidak mampu memproses atau memahami informasi kaya yang terkandung dalam gambar, seperti gaya, warna, pola, atau estetika keseluruhan, yang seringkali sulit diartikulasikan oleh pengguna dengan kata-kata.

Keterbatasan ini secara kolektif menciptakan apa yang dikenal sebagai "friksi pencarian" (*search friction*), yaitu hambatan yang mempersulit pengguna untuk menemukan produk yang mereka inginkan. Menurut Teori Beban Kognitif (Cognitive Load Theory), peningkatan friksi ini secara langsung meningkatkan beban kognitif ekstrinsik upaya mental yang tidak perlu yang

disebabkan oleh cara informasi disajikan (Li et al., 2025). Ketika beban kognitif ekstrinsik terlalu tinggi, kapasitas mental pengguna untuk mengevaluasi produk (tugas utama) menjadi berkurang, yang pada akhirnya menyebabkan frustrasi, kelelahan dalam mengambil keputusan, dan potensi pengabaian sesi belanja (Li et al., 2025).

Perkembangan terkini dalam Kecerdasan Buatan (AI) telah melahirkan paradigma baru yang dikenal sebagai perdagangan percakapan (*conversational commerce*). Pendekatan ini mengubah interaksi belanja daring dari model kueri-respons yang statis menjadi dialog interaktif dua arah yang dinamis. Tujuannya adalah untuk mereplikasi pengalaman konsultatif saat berbelanja di toko fisik, di mana pelanggan dapat berdiskusi, mengajukan pertanyaan lanjutan, dan menerima rekomendasi yang dipersonalisasi dari seorang asisten penjualan (Jannach, 2021).

Teknologi kunci yang memungkinkan pergeseran ini adalah Large Language Models (LLM). Dengan kemampuan pemahaman dan generasi bahasa alami yang canggih, LLM dapat memfasilitasi interaksi yang menyerupai manusia dalam skala besar, memberikan respons yang relevan secara kontekstual dan dipersonalisasi. Penggunaan LLM membuka peluang untuk menciptakan interaksi pelanggan yang lebih menarik dan memuaskan, yang pada akhirnya dapat meningkatkan loyalitas dan tingkat konversi secara signifikan.

Meskipun LLM memiliki kemampuan percakapan yang kuat, penerapannya secara langsung dalam domain kritis seperti *e-commerce* menghadapi dua tantangan utama: halusinasi dan pengetahuan yang usang. LLM diketahui dapat menghasilkan informasi yang terdengar meyakinkan namun secara faktual salah (halusinasi), dan pengetahuannya terbatas pada data terakhir yang digunakan untuk pelatihannya (Gupta et al., 2024). Isu-isu ini membuat LLM yang berdiri sendiri tidak dapat diandalkan untuk memberikan informasi produk yang akurat dan terkini.

Untuk mengatasi masalah ini, arsitektur *Retrieval-Augmented Generation* (RAG) diperkenalkan sebagai kerangka kerja AI yang inovatif (Gupta et al., 2024). RAG meningkatkan keandalan LLM dengan menghubungkannya ke basis pengetahuan eksternal yang dapat diverifikasi. Proses RAG terdiri dari dua fase utama: pertama, fase Retrieval, di mana sistem mengambil informasi atau fakta yang relevan dari sumber data eksternal (misalnya, katalog produk); kedua, fase Generation, di mana LLM menggunakan fakta yang telah diambil tersebut sebagai konteks untuk menghasilkan respons yang akurat dan relevan (Gupta et al., 2024). Dengan demikian, RAG memastikan bahwa jawaban yang diberikan

didasarkan pada data yang faktual dan mutakhir, sehingga meningkatkan kepercayaan pengguna (Gupta et al., 2024).

Penggabungan dari kemajuan teknologi yang telah dibahas perdagangan percakapan, LLM, dan RAG melahirkan solusi generasi berikutnya: RAG *Multimodal*. Sistem ini tidak hanya mewarisi kemampuan percakapan dari LLM dan landasan faktual dari RAG, tetapi juga secara krusial menambahkan kemampuan untuk memahami dan memproses informasi visual, mengatasi "kebutaan visual" dari sistem sebelumnya. Dengan mengintegrasikan data dari berbagai modalitas seperti teks dan gambar, sistem RAG *Multimodal* dapat menjawab kueri yang kompleks dan bernuansa yang tidak mungkin ditangani oleh sistem *unimodal* (Mortaheb et al., 2025).

Untuk membangun sistem RAG Multimodal yang fungsional, penelitian ini akan memanfaatkan kombinasi model-model canggih. Pada lapisan retrieval, sistem akan menggunakan model embedding google/siglip-so400m-patch14-384. Model ini dipilih karena kemampuannya yang unggul dalam menciptakan ruang embedding bersama untuk input teks dan gambar, yang difasilitasi oleh inovasi utamanya pada penggunaan fungsi *Sigmoid Loss* (Zhai et al., 2023). Fungsi loss ini terbukti lebih efisien dan stabil dibandingkan dengan *softmax-based contrastive loss* yang digunakan oleh model-model sebelumnya, memungkinkan pencarian kesamaan semantik yang cepat dan akurat (Zhai et al., 2023).

Pada lapisan generasi, sistem akan menggunakan Google Gemini 2.5 Pro. Model ini dipilih karena tiga keunggulan utamanya: pertama, ia dirancang sebagai model yang natively multimodal, mampu memahami dan menalar berdasarkan konteks yang mengandung teks dan gambar secara bersamaan. Kedua, ia menggunakan arsitektur *Mixture-of-Experts* (MoE) yang efisien, memungkinkan kinerja model raksasa dengan biaya komputasi yang jauh lebih rendah. Ketiga, Gemini 2.5 Pro memiliki jendela konteks yang sangat besar hingga 1 juta token, memungkinkannya memproses sejumlah besar informasi yang diambil dari katalog produk dalam satu prompt tunggal, yang krusial untuk menghasilkan respons yang komprehensif dan bernuansa (Gemini Team, 2025).

Untuk membangun fondasi evaluasi yang objektif, penelitian ini mengadopsi serangkaian metrik kuantitatif yang merupakan standar industri untuk mengukur kinerja sistem rekomendasi (Jadon & Patil, 2023). Metodologi penelitian secara spesifik menetapkan penggunaan metrik *ranking* seperti *Precision@10*, *Recall@10*, *MAP@10*, dan *nDCG@10* untuk menilai kualitas *retrieval* (Jadon & Patil, 2023). Metrik-metrik ini akan digunakan untuk

secara objektif membandingkan kinerja dari keempat mode pencarian, sehingga menetapkan *baseline* kinerja teknis.

Selain itu, untuk mengevaluasi komponen generatif dari sistem RAG, metrik *Faithfulness* dan *Answer Relevance* akan digunakan. *Faithfulness* mengukur apakah jawaban yang dihasilkan AI didasarkan sepenuhnya pada konteks yang disediakan, sehingga mencegah halusinasi (Es et al., 2023). *Answer Relevance* menilai apakah respons tersebut benar-benar menjawab maksud dari pertanyaan pengguna (Es et al., 2023). Kerangka evaluasi kuantitatif ini secara cerdas memisahkan masalah: metrik *ranking* mengisolasi kinerja komponen *Retriever*, sementara *Faithfulness* dan *Answer Relevance* mengisolasi kinerja komponen *Generator*. Pendekatan terpisah ini memungkinkan diagnosis yang lebih tepat tentang di mana letak kekuatan atau kelemahan sistem. Secara esensial, lensa kuantitatif menjawab pertanyaan "seberapa akurat dan relevan hasilnya?" dengan menyediakan data numerik yang terukur. Namun, keterbatasan utamanya adalah ia tidak dapat menjelaskan "mengapa" di balik angka-angka tersebut, sebuah celah yang diisi oleh lensa kualitatif.

Batu penjuru dari seluruh argumen metodologis penelitian ini terletak pada pengakuan bahwa metrik teknis saja tidak cukup. Sebagaimana dinyatakan dalam kesimpulan dari survei komprehensif tentang evaluasi sistem rekomendasi: *"Importantly, real-world business value and user satisfaction emerge not solely from these ML metrics but significantly from user feedback."*(Jadon & Patil, 2023). Kutipan ini secara eksplisit menyatakan bahwa nilai bisnis nyata dan kepuasan pengguna tidak hanya berasal dari metrik *machine learning*, tetapi secara signifikan dari umpan balik pengguna.

Untuk secara langsung menangkap "umpan balik pengguna" ini, penelitian ini memposisikan evaluasi kualitatif sebagai komponen sentral. Metodologi yang diusulkan adalah penggunaan *Think-Aloud Protocol* dan wawancara semi-terstruktur. *Think-Aloud Protocol* adalah metode di mana peserta diminta untuk menyuarakan pikiran, perasaan, dan alasan mereka secara verbal saat mereka berinteraksi dengan produk (Fan & Zhu, 2022). Metode ini dirancang untuk menangkap wawasan yang tidak dapat diukur oleh metrik kuantitatif, seperti proses kognitif pengguna, ekspektasi mereka (model mental), alasan di balik tindakan mereka, serta momen frustrasi dan kepuasan yang secara langsung terkait dengan pengalaman mereka secara keseluruhan.

Kerangka evaluasi *mixed-methods* ini secara langsung dirancang untuk menjembatani kesenjangan yang diidentifikasi dalam literatur. Lensa kuantitatif akan menunjukkan *bahwa* RAG Multimodal memiliki kinerja teknis yang unggul. Lensa kualitatif kemudian akan menjelaskan *mengapa* keunggulan teknis itu penting dari perspektif manusia. Dengan demikian, data kualitatif memberikan narasi dan makna "dunia nyata" pada angka-angka kuantitatif, menunjukkan bagaimana "kinerja teknis" diterjemahkan menjadi "kepuasan pengguna" dan "nilai dunia nyata". Pendekatan ini secara langsung menjawab pertanyaan "mengapa pengguna merasa hasilnya relevan?" dan "bagaimana pengalaman interaksinya?" dengan mengumpulkan data naratif melalui metode seperti *Think-Aloud Protocol*. Meskipun hasilnya bersifat subjektif dan tidak dapat digeneralisasi secara statistik, ia memberikan pemahaman mendalam tentang persepsi dan konteks pengguna yang tidak dapat ditangkap oleh metrik kuantitatif.

Alur logis yang telah dibangun menegaskan bahwa pencarian *e-commerce* saat ini memiliki keterbatasan fundamental yang menciptakan friksi dan beban kognitif bagi pengguna. Paradigma perdagangan percakapan yang didukung oleh LLM menawarkan model interaksi yang superior, namun LLM mandiri tidak dapat diandalkan karena masalah halusinasi dan pengetahuan yang usang. RAG Multimodal muncul sebagai solusi arsitektural yang canggih, menggabungkan pemahaman multimodal, penalaran generatif, dan landasan faktual, yang diimplementasikan menggunakan tumpukan teknologi yang secara teknis dan konseptual sangat cocok untuk tugas ini.

Kontribusi utama dari penelitian ini tidak hanya terletak pada implementasi sistem yang canggih, tetapi pada validasi komprehensifnya melalui kerangka evaluasi dua lensa. Dengan mensintesis bukti kuantitatif tentang keunggulan teknis dengan bukti kualitatif tentang pengalaman pengguna yang superior, penelitian ini secara langsung menjawab tantangan yang diidentifikasi dalam literatur : menjembatani kesenjangan antara metrik *machine learning* dan nilai bisnis serta kepuasan pengguna di dunia nyata. Ini memposisikan penelitian ini sebagai studi kasus yang relevan dan teliti dalam pengembangan sistem AI yang berpusat pada manusia.

1.2 Rumusan Masalah

a. Bagaimana merancang dan mengimplementasikan sistem RAG Multimodal end-to-end untuk e-commerce yang mendukung empat mode: keyword search, semantic text, semantic image, serta chat asisten RAG?
b. Sejauh mana efektivitas retrieval keempat mode *(keyword, semantic text, multimodal search, RAG multimodal*) jika diukur dengan *Precision@10*, *Recall@10*, *MAP@10*, dan *nDCG@10*, serta bagaimana kualitas jawaban fitur RAG dinilai melalui *Faithfulness* dan *Answer Relevance*?
c. Bagaimana pengalaman dan preferensi pengguna terhadap setiap fitur berdasarkan evaluasi kualitatif berbasis skenario dan *Think-Aloud Protocol,* termasuk skenario di mana RAG multimodal memberikan nilai tambah paling signifikan (kompleks, ambigu, dan eksploratif)?

1.3 Batasan Masalah

Batasan penelitian digunakan untuk mengerucutkan dan memfokuskan penelitian sehingga tidak terjadi pelebaran konteks penelitian. Berikut adalah detail dari batasan penelitian ini:

a. produk alas kaki hasil *web scraping* dari Amazon UK berjumlah 11.604 item dengan hanya metadata teks terpilih dan gambar utama yang dipakai untuk embedding dan retrieval.
b. SigLIP (dimensi embedding 1152) digunakan untuk teks dan gambar, PostgreSQL+pgvector dengan indeks HNSW untuk ANN, Gemini 2.5 Pro untuk generation, implementasi UI berbasis Streamlit, *late fusion* untuk *multimodal query*.
c. Evaluasi bersifat kualitatif, berfokus pada pemahaman mendalam tentang *perceived relevance* (relevansi yang dirasakan), kemudahan penggunaan, dan pengalaman pengguna secara keseluruhan.
d. teks dikombinasikan dan di-*chunk* dengan batas ±60 token dan overlap 10, input embedding dibatasi untuk menjaga konsistensi dimensi dan stabilitas inferensi.
e. Sistem dinilai dengan kualitas *retrieval* (*Precision@10, Recall@10, MAP@10, nDCG@10*) dan kualitas jawaban RAG (*Faithfulness*, *Answer Relevance*) dengan tidak mengevaluasi metrik sistem seperti latensi, throughput, atau biaya komputasi. *Think-Aloud Protocol* dan wawancara semi-terstruktur dengan *purposive sampling* dengan tugas skenario.

1.4 Tujuan Penelitian

a. Menghasilkan prototipe sistem RAG Multimodal e-commerce yang fungsional dengan empat mode pencarian baseline, arsitektur *Retrieval Augmented Generation*, dan antarmuka percakapan.
b. Menyusun baseline objektif kualitas retrieval untuk keempat mode (keyword, semantic text, multimodal, RAG) menggunakan *Precision@10, Recall@10, MAP@10, dan nDCG@10*, serta menilai kualitas jawaban RAG dengan *Faithfulness* dan *Answer Relevance*.
c. Mengevaluasi pengalaman pengguna melalui tugas skenario dan *Think-Aloud* untuk memetakan persepsi relevansi, kemudahan, kepercayaan, dan preferensi fitur, serta mengidentifikasi skenario bernilai tambah terbesar bagi RAG.

1.5 Manfaat Penelitian

a. Memperkaya literatur ilmiah mengenai evaluasi perbandingan model embeddings dalam domain pencarian semantik e-commerce, dengan penekanan pada evaluasi kualitatif dan persepsi pengguna.
b. Menyumbangkan pemahaman mengenai faktor-faktor yang memengaruhi kualitas pencarian semantik dari perspektif pengguna dengan keahlian domain, termasuk determinan relevansi yang dirasakan dan kepercayaan terhadap sistem.
c. Menghadirkan artefak rekayasa *end-to-end* RAG multimodal yang replikabel beserta arsitektur *Retrieval Augmentated Generation* sebagai acuan teknis untuk studi dan industri.
d. Memberikan kontribusi pada literatur di bidang Kecerdasan Buatan (AI) melalui studi kasus empiris tentang pengalaman pengguna pada sistem RAG multimodal di e-commerce umum, melampaui evaluasi sistem-sentris.
e. Menawarkan kerangka evaluasi campuran yang menggabungkan metrik retrieval berbasis *ground truth* (*Precision@10, Recall@10, MAP@10, nDCG@10, Faithfulness, Answer Relevance*) dan metode *Think-Aloud/*wawancara untuk menilai kualitas grounding dan percakapan.

1.6 Sistematika Penulisan

Untuk memberikan gambaran yang jelas dan terstruktur mengenai pembahasan dalam penelitian ini, berikut adalah sistematika penulisan yang digunakan:

a. BAB I: PENDAHULUAN

Bab ini menguraikan konteks penelitian, dimulai dari latar belakang masalah mengenai keterbatasan pencarian pada platform e-commerce modern. Bab ini juga merinci rumusan masalah, tujuan penelitian yang berfokus pada implementasi dan evaluasi sistem RAG Multimodal, batasan masalah yang ditetapkan, serta manfaat dan sistematika penulisan skripsi secara keseluruhan.

b. BAB II: LANDASAN TEORI

Bab ini membahas konsep-konsep teoretis yang menjadi fondasi penelitian. Teori yang dibahas meliputi arsitektur *Retrieval-Augmented Generation* (RAG), model *embedding* multimodal seperti SigLIP, *Large Language Models* (LLM) seperti Gemini, serta teknologi penyimpanan dan pencarian vektor menggunakan database PostgreSQL dengan ekstensi pgvector. Selain itu, bab ini juga mengulas penelitian terdahulu yang relevan dengan topik sistem rekomendasi berbasis AI dan evaluasi pengalaman pengguna.

c. BAB III: METODOLOGI PENELITIAN

Bab ini memaparkan desain *mixed-methods* yang mengikat langsung tujuan dan pertanyaan riset dengan keputusan metodologis, meliputi domain dan batasan penelitian, serta prosedur pengembangan berbasis Incremental Model yang terdiri atas requirement analysis, desain pengembangan, implementasi, dan pengujian evaluasi beserta artefak yang dihasilkan. Serta protokol evaluasi kuantitatif (*Precision@10, Recall@10, MAP@10, nDCG@10*) dan kualitatif (Think-Aloud, wawancara semi-terstruktur, analisis tematik) agar prosedur analisis dan penilaian kualitas jawaban dapat dinilai dan diulang.

d. BAB IV: HASIL DAN PEMBAHASAN

Bab ini menyajikan lingkungan uji dan realisasi tiap *increment* untuk menunjukkan keterlacakan dari rancangan ke implementasi. Pemaparan hasil kuantitatif *retrieval* dan kualitas jawaban dalam tabel/figur yang relevan diikuti pembahasan terpisah yang menafsirkan pola performa, menjelaskan keunggulan RAG multimodal pada kueri kompleks.

e. BAB V: KESIMPULAN DAN SARAN

Bab terakhir ini merangkum seluruh hasil penelitian dan pembahasan untuk menjawab rumusan masalah. Bab ini berisi kesimpulan mengenai keberhasilan implementasi sistem

RAG Multimodal dan dampaknya terhadap pengalaman pengguna, serta menyajikan saran konkret untuk penelitian selanjutnya.

BAB II
LANDASAN TEORI

Bab ini menyajikan kerangka teoritis yang menjadi landasan bagi keseluruhan arsitektur penelitian. Peran bab ini bukan sekadar kompilasi teori, melainkan sebagai fondasi argumentatif yang menopang alur pemikiran logis dan dapat dipertanggungjawabkan secara ilmiah. Pembahasan akan dimulai dari konsep-konsep fundamental dalam domain *e-commerce,* seperti sistem rekomendasi dan evolusi teknologi pencarian, yang secara bertahap membangun pemahaman menuju arsitektur teknologi yang lebih canggih.

Bab ini akan merincikan setiap teknologi yang diadopsi dalam penelitian ini yang merupakan respons logis terhadap keterbatasan dari paradigma sebelumnya. Alur ini dimulai dengan mengidentifikasi tantangan fundamental dalam sistem rekomendasi klasik, yang mendorong lahirnya kebutuhan akan pemahaman makna melalui pencarian semantik. Selanjutnya, bab ini akan mengupas secara mendalam komponen-komponen inti yang membentuk solusi dalam penelitian ini, yaitu *vector embeddings* sebagai bahasa universal data, model multimodal sebagai jembatan antara visual dan teks, arsitektur *Retrieval-Augmented Generation* (RAG) sebagai kerangka kerja penalaran, dan teknologi basis data vektor sebagai infrastruktur pendukungnya. Setiap komponen akan dianalisis tidak hanya dari aspek teknisnya, tetapi juga dari justifikasi pemilihannya dalam konteks sistem rekomendasi e-commerce yang konsultatif.

Bab ini diakhiri dengan tinjauan terhadap penelitian-penelitian terkait (*related work*) untuk memposisikan penelitian ini dalam lanskap riset terkini. Pemahaman mendalam terhadap fondasi teori ini akan menjadi dasar untuk menyoroti kebaruan (*novelty*) dan kontribusi orisinal yang ditawarkan oleh penelitian ini, sekaligus menjadi jembatan yang kokoh menuju pembahasan metodologi penelitian pada bab berikutnya.

2.1 Dasar Teori

Bagian ini menguraikan serangkaian konsep dan teknologi yang menjadi fondasi teknis dari sistem yang diimplementasikan. Uraian disusun secara sistematis, di mana keterbatasan dari suatu paradigma teknologi mendorong pengembangan paradigma berikutnya. Alur ini dimulai dari sistem rekomendasi klasik, berlanjut ke pencarian semantik, dan berpuncak pada arsitektur RAG Multimodal yang menjadi inti dari penelitian ini.

2.1.1 Sistem Rekomendasi di *E-commerce*

Lanskap e-commerce modern dapat digambarkan sebagai sebuah ekosistem digital yang sangat jenuh, dipenuhi oleh jutaan produk dan informasi yang tak terhingga. Bagi pengguna, situasi ini sering kali menciptakan sebuah fenomena yang dikenal sebagai ledakan informasi atau *information overload* (Nurdin & Abidin, 2023). Ketika dihadapkan pada pilihan yang terlalu banyak, pengguna cenderung merasa bingung, frustrasi, dan mengalami kelelahan kognitif, yang pada akhirnya menghambat proses pengambilan keputusan pembelian dan menurunkan tingkat kepuasan secara keseluruhan (Nurdin & Abidin, 2023). Fenomena ini menjadi tantangan signifikan bagi platform e-commerce yang bertujuan untuk memberikan pengalaman berbelanja yang efisien dan menyenangkan.

Untuk mengatasi tantangan ini, sistem rekomendasi (*recommender systems*) hadir sebagai solusi teknologi yang krusial. Sistem ini berfungsi sebagai alat penyaring informasi yang cerdas, yang bertujuan untuk memangkas kebisingan dan menyajikan item-item yang paling relevan bagi setiap pengguna secara individual (Jannach, 2021). Dengan menganalisis data perilaku pengguna, preferensi historis, dan atribut produk, sistem rekomendasi dapat memprediksi item yang kemungkinan besar akan diminati. Tujuannya tidak hanya untuk mempermudah proses penemuan produk, tetapi juga untuk meningkatkan pengalaman pengguna secara holistik, mendorong tingkat konversi penjualan, dan pada akhirnya membangun loyalitas pelanggan jangka Panjang (Nurdin & Abidin, 2023). Secara historis, terdapat dua paradigma utama yang mendominasi pengembangan sistem rekomendasi:

A. *Content-Based Filtering* (CBF)

Pendekatan *Content-Based Filtering* (CBF) beroperasi berdasarkan prinsip kemiripan atribut. Sistem ini merekomendasikan item kepada pengguna dengan menganalisis fitur-fitur dari item yang pernah disukai oleh pengguna tersebut di masa lalu (Uta et al., 2024). Logika dasarnya sederhana dan intuitif: "Jika Anda menyukai produk X yang memiliki fitur A, B, dan C, maka Anda kemungkinan besar juga akan menyukai produk Y yang juga memiliki fitur A, B, dan C". Misalnya, jika seorang pengguna sering membeli sepatu lari dari merek tertentu dengan fitur "bantalan empuk", sistem CBF akan cenderung merekomendasikan sepatu lari lain dari merek yang sama atau berbeda yang juga memiliki deskripsi fitur "bantalan empuk".

Kelebihan utama dari CBF adalah transparansi dan kemandiriannya. Rekomendasi yang dihasilkan dapat dijelaskan dengan mudah ("Kami merekomendasikan ini karena mirip dengan yang Anda beli sebelumnya"), dan sistem tidak memerlukan data dari pengguna lain untuk berfungsi. Namun, pendekatan ini memiliki dua kelemahan kritis. Pertama adalah *overspecialization*, yaitu kecenderungan untuk merekomendasikan item yang sangat mirip dengan apa yang sudah diketahui pengguna, sehingga membatasi penemuan item baru yang beragam (*serendipity*) (Uta et al., 2024). Pengguna akan terjebak dalam preferensi mereka sendiri. Kedua adalah masalah *user cold-start*, di mana sistem kesulitan memberikan rekomendasi yang berarti bagi pengguna baru yang belum memiliki riwayat interaksi atau preferensi yang tercatat (Uta et al., 2024).

B. *Collaborative Filtering* (CF)

Berbeda dengan CBF, *Collaborative Filtering* (CF) bekerja berdasarkan prinsip kearifan kolektif atau "dari mulut ke mulut" secara digital. Pendekatan ini merekomendasikan item kepada seorang pengguna dengan mengidentifikasi pengguna lain yang memiliki selera atau pola perilaku serupa (*nearest neighbor*). Logikanya adalah: "Jika Pengguna A dan Pengguna B menyukai item yang sama di masa lalu, maka item yang disukai oleh Pengguna B di masa depan (tetapi belum dilihat oleh Pengguna A) kemungkinan besar juga akan disukai oleh Pengguna A" (Uta et al., 2024).

Keunggulan terbesar CF adalah kemampuannya untuk menghasilkan rekomendasi yang baru dan tak terduga (*serendipity*). Sistem ini dapat merekomendasikan item dari kategori yang sama sekali berbeda, selama "*neighbor*" pengguna menyukainya. Ini memungkinkan penemuan lintas-genre yang tidak mungkin dilakukan oleh CBF. Namun, CF juga memiliki keterbatasan yang signifikan. Pendekatan ini sangat rentan terhadap masalah *cold-start*, baik untuk pengguna baru (*user cold-start*) maupun untuk item baru (*item cold-start*) yang belum memiliki data interaksi sama sekali (Uta et al., 2024). Selain itu, CF sangat bergantung pada ketersediaan data yang padat. Dalam skenario nyata di mana data interaksi pengguna-item sangat sedikit, kondisi yang dikenal sebagai *data sparsity* ini menyebabkan kinerja CF menurun drastis karena kesulitan menemukan "tetangga" yang andal (Mouhiha et al., 2024).

Keterbatasan yang melekat pada kedua paradigma klasik ini bukanlah sekadar isu teknis, melainkan sebuah kegagalan konseptual yang fundamental. Baik CBF maupun CF

beroperasi pada level pencocokan, CBF mencocokkan atribut item, sementara CF mencocokkan pola perilaku pengguna. Keduanya tidak memiliki kemampuan untuk *memahami* makna, niat, atau konteks di balik interaksi pengguna. Sebagai contoh, ketika seorang pengguna mencari "hadiah unik untuk ayah yang suka berkebun", sistem klasik akan gagal. CBF tidak dapat menghubungkan konsep abstrak seperti "hadiah unik" dengan atribut produk konkret. CF juga akan kesulitan kecuali ada banyak pengguna lain dengan riwayat pencarian dan pembelian yang identik.

2.1.2 Sistem Rekomendasi di *E-commerce*

Pencarian semantik merepresentasikan pergeseran fundamental dalam cara sistem informasi menginterpretasikan kueri pengguna. Pergeseran ini adalah dari pencarian leksikal, yang berfokus pada pencocokan kata kunci secara harfiah, ke pencarian konseptual, yang bertujuan untuk memahami makna dan niat di balik kata-kata tersebut (M. Liang & Niu, 2022).

Metode pencarian tradisional, seperti yang diimplementasikan menggunakan model statistik TF-IDF (*Term Frequency-Inverse Document Frequency*), bekerja dengan memberi bobot pada kata berdasarkan seberapa sering kata itu muncul dalam sebuah dokumen (TF) dan seberapa langka kata itu di seluruh koleksi dokumen (IDF) (M. Liang & Niu, 2022). Meskipun efektif untuk tugas-tugas dasar, pendekatan ini memiliki keterbatasan fundamental: ia tidak memiliki pemahaman semantik. Bagi model TF-IDF, kata "mobil" dan "kendaraan" adalah dua entitas yang sama sekali berbeda dan tidak berhubungan, karena string karakternya berbeda (M. Liang & Niu, 2022). Akibatnya, jika pengguna mencari "kendaraan roda empat", dokumen yang hanya berisi kata "mobil" mungkin tidak akan ditemukan, meskipun secara konseptual sangat relevan.

Pencarian semantik mengatasi kelemahan ini dengan fokus pada makna. Tujuannya adalah untuk memahami bahwa "mobil" adalah jenis dari "kendaraan", dan bahwa kueri seperti "pakaian untuk cuaca dingin" secara konseptual terkait dengan produk seperti "jaket", "sweater", dan "syal", bahkan jika kata-kata tersebut tidak muncul secara eksplisit dalam kueri. Untuk mencapai pemahaman konseptual ini, teknologi modern beralih dari representasi berbasis kata ke representasi berbasis vektor, yang dikenal sebagai *vector embeddings* (Daniel & Martin, 2021).

Inti dari pencarian semantik modern adalah transformasi data, baik itu kueri pengguna, deskripsi produk, atau bahkan gambar menjadi representasi numerik dalam bentuk vektor

(Daniel & Martin, 2021). Proses ini, yang disebut *embedding*, memetakan setiap potongan data ke sebuah titik dalam ruang matematika multidimensi. Di dalam ruang ini, konsep kedekatan spasial secara langsung mencerminkan kemiripan semantik. Artinya, dua potongan data yang memiliki makna serupa akan memiliki representasi vektor yang saling berdekatan.

Untuk mengukur "kedekatan" atau kemiripan antara dua vektor ini, metrik yang paling umum digunakan adalah *Cosine Similarity*. Alih-alih mengukur jarak Euclidean (jarak garis lurus), *cosine similarity* mengukur kosinus dari sudut antara dua vektor. Nilai yang mendekati +1 menunjukkan kemiripan yang sangat tinggi (sudut mendekati 0 derajat), nilai 0 menunjukkan tidak ada korelasi (*ortogonal*), dan nilai yang mendekati -1 menunjukkan pertentangan makna. Rumus matematis untuk *cosine similarity* antara dua vektor A dan B adalah sebagai berikut (Daniel & Martin, 2021):

$$\text{Similarity}(\mathrm{A},\mathrm{B}) = \backslash\cos\{\backslash\text{left}(\backslash\text{theta}\backslash\text{right})\backslash =\}\frac{\mathrm{A}\cdot\mathrm{B}}{\|\mathrm{A}\|-\|\mathrm{B}\|} = \frac{\sum_{i=1}^{n}\mathrm{A}_i\,\mathrm{B}_i}{\sqrt{\sum_{i=1}^{n}\mathrm{A}_{2_i}^{\ 2}}-\sqrt{\sum_{i=1}^{n}B_i^2}} \quad (2.1)$$

di mana A dan B adalah dua vektor yang dibandingkan. Keunggulan utama dari cosine similarity adalah independensinya terhadap magnitudo vektor, sehingga sangat cocok untuk membandingkan teks dengan panjang yang berbeda. Pergeseran ke pencarian berbasis vektor adalah lompatan konseptual yang fundamental, menciptakan " *shared embedding space*" di mana berbagai jenis data teks dan gambar dapat dibandingkan secara semantik, yang menjadi prasyarat mutlak bagi sistem multimodal.

2.1.3 *Vector Embeddings*

Vector embedding adalah konsep inti yang mendasari hampir semua aplikasi kecerdasan buatan modern, mulai dari pemrosesan bahasa alami hingga visi komputer. Teknologi ini menyediakan mekanisme untuk mengubah data mentah yang tidak terstruktur menjadi format yang dapat dipahami dan diproses oleh mesin (Daniel & Martin, 2021).

Secara definitif, *vector embedding* adalah proses transformasi data mentah seperti kata, kalimat, paragraf, atau bahkan gambar utuh menjadi sebuah representasi numerik dalam bentuk vektor yang padat (*dense vector*) di dalam ruang multidimensi (Daniel & Martin, 2021).Vektor ini, yang terdiri dari serangkaian angka *floating-point*, berfungsi sebagai "sidik jari" semantik dari data asli. *Embeddings* ini tidak dibuat secara acak. Mereka dihasilkan oleh model *deep*

learning, khususnya arsitektur *transformer*, yang telah dilatih pada dataset berskala masif. Selama proses pelatihan, model belajar untuk menangkap hubungan, konteks, dan nuansa semantik dari data. Hasilnya adalah sebuah ruang vektor (dikenal sebagai *embedding space* atau *latent space*) di mana posisi setiap vektor memiliki makna. Prinsip fundamental yang mendasari *vector embeddings* adalah "kedekatan spasial mencerminkan kemiripan semantik" (Daniel & Martin, 2021). Artinya, dua entitas data yang memiliki makna atau konsep yang mirip akan memiliki representasi vektor yang berdekatan di dalam *embedding space*.

Fenomena ini memungkinkan dilakukannya operasi matematis yang intuitif pada konsep-konsep abstrak. Contoh klasik yang sering dikutip untuk mengilustrasikan kekuatan ini adalah analogi linguistik: dengan melakukan operasi vektor("Raja") - vektor("Pria") + vektor("Wanita"), hasilnya akan menjadi sebuah vektor yang sangat dekat dengan vektor("Ratu") di dalam *embedding space* (Daniel & Martin, 2021) . Ini menunjukkan bahwa model tidak hanya menghafal kata, tetapi juga memahami hubungan konseptual di antara mereka (misalnya, hubungan gender dan status kerajaan).

Manfaat terbesar dari *vector embeddings* melampaui sekadar peningkatan kualitas pencarian teks. Peran paling transformatif dari *embeddings* adalah sebagai bahasa perantara universal untuk kecerdasan buatan. Teknologi ini memungkinkan modalitas data yang secara fundamental berbeda, seperti piksel dalam gambar dan karakter dalam teks, untuk direpresentasikan dan dibandingkan dalam satu kerangka konseptual yang sama.

Secara tradisional, gambar (*pixel matrix*) dan teks (*sequence of characters*) adalah tipe data yang tidak kompatibel. Mustahil untuk secara langsung membandingkan keduanya. Namun, dengan melatih model *embedding* multimodal, baik gambar maupun teks dapat dipetakan ke dalam ruang vektor bersama. Di dalam ruang ini, vektor yang merepresentasikan gambar seekor anjing akan berada sangat dekat dengan vektor yang merepresentasikan frasa teks "seekor anjing yang lucu". Ini berarti kita sekarang dapat melakukan operasi matematis, seperti menghitung *cosine similarity*, antara sebuah gambar dan sebuah kalimat.

Kemampuan lintas-modalitas ini adalah lompatan konseptual yang menjadi fondasi mutlak bagi sistem canggih seperti yang diusulkan dalam penelitian ini. Tanpa adanya *embeddings* universal yang dapat menyatukan representasi dari berbagai modalitas, sistem RAG multimodal yang mampu memahami kueri teks dan mengambil produk berdasarkan gambar dan deskripsinya tidak akan mungkin terwujud. Inilah jembatan yang menghubungkan dunia

visual dan tekstual, memungkinkan mesin untuk "memahami" keduanya berdasarkan makna konseptual, bukan lagi representasi permukaan.

2.1.4 Model *Multimodal (Vision-Language Models)*

Setelah memahami bahwa data dari modalitas yang berbeda dapat direpresentasikan dalam format vektor yang seragam, langkah logis berikutnya adalah melatih model untuk memahami hubungan antara vektor-vektor tersebut, khususnya antara teks dan gambar. Model yang memiliki kemampuan ini dikenal sebagai model multimodal atau *Vision-Language Models* (VLM).

Kebutuhan akan multimodalitas sangat krusial dalam domain *e-commerce*. Pengalaman berbelanja online secara inheren bersifat multimodal; pengguna tidak hanya membaca deskripsi produk (teks), tetapi juga secara intensif melihat foto produk (gambar) (Hendriksen, 2022) . Seringkali, niat pembelian dipicu oleh inspirasi visual, namun diekspresikan melalui bahasa, baik dalam bentuk kueri pencarian teks maupun percakapan dengan asisten virtual.[16] Oleh karena itu, sebuah sistem rekomendasi yang benar-benar efektif harus mampu memproses dan menghubungkan kedua modalitas ini secara simultan untuk dapat menangkap niat pengguna secara holistik dan akurat (Hendriksen, 2022).

Survei komprehensif mengenai *multimodal deep learning* menunjukkan bahwa arsitektur VLM modern umumnya bersifat modular (Erukude et al., 2025). Arsitektur ini biasanya terdiri dari dua jalur *encoder* yang terpisah, yang bekerja secara paralel:

A. *Image Encoder*

Komponen ini bertugas memproses data visual. Arsitektur yang umum digunakan adalah *Vision Transformer* (ViT) atau varian dari *Convolutional Neural Network* (CNN) seperti ResNet. *Encoder* ini mengubah data piksel dari sebuah gambar menjadi sebuah *embedding* vektor.

B. *Image Encoder*

Komponen ini memproses data tekstual. Arsitektur yang dominan adalah model berbasis *Transformer*, seperti BERT atau variannya, yang mengubah urutan kata menjadi *embedding* vektor.

Tujuan utama dari pelatihan VLM adalah memproyeksikan *embeddings* dari kedua *encoder* ini ke dalam sebuah ruang laten bersama (*shared latent space*). Di dalam ruang ini, *embedding* dari sebuah gambar dan *embedding* dari deskripsi teks yang relevan akan dipaksa untuk berdekatan. Sebagai contoh, *embedding* dari gambar seekor kucing yang sedang bermain bola benang harus berada sangat dekat dengan *embedding* dari teks "kucing oranye bermain dengan benang wol"(Erukude et al., 2025).

Salah satu kerangka kerja paling berpengaruh dalam pelatihan VLM adalah CLIP *(Contrastive Language-Image Pre-training).* CLIP menggunakan metode pembelajaran kontrastif (*contrastive learning*) untuk melatih *image encoder* dan *text encoder* secara bersamaan pada jutaan pasangan gambar-teks yang dikumpulkan dari internet. Tujuan utamanya adalah memaksimalkan kemiripan antara *embedding* dari pasangan gambar-teks yang benar-benar cocok (*positive pairs*) dan pada saat yang sama meminimalkan kemiripan untuk semua pasangan lain yang tidak cocok (*negative pairs*) dalam satu *batch* pelatihan (Erukude et al., 2025). Secara tradisional, CLIP mencapai ini dengan menggunakan fungsi *loss* berbasis *softmax*.

Sebagai evolusi dari arsitektur berbasis CLIP, SigLIP *(Sigmoid Loss for Language Image Pre-training)* diusulkan oleh (Zhai et al., 2023) untuk mengatasi beberapa keterbatasan konseptual dan komputasi dari pendekatan *softmax (Erukude et al., 2025)*. Perbedaan fundamental antara keduanya terletak pada fungsi *loss* yang digunakan.

Pilihan antara *softmax* dan *sigmoid* bukan hanya sekadar optimisasi teknis, melainkan mencerminkan perbedaan filosofi yang mendalam dalam tugas pencocokan. Fungsi *loss softmax* pada CLIP melakukan normalisasi secara global terhadap semua skor kemiripan dalam satu *batch*. Ini berarti skor untuk satu pasangan gambar-teks sangat bergantung pada semua pasangan lainnya, menciptakan sebuah kompetisi di mana model dipaksa untuk memilih satu "pemenang" atau kecocokan terbaik per gambar(Zhai et al., 2023). Pendekatan ini dapat dianalogikan seperti kuis pilihan ganda: "Dari semua pilihan ini, mana yang paling cocok?". Hal ini juga menciptakan ketergantungan yang kuat pada ukuran *batch* yang sangat besar dan beragam untuk mencapai performa yang baik.

Sebaliknya, fungsi *loss sigmoid* pada SigLIP beroperasi pada setiap pasangan gambar-teks secara independen. Ia mengubah tugas menjadi serangkaian masalah klasifikasi biner: untuk setiap pasangan, model hanya perlu menjawab pertanyaan "apakah pasangan ini cocok atau

tidak?" tanpa dipengaruhi oleh pasangan lain (Zhai et al., 2023). Filosofi ini lebih mirip dengan proses verifikasi atau pemberian label, di mana bisa ada banyak jawaban "ya" yang valid.

Dalam konteks *e-commerce*, filosofi *sigmoid* secara fundamental lebih selaras dengan sifat masalah. Sebuah gambar produk, misalnya "sepatu lari biru", bisa saja cocok dengan beberapa deskripsi tekstual yang berbeda seperti "sepatu untuk jogging", "sneakers biru yang nyaman", atau "alas kaki atletik untuk pria". Sifat kompetitif dari *softmax* yang memaksa satu pilihan kurang ideal untuk skenario ini, sedangkan sifat *sigmoid* yang memungkinkan banyak kecocokan valid secara bersamaan jauh lebih cocok. Oleh karena itu, pemilihan SigLIP sebagai *model embedding* dalam penelitian ini didasarkan pada tiga pilar justifikasi yang kuat:

A. Efisiensi dan Skalabilitas Pelatihan

Karena tidak memerlukan normalisasi global, SigLIP secara signifikan mengurangi ketergantungan pada ukuran *batch*, memungkinkan pelatihan yang lebih efisien dengan *batch size* yang lebih kecil atau penskalaan ke *batch size* yang jauh lebih besar (Zhai et al., 2023).

B. Performa yang Lebih Unggul

Studi empiris oleh (Zhai et al., 2023) menunjukkan bahwa SigLIP tidak hanya lebih efisien tetapi juga mencapai performa yang lebih baik atau setidaknya sebanding dengan CLIP, terutama pada rezim ukuran *batch* yang lebih kecil atau sangat besar (Erukude et al., 2025).

C. Keselarasan Konseptual

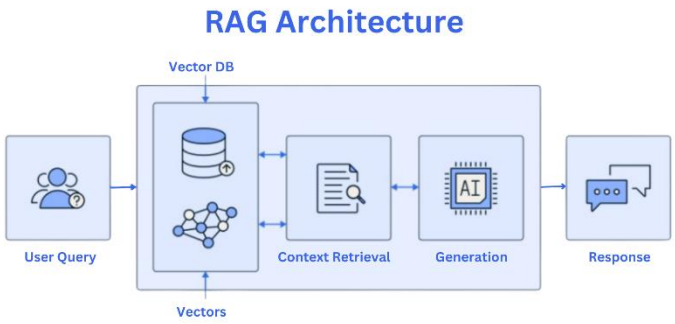
Filosofi penilaian independen dari *sigmoid* lebih cocok untuk tugas pencarian dan rekomendasi di dunia nyata, di mana relevansi sering kali tidak bersifat eksklusif (Sanguigni et al., 2025).

2.1.5 *Retrieval-Augmented Generation (RAG)*

Retrieval-Augmented Generation (RAG) adalah arsitektur inti yang diimplementasikan dan dievaluasi dalam penelitian ini. RAG merupakan sebuah paradigma yang dirancang untuk meningkatkan kemampuan *Large Language Models* (LLM) dengan menggabungkan kekuatan memori parametrik internal mereka dengan akses dinamis ke memori non-parametrik eksternal.

LLM seperti seri Gemini menyimpan sejumlah besar pengetahuan faktual yang mereka pelajari selama proses pelatihan di dalam parameter jaringan saraf mereka (*Parametric memory*). Namun, pengetahuan ini memiliki beberapa kelemahan kritis. Pertama, pengetahuan tersebut bersifat statis dan terikat pada waktu data pelatihan berakhir, sehingga tidak mencakup informasi atau peristiwa terbaru (Abootorabi et al., 2025). Kedua, pengetahuan ini tidak dapat diperbarui dengan informasi baru tanpa melalui proses *fine-tuning* yang sangat mahal secara komputasi. Ketiga, dan yang paling problematik, LLM rentan terhadap halusinasi, yaitu kecenderungan untuk menghasilkan informasi yang terdengar sangat meyakinkan tetapi sebenarnya salah, tidak faktual, atau dibuat-buat (Lewis et al., 2020).

RAG, yang diperkenalkan secara formal oleh (Lewis et al., 2020), dirancang secara spesifik untuk mengatasi masalah-masalah ini. Arsitektur RAG melengkapi LLM dengan kemampuan untuk "mencari" atau "mengambil" (*retrieve*) informasi yang relevan dari sumber pengetahuan eksternal (misalnya, basis data produk, Wikipedia, atau kumpulan dokumen lainnya) secara *real-time* sebelum menghasilkan sebuah respons. Dengan "mendasarkan" (*grounding*) responsnya pada data eksternal yang konkret dan terkini, RAG dapat menghasilkan jawaban yang jauh lebih akurat, faktual, dapat diverifikasi, dan relevan dengan konteks saat ini.



Gambar 2.1 Arsitektur RAG

Sumber: *freecodecamp.org*

Berdasarkan Gambar 2.1, Arsitektur RAG pada dasarnya terdiri dari tiga komponen utama yang bekerja dalam sebuah alur kerja yang terkoordinasi (Lewis et al., 2020) :

A. *Retriever* (Pengambil Informasi):

Komponen ini adalah tulang punggung dari sistem RAG. Ketika pengguna memberikan kueri (misalnya, "Saya butuh laptop ringan untuk bekerja dari kafe"), kueri tersebut pertama-tama diubah menjadi *vector embedding*. *Retriever* kemudian menggunakan *embedding* ini untuk melakukan pencarian kemiripan (misalnya, menggunakan *cosine similarity*) terhadap indeks vektor di dalam sebuah basis data pengetahuan eksternal (dalam kasus ini, basis data produk e-commerce). Hasilnya adalah daftar dokumen atau potongan informasi (*chunks*) dengan peringkat teratas (*top-K*) yang paling relevan secara semantik dengan kueri pengguna (Lewis et al., 2020).

B. *Augmenter* (Pengaya Konteks):

Ini adalah langkah krusial di mana informasi yang telah diambil oleh *retriever* digabungkan dengan kueri asli pengguna. Proses ini biasanya menggunakan sebuah templat *prompt* yang telah dirancang dengan cermat (*prompt engineering*). Tujuannya adalah untuk menciptakan sebuah *prompt* baru yang jauh lebih informatif dan kaya konteks. *Prompt* yang diperkaya ini memberikan instruksi yang jelas kepada LLM, membingkai masalah, dan menyediakan semua konteks yang diperlukan (misalnya, "Berdasarkan deskripsi produk berikut: [...], jawab pertanyaan pengguna: 'Saya butuh laptop ringan untuk bekerja dari kafe'").

C. *Generator* (Pembangkit Respons):

Komponen ini adalah LLM itu sendiri (misalnya, model Gemini). *Generator* menerima *prompt* yang telah diperkaya oleh *augmenter* dan bertugas untuk mensintesis informasi tersebut menjadi sebuah respons yang koheren, relevan, dan disajikan dalam format bahasa yang alami dan bersifat percakapan. Alih-alih hanya mengandalkan pengetahuan internalnya yang statis, LLM kini bertindak sebagai mesin penalaran yang canggih, memproses konteks yang diberikan secara *real-time* untuk merumuskan jawaban yang didasarkan pada fakta dari sumber eksternal.

Melihat RAG hanya sebagai pipa pemrosesan data yang linear (*Retrieve -> Augment -> Generate*) adalah sebuah penyederhanaan. Dalam aplikasi yang canggih, RAG berfungsi sebagai sebuah paradigma orkestrasi dinamis. *Retriever* tidak hanya mengambil data secara

pasif, tetapi secara aktif membangun ruang kerja pengetahuan untuk LLM. *Augmenter* bukan sekadar penggabung teks, melainkan seorang arsitek *prompt* yang secara strategis mengarahkan proses penalaran LLM. Dan *Generator* tidak hanya menjawab; ia mensintesis, membandingkan, dan bahkan dapat menyimpulkan adanya kesenjangan dalam informasi yang diberikan. Kemampuan ini berpotensi memicu putaran *retrieval* lanjutan, sebuah konsep yang dieksplorasi dalam arsitektur RAG tingkat lanjut seperti *Self-RAG (Hendriksen, 2022)*. Fokus penelitian ini pada penciptaan "dialog konsultatif" secara inheren mengadopsi pandangan RAG sebagai kerangka kerja orkestrasi yang interaktif ini, bukan sekadar pipa linear sederhana.

2.1.6 *Vector Database*

Agar arsitektur RAG dapat berfungsi secara efisien, diperlukan infrastruktur basis data khusus yang dioptimalkan untuk menangani *vector embeddings*. Basis data relasional tradisional, yang dirancang untuk menyimpan dan melakukan kueri pada data terstruktur dalam baris dan kolom, sama sekali tidak efisien untuk tugas pencarian kemiripan pada jutaan vektor berdimensi tinggi. Melakukan pencarian semacam itu pada basis data relasional akan setara dengan memindai setiap baris satu per satu, sebuah operasi yang terlalu lambat untuk aplikasi *real-time*. Di sinilah *vector database* berperan.

Vector database adalah sistem basis data yang dirancang khusus untuk menyimpan, mengindeks, dan melakukan kueri terhadap *vector embeddings* dengan kecepatan dan efisiensi tinggi (Hendriksen, 2022). Operasi fundamentalnya adalah pencarian kemiripan (*similarity search*), yang bertujuan untuk menemukan vektor-vektor dalam basis data yang paling "dekat" (berdasarkan metrik seperti *cosine similarity*) dengan sebuah vektor kueri.

Untuk mencapai efisiensi yang diperlukan untuk aplikasi interaktif, *vector database* tidak melakukan pencarian penuh (*exhaustive search*). Sebaliknya, mereka mengandalkan algoritma *Approximate Nearest Neighbor* (ANN). Sesuai dengan namanya, algoritma ANN tidak menjamin akan menemukan tetangga yang benar-benar paling dekat, tetapi akan menemukan tetangga yang "*close enough*" atau kemungkinan besar paling dekat "*nearest*" (Hendriksen, 2022). Pengorbanan sedikit akurasi ini menghasilkan peningkatan kecepatan pencarian yang dramatis, seringkali beberapa kali lipat lebih cepat, yang sangat penting untuk menjaga pengalaman pengguna yang responsif.

Salah satu algoritma ANN yang paling canggih, populer, dan banyak digunakan dalam *vector database* modern adalah HNSW (*Hierarchical Navigable Small World*) (Munyampirwa et al., 2025) . HNSW adalah algoritma berbasis graf yang membangun struktur data cerdas untuk memungkinkan navigasi yang sangat cepat melalui ruang vektor. Cara kerjanya dapat dianalogikan dengan sistem navigasi jalan raya.

Bayangkan semua vektor produk dalam basis data sebagai kota-kota di sebuah peta. Melakukan pencarian dengan memeriksa setiap kota satu per satu akan sangat lambat. HNSW mengatasi ini dengan membangun beberapa lapisan "jalan" di atas peta tersebut:

A. Struktur Hierarkis (H):

HNSW membangun beberapa graf berlapis. Lapisan paling atas adalah yang paling renggang, hanya menghubungkan beberapa "kota besar" yang sangat berjauhan. Ini berfungsi sebagai "jalan tol" atau "penerbangan antarbenua" yang memungkinkan pencarian untuk melompat dari satu wilayah ruang vektor ke wilayah lain yang jauh dengan sangat cepat. Setiap lapisan di bawahnya menjadi semakin padat, seperti "jalan permukiman" yang menghubungkan kota-kota yang lebih kecil dan lebih berdekatan (Munyampirwa et al., 2025).

B. Navigasi (*Navigable Small World*):

Proses pencarian dimulai dari titik masuk acak di lapisan teratas (jalan tol). Algoritma secara serakah (*greedy search*) bergerak di sepanjang koneksi jarak jauh untuk secepat mungkin mendekati wilayah target (area di sekitar vektor kueri). Ketika tidak ada lagi tetangga yang lebih dekat di lapisan saat ini, ia akan "turun" ke lapisan di bawahnya yang lebih padat untuk melakukan pencarian yang lebih halus. Proses ini berulang hingga mencapai lapisan terbawah “jalan permukiman”, di mana pencarian yang sangat akurat dilakukan di sekitar kandidat terbaik untuk menemukan tetangga terdekat yang sebenarnya (Munyampirwa et al., 2025).

Dengan mengimplementasikan indeks berbasis HNSW, sistem yang dibangun dalam penelitian ini mampu melakukan pencarian kemiripan multimodal pada jutaan produk secara efisien dan dalam hitungan milidetik, yang merupakan prasyarat teknis absolut untuk menciptakan pengalaman pengguna yang mulus dan responsif dalam sebuah aplikasi RAG interaktif.

2.2 Tinjauan Pustaka

Pemahaman mengenai tantangan dari sisi pengguna dan dasar teori teknologi pencarian semantik telah menyediakan landasan yang diperlukan. Namun, untuk mengetahui pendekatan metodologi yang sesuai untuk membandingkan efektivitas sistem perlu dilakukan studi terdahulu yang relevan. Keberhasilan sebuah sistem pencarian di *e-commerce* tidak hanya ditentukan oleh akurasi teknis, tetapi persepsi pengguna terhadap relevansi juga menjadi penting untuk diamati. Oleh karena itu, kajian pustaka ini dilakukan untuk mencari dan memilih metodologi evaluasi yang sesuai dengan menganalisis pendekatan yang digunakan dalam penelitian-penelitian terdahulu. Analisis ini bertujuan untuk mengidentifikasi kekuatan dan keterbatasan dari evaluasi yang ada dan membenarkan pilihan metodologis untuk penelitian ini.

Tabel 2. 1 Tinjauan Pustaka

Judul Penelitian	Domain	Model / Teknologi yang Digunakan	Metodologi Evaluasi
Fashion-RAG: Multimodal fashion image editing via retrieval-augmented generation (Sanguigni et al., 2025)	E-commerce (Fashion)	RAG Multimodal	Kuantitatif, berfokus pada kualitas gambar yang dihasilkan.
Design and Research of Cross-Border E-Commerce Short Video Recommendation System Based on Multi-Modal Fusion Transformer Model (Hu, 2024)	E-commerce (Video)	Transformer Multimodal	Kuantitatif (Akurasi, F1-score).

Rationale-Guided Retrieval Augmented Generation for Medical Question Answering (Sohn et al., 2024)	Medis	RAG (RAG2)	Kuantitatif, berfokus pada keandalan dan akurasi faktual.
Retrieval Augmented Generation Evaluation in the Era of Large Language Models: A Comprehensive Survey (Gan et al., 2025)	Umum (Survei)	Tinjauan Literatur	Meringkas metrik kuantitatif (Presisi, Recall, ROUGE).
A survey on conversational recommender systems (Jannach, 2021)	Umum (Survei)	Tinjauan Literatur	Meringkas metrik kualitatif/HCI (Kualitas Dialog, Kepuasan Pengguna).

Analisis terhadap penelitian-penelitian terdahulu pada Tabel 2.1 menunjukkan sebuah pola yang kompleks namun jelas. Di satu sisi, terdapat fokus yang kuat pada evaluasi kuantitatif yang bersifat sistem-sentris, di mana pemilihan metodologi selaras dengan objektif penelitian untuk mengukur performa teknis. Studi oleh (Hu, 2024) misalnya, secara tepat menggunakan metrik akurasi dan F1-score untuk mengevaluasi sistem rekomendasi video. Demikian pula, penelitian dalam domain non-komersial seperti medis oleh (Sohn et al., 2024) juga berfokus pada metrik kuantitatif untuk mengukur keandalan dan kebenaran faktual dari sistem RAG.

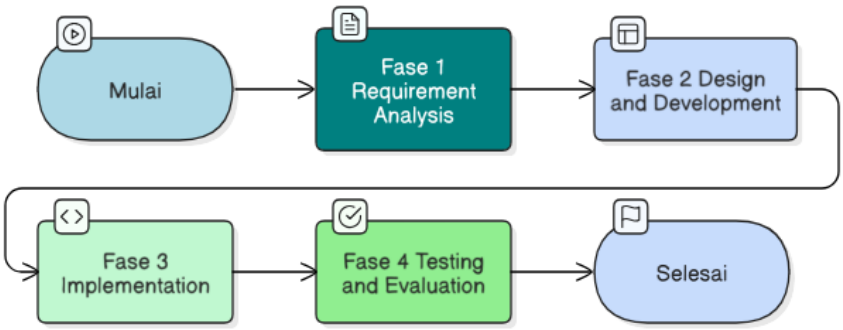
Di sisi lain, penelitian seperti survei oleh (Yu et al., 2024) menunjukkan bahwa evaluasi kualitatif dan berpusat pada pengguna (HCI) memang telah diakui kepentingannya, terutama dalam konteks sistem percakapan (Yu et al., 2024). Namun, pendekatannya sering kali diterapkan pada sistem yang lebih sederhana dan belum menyentuh arsitektur RAG multimodal yang canggih. Pola ini juga ditemui pada penelitian yang dilakukan dalam domain e-commerce.

Studi oleh (Sanguigni et al., 2025),meskipun menggunakan RAG multimodal dalam konteks e-commerce, memiliki objektif utama untuk mengevaluasi kualitas output generatif secara teknis, bukan pengalaman interaksi pengguna.

Kombinasi temuan ini secara jelas menyoroti adanya kesenjangan yang spesifik: kurangnya penelitian yang menjadikan persepsi dan pengalaman pengguna sebagai objektif dan metodologi evaluasi inti untuk sistem RAG multimodal dalam konteks *e-commerce* umum. Penelitian ini berperan untuk mengisi kesenjangan tersebut. Dengan menetapkan pemahaman persepsi pengguna sebagai objektif inti, maka pemilihan metodologi evaluasi kualitatif menjadi sebuah keharusan, bukan sekadar pelengkap. Pendekatan ini memungkinkan studi ini untuk memahami “mengapa” dan “bagaimana” pengguna merasakan relevansi dan kualitas interaksi, sebuah aspek yang penting namun secara inheren berada di luar cakupan indikator kuantitatif tradisional.

BAB III
METODOLOGI PENELITIAN

Penelitian ini menggunakan metodologi Incremental Model sebagai kerangka kerja pengembangan aplikasi untuk implementasi sistem rekomendasi *e-commerce* dengan pencarian *multimodal*. Pendekatan ini dipilih karena sangat sesuai dengan sifat penelitian yang melibatkan pembangunan beberapa modul fungsional yang berbeda namun saling terkait (pencarian kata kunci, semantik, dan RAG). *Incremental Model* memungkinkan pengembangan dilakukan secara bertahap, di mana setiap fungsionalitas (*increment*) dapat dibangun, diuji secara internal, dan distabilkan sebelum melanjutkan ke fungsionalitas yang lebih kompleks. Alur kerja yang terstruktur ini mengurangi risiko teknis dan memastikan setiap komponen berfungsi dengan baik sebelum diintegrasikan. Kerangka kerja ini melibatkan serangkaian tahapan sistematis, termasuk *Requirement Analysis*, *Design & Development*, *Implementation*, dan *Testing & Evaluation*, untuk mengembangkan sebuah aplikasi yang mengintegrasikan teknologi RAG *Multimodal* serta tiga fitur pencarian *baseline*, dan kemudian melakukan evaluasi kualitatif terhadap pengalaman pengguna dalam berinteraksi dengan keempat fitur tersebut.



Gambar 3.1 Alur metode *Incremental Model*

Gambar 3.1 dalam penelitian ini mengacu pada alur metodologi *Incremental Model*, yang menjadi kerangka kerja utama dalam pengembangan sistem. Metodologi ini bersifat inkremental, di mana proses pengembangan dipecah menjadi beberapa fase sekuensial yang saling membangun. Alur dimulai dengan Fase 1: *Requirement Analysis* untuk mendefinisikan kebutuhan fungsional dan antarmuka sistem secara menyeluruh. Hasil dari fase ini menjadi landasan bagi Fase 2: *Design & Development*, di mana rancangan arsitektur, *use case*, dan skema *database* dirumuskan. Selanjutnya, pada Fase 3: *Implementation*, rancangan tersebut

diwujudkan menjadi kode fungsional. Proses diakhiri dengan Fase 4: *Testing & Evaluation*, di mana aplikasi yang telah dibangun diuji secara internal sebelum dievaluasi secara kualitatif dengan pengguna. Pendekatan inkremental ini memungkinkan pembangunan yang terstruktur dan terverifikasi di setiap tahap. Dengan demikian, penelitian ini memberikan kontribusi dalam pemahaman praktis tentang penerapan *Incremental Model* dalam pengembangan sistem e-commerce dengan teknologi RAG, sekaligus menghasilkan wawasan empiris mengenai pengalaman pengguna terhadap berbagai metode pencarian produk.

3.1 Fase 1: *Requirement Analysis*

Fase *Requirement Analysis* dalam siklus *Incremental Model* penelitian ini berfokus pada penetapan lingkup dan definisi kebutuhan yang diperlukan sistem. Kegiatan utama dalam fase ini adalah analisis kebutuhan sistem, di mana peneliti melakukan identifikasi dan perumusan kebutuhan fungsional dan kebutuhan tampilan antarmuka pengguna. Proses analisis ini mendasarkan diri pada temuan-temuan pada Bab II dan observasi terhadap tantangan pencarian di platform e-commerce.

3.1.1 Analisis Kebutuhan Sistem

Sistem yang dikembangkan adalah aplikasi berbasis web yang mengintegrasikan empat fitur pencarian. Pemilihan tumpukan teknologi didasarkan pada kebutuhan untuk pengembangan yang cepat, performa tinggi, dan akses ke model AI mutakhir. *Framework* Streamlit dipilih karena kemampuannya untuk membangun aplikasi web interaktif dengan cepat, yang sangat ideal untuk lingkup penelitian. Model SigLIP (google/siglip-so400m-patch14-384) dipilih karena performanya yang unggul dalam menyelaraskan representasi semantik antara teks dan gambar. Gemini 2.5 Pro dipilih sebagai *generator* karena kemampuan generatifnya yang canggih dan pemahaman konteks yang mendalam. Terakhir, PostgreSQL dengan ekstensi pgvector dipilih karena merupakan solusi *open-source* yang kuat, mengintegrasikan penyimpanan data relasional dan data vektor dalam satu sistem, sehingga menyederhanakan arsitektur. Detail dari kebutuhan sistem yang dikembangkan adalah sebagai berikut.

A. **Kebutuhan Fungsional**

Sistem harus menyediakan empat fitur pencarian yang berbeda. Keberadaan empat fitur ini bertujuan untuk menciptakan sebuah lingkungan terkontrol di mana setiap metode

pencarian dapat dievaluasi secara terisolasi dan dibandingkan satu sama lain untuk menjawab rumusan masalah penelitian.

1. Pencarian Berbasis Kata Kunci: Fitur ini diwajibkan ada untuk merepresentasikan metode pencarian paling dasar dan umum di *e-commerce*. Fungsinya adalah sebagai *baseline* untuk menangani skenario pencarian di mana pengguna memiliki niat yang jelas dan mengetahui kata kunci spesifik.
2. Pencarian Semantik Teks: Fitur ini diperlukan untuk menjembatani kesenjangan ketika pengguna tidak mengetahui kata kunci yang tepat tetapi memahami konsep atau atribut produk yang diinginkan. Ini memungkinkan evaluasi terhadap pencarian berbasis makna.
3. Pencarian Semantik Gambar: Fitur ini esensial untuk memenuhi kebutuhan pencarian berbasis inspirasi visual, sebuah skenario umum di mana kata-kata tidak cukup untuk mendeskripsikan produk yang diinginkan.
4. Pencarian Percakapan (RAG Multimodal): Fitur ini merupakan puncak dari sistem yang dibangun. Keberadaannya bertujuan untuk mensimulasikan interaksi konsultatif dan menangani kueri yang ambigu serta kompleks, yang tidak dapat diselesaikan secara efektif oleh tiga metode pencarian lainnya.

B. Kebutuhan Tampilan Antarmuka

Desain tampilan antarmuka pengguna (UI) harus mendukung tujuan utama penelitian, yaitu evaluasi aplikasi RAG multimodal.

1. Struktur Berbasis Tab

   Antarmuka harus dirancang dengan empat tab yang jelas dan terpisah, di mana setiap tab merepresentasikan satu fitur pencarian. Hal ini memungkinkan pengguna untuk berinteraksi dengan setiap metode secara terisolasi dan memudahkan proses perbandingan selama sesi evaluasi.
2. Komponen Input yang Sesuai

   Setiap tab harus memiliki komponen input yang sesuai dengan fungsinya. Tab pencarian kata kunci dan semantik teks memerlukan sebuah kolom input teks. Tab pencarian semantik gambar memerlukan mekanisme unggah berkas. Tab Asisten Chat RAG memerlukan antarmuka percakapan yang dapat menerima input teks dan unggahan gambar.

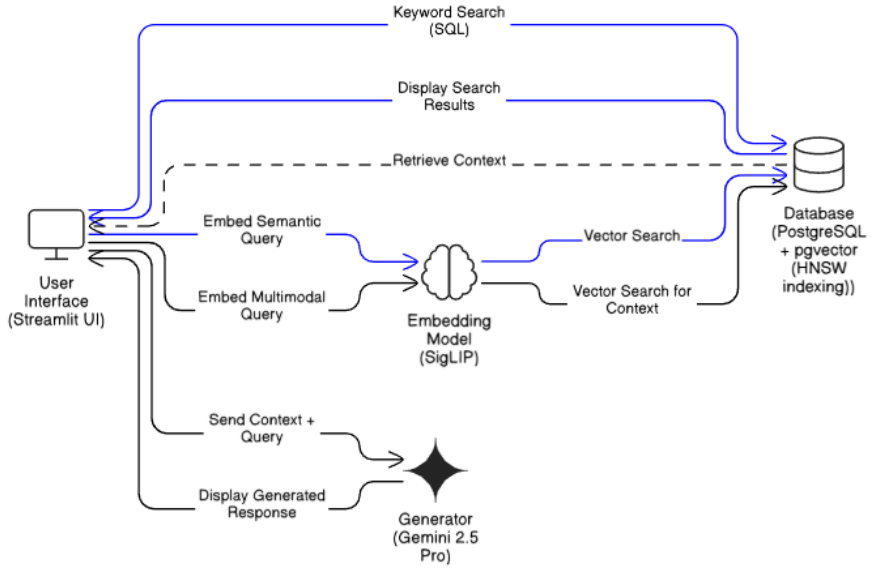
.

3.2 Fase 2: *Design & Development*

Pada fase ini, dijelaskan secara mendetail mengenai desain arsitektur sistem yang dikembangkan. Fase *Design & Development* ini merupakan tahap di mana dilakukan perancangan konsep-konsep dasar yang akan merepresentasikan alur data, interaksi pengguna dengan sistem, dan struktur data. Beberapa konsep dasar yang dibuat pada fase ini adalah desain arsitektur sistem, *use case diagram*, dan desain skema *database*.

3.2.1 Desain Arsitektur Sistem

Pada penelitian ini, sistem diimplementasikan untuk mendukung dua alur fungsional utama: alur pencarian *baseline* (kata kunci dan semantik) dan alur RAG Multimodal. Prosesnya adalah kueri pencarian yang ditulis pengguna akan diproses sesuai fitur yang digunakan. Untuk pencarian semantik dan RAG, kueri akan ditransformasikan menjadi nilai vektor menggunakan model SigLIP, kemudian dilakukan pencarian kemiripan dengan data produk di *database* pgvector. Untuk fitur RAG, hasil pencarian ini kemudian digunakan sebagai konteks oleh model Gemini 2.5 Pro untuk menghasilkan respons. Berikut adalah detail gambaran umum dari sistem yang diimplementasikan.



Gambar 3. 2 *Flow* Arsitektur Sistem.

Gambar 3.2 mengilustrasikan arsitektur sistem yang dikembangkan. Arsitektur ini terdiri dari beberapa komponen utama yang saling berinteraksi:

A. User Interface (Streamlit UI): Komponen ini berfungsi sebagai jembatan interaksi utama antara pengguna dan sistem. Perannya adalah untuk menangkap input pengguna (teks dan/atau gambar) melalui salah satu dari empat tab yang tersedia, mengirimkan

input tersebut ke komponen *backend* yang sesuai, dan kemudian merender atau menampilkan hasil yang diterima. *Framework* Streamlit dipilih karena kemampuannya untuk memfasilitasi pengembangan antarmuka yang fungsional dengan cepat, sehingga memungkinkan peneliti untuk lebih fokus pada logika *backend* dan alur kerja sistem.

B. Embedding Model (SigLIP): Komponen ini merupakan inti dari kemampuan pemahaman semantik sistem. Fungsinya ada dua: pertama, pada tahap pra-pemrosesan, model ini digunakan untuk mengubah seluruh data produk (teks dan gambar) menjadi representasi vektor numerik (*embedding*) yang kemudian disimpan di *database*. Kedua, pada saat penggunaan, model ini mentransformasikan kueri teks atau gambar dari pengguna menjadi vektor secara *real-time*. Model SigLIP dipilih karena performanya yang unggul sebagai *vision-language model*, yang mampu menyelaraskan makna antara teks dan gambar dalam satu ruang vektor yang sama, sebuah kapabilitas yang krusial untuk pencarian multimodal.

C. Database (PostgreSQL + pgvector**):** Komponen ini berfungsi sebagai pusat penyimpanan data. Perannya adalah untuk menyimpan baik data metadata produk (seperti nama, merek, dan fitur) dalam format relasional standar, maupun data vektor *embedding* yang dihasilkan oleh model SigLIP. Dengan ekstensi pgvector dan indeks HNSW, *database* ini mampu melakukan dua jenis pencarian: pencarian SQL standar untuk fitur kata kunci, dan pencarian kemiripan vektor yang sangat efisien untuk fitur semantik dan RAG. Pemilihan PostgreSQL yang terintegrasi dengan pgvector bertujuan untuk menyederhanakan arsitektur dengan menggunakan satu sistem basis data yang dapat menangani kedua jenis data tersebut.

D. Generator (Gemini 2.5 Pro): Komponen ini secara eksklusif digunakan dalam alur RAG Multimodal. Fungsinya adalah untuk menerima konteks (berupa informasi produk yang relevan dari hasil pencarian) dan kueri asli pengguna, lalu menghasilkan respons percakapan yang koheren, informatif, dan bersifat konsultatif. Model Gemini 2.5 Pro dipilih karena kemampuan generatifnya yang canggih, pemahaman konteks yang mendalam, dan kemampuannya untuk memberikan justifikasi, yang sangat penting untuk mengubah interaksi pencarian dari sekadar transaksional menjadi dialogis.

Alur kerja sistem ini melibatkan serangkaian langkah yang berbeda tergantung pada fitur yang diaktifkan. Proses diawali ketika *User Interface* menerima sebuah input. Untuk alur fitur *baseline* (panah biru), prosesnya lebih langsung. Jika input berupa teks untuk

Keyword Search, aplikasi akan membangun dan mengeksekusi kueri SQL ILIKE ke *Database*. Jika input berupa teks atau gambar untuk *Semantic Search*, aplikasi akan mengirimkan input tersebut ke *Embedding Model* untuk menghasilkan sebuah *output* berupa vektor kueri. Vektor ini kemudian digunakan untuk melakukan *Vector Search* di *Database*.

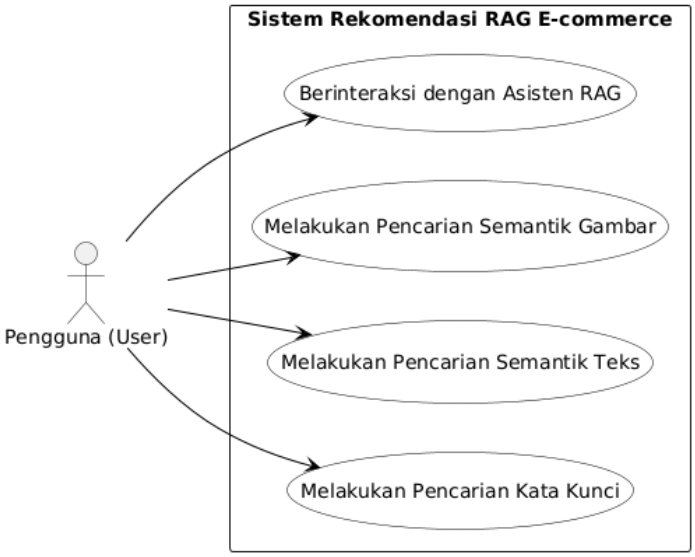
Hasil dari *query* ke *Database* pada alur *baseline*, baik dari SQL maupun pencarian vektor, adalah sebuah *output* berupa daftar ID produk yang relevan. Aplikasi kemudian melakukan *query* sekunder untuk mengambil metadata lengkap (nama, gambar, dll.) berdasarkan daftar ID tersebut. Data ini disusun menjadi sebuah *output* akhir berupa objek data terstruktur (JSON) yang berisi daftar produk. Objek data ini kemudian dikirim kembali ke *User Interface* untuk dirender sebagai daftar produk yang dapat dilihat.

Untuk alur RAG Multimodal (panah hitam dan putus-putus), prosesnya melibatkan langkah *generation* tambahan. Tahap *retrieval* diawali serupa dengan pencarian semantik: *User Interface* menerima input multimodal dan mengirimkannya ke *Embedding Model* untuk diubah menjadi vektor. Vektor ini digunakan untuk melakukan *Vector Search for Context* di *Database*, yang menghasilkan *output* berupa daftar ID produk relevan. Tahap berikutnya adalah pengambilan konteks, di mana aplikasi mengambil metadata lengkap untuk ID tersebut, menghasilkan sebuah *output* berupa data konteks yang terstruktur.

Pada tahap *generation*, aplikasi mengambil data konteks ini beserta kueri asli sebagai *input*. Keduanya digabungkan untuk membentuk sebuah *prompt* yang detail, yang kemudian dikirim sebagai *input* ke *Generator*. *Generator* memproses *prompt* ini dan menghasilkan *output* berupa respons percakapan dalam format terstruktur. Akhirnya, pada fase *output*, respons ini dikirim kembali ke *User Interface* untuk dirender sebagai sebuah jawaban konsultatif, melengkapi siklus RAG.

3.2.2 *Use Case Diagram*

Sistem ini dirancang menggunakan UML *Use Case Diagram* untuk memodelkan interaksi antara aktor dan fungsionalitas utama sistem. Diagram *use case* dari sistem ini memiliki satu aktor dan empat aktivitas utama. Aktor tersebut adalah Pengguna (*User*), yang merepresentasikan partisipan dalam sesi evaluasi. Aktivitas yang telah didefinisikan dan dibutuhkan oleh sistem dapat dilihat pada Gambar 3.3 berikut.



Gambar 3. 3 *Use case diagram*

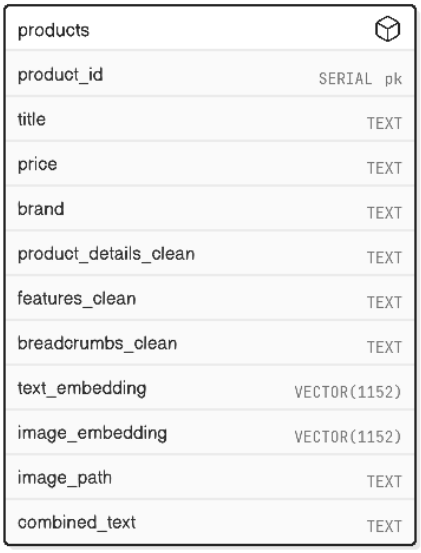
Pada Gambar 3.3, setiap *use case* diuraikan mulai dari aktor yang terlibat dan deskripsi dari *use case* terkait pada Tabel 3.1 Penjelasan *Use Case Diagram* berikut:

Tabel 3. 1 Penjelasan *Use Case Diagram*

No.	**Aktor**	**Use Case**	**Deskripsi**
1	User	Melakukan Pencarian Kata Kunci	Pengguna memasukkan kueri teks spesifik untuk menemukan produk berdasarkan pencocokan literal.
2	User	Melakukan Pencarian Semantik Teks	Pengguna memasukkan kueri teks abstrak untuk menemukan produk berdasarkan relevansi makna.
3	User	Melakukan Pencarian Semantik Gambar	Pengguna mengunggah sebuah gambar untuk menemukan produk yang mirip secara visual.
4	User	Berinteraksi dengan Asisten RAG	Pengguna melakukan percakapan dengan sistem, memberikan input teks dan/atau gambar untuk mendapatkan rekomendasi produk yang konsultatif.

3.2.3 Desain Skema *Database*

Desain skema *database* memberikan gambaran hubungan antara entitas dalam basis data. Mengingat fokus penelitian adalah pada sistem pencarian produk, *Entity Relationship Diagram* (ERD) dirancang secara minimalis untuk hanya mencakup entitas yang relevan, yaitu tabel products.



Gambar 3.4 *Entity Relationship Diagram*

Berikut penjelasan detail mengenai skema tabel products yang digunakan dalam penelitian ini. Tabel ini menyimpan informasi detail mengenai setiap produk yang tersedia dalam sistem. Setiap kolom dirancang untuk memenuhi kebutuhan spesifik, baik sebagai data mentah untuk *embedding*, sebagai metadata untuk ditampilkan kepada pengguna, maupun untuk mendukung fungsionalitas pencarian.

1. product_id (SERIAL): Berfungsi sebagai *Primary Key* yang unik untuk setiap produk, memastikan integritas data.
2. title (TEXT): Menyimpan nama produk yang akan ditampilkan kepada pengguna dan menjadi salah satu sumber utama untuk *text embedding*.
3. price (TEXT): Menyimpan harga produk. Tipe data teks dipilih untuk mengakomodasi berbagai format mata uang dan simbol tanpa kehilangan informasi.
4. brand (TEXT): Menyimpan merek produk, yang juga merupakan informasi penting untuk *text embedding* dan ditampilkan kepada pengguna.

5. product_details_clean (TEXT): Menyimpan detail teknis atau spesifikasi produk yang telah melalui proses pra-pemrosesan.
6. features_clean (TEXT): Menyimpan poin-poin fitur utama produk yang telah dibersihkan, menjadi input penting untuk *embedding*.
7. breadcrumbs_clean (TEXT): Menyimpan jalur kategori produk yang telah dibersihkan, memberikan konteks tambahan mengenai klasifikasi produk.
8. text_embedding (VECTOR(1152)): Menyimpan representasi vektor dari gabungan informasi tekstual produk. Dimensi 1152 disesuaikan dengan *output* dari model SigLIP.
9. image_embedding (VECTOR(1152)): Menyimpan representasi vektor dari gambar utama produk, dengan dimensi yang sama untuk memungkinkan perbandingan multimodal.
10. image_path (TEXT): Menyimpan referensi atau *path* ke lokasi penyimpanan gambar produk untuk ditampilkan di antarmuka.
11. combined_text (TEXT): Menyimpan gabungan dari beberapa kolom teks (title, brand, features, breadcrumbs) yang digunakan sebagai sumber tunggal untuk proses generasi text_embedding, memastikan konsistensi.

Selain struktur tabel di atas, dua indeks HNSW (*Hierarchical Navigable Small World*) juga dibuat pada kolom text_embedding dan image_embedding. Penggunaan indeks ini krusial untuk mempercepat proses pencarian kemiripan vektor (*vector similarity search*) secara signifikan, yang merupakan inti dari fungsionalitas pencarian semantik dan RAG.

3.3 Desain dan Prosedur Evaluasi Sistem

Untuk menjawab rumusan masalah penelitian, evaluasi sistem dilakukan melalui pendekatan metode campuran (mixed-methods) yang menggabungkan analisis kuantitatif dan kualitatif. Tahap kuantitatif bertujuan untuk mengukur kinerja relevansi dari setiap fitur pencarian secara objektif, sementara tahap kualitatif bertujuan untuk memahami pengalaman, persepsi, dan preferensi pengguna secara mendalam.

3.3.1 Persiapan Perangkat Evaluasi

Langkah pertama dalam evaluasi adalah mempersiapkan perangkat yang diperlukan untuk pengujian, yang meliputi pembuatan dataset pengujian (ground truth) dan daftar skenario tugas.

A. Dataset Ground Truth

Untuk memungkinkan pengukuran kuantitatif, sebuah dataset ground truth dibuat. Dataset ini terdiri dari 60 kueri pengujian yang dirancang untuk merepresentasikan berbagai tingkat kompleksitas dan modalitas (misalnya kueri kata kunci sederhana, kueri semantik abstrak, dan kueri multimodal). Untuk setiap kueri, peneliti secara manual mengidentifikasi dan memberi label produk dari database yang dianggap sebagai hasil paling relevan. Dataset ini digunakan dalam evaluasi metrik retrieval seperti Precision@10, Recall@10, MAP@10, dan nDCG@10.

Selain itu, ditambahkan 20 kueri representatif yang dipilih dari berbagai kategori. Kueri-kueri ini digunakan khusus untuk mengevaluasi kualitas jawaban sistem berdasarkan metrik Answer Relevance dan Faithfulness, di mana penilaian dilakukan secara manual.

B. Skenario Tugas Kualitatif

Lima skenario tugas naratif dirancang untuk memandu partisipan dalam sesi evaluasi kualitatif. Skenario-skenario ini dibuat untuk secara alami mendorong partisipan menggunakan keempat fitur pencarian dalam konteks yang realistis.

3.3.2 Prosedur Evaluasi Kuantitatif

Evaluasi kuantitatif dilakukan untuk mendapatkan baseline kinerja dari setiap fitur pencarian, dengan fokus pada kualitas pengambilan informasi (retrieval) dan kualitas generasi respons (untuk RAG).

A. Eksekusi Pengujian

Setiap kueri dari dataset ground truth dijalankan pada keempat fitur pencarian. Untuk fitur RAG, yang diuji adalah tahap retrieval-nya (produk yang diambil sebelum dikirim ke LLM) dan respons akhir yang dihasilkan.

B. Pengukuran Metrik

Hasil dari setiap pencarian (daftar 10 produk teratas) dibandingkan dengan ground truth untuk menghitung metrik-metrik berikut:

1. Precision@10

   Metrik ini mengukur seberapa akurat 10 hasil pencarian teratas. Ia menjawab pertanyaan: "Dari 10 produk yang ditampilkan, berapa persen yang benar-benar relevan?". Alasan penggunaannya adalah karena metrik ini secara langsung

merefleksikan kepuasan pengguna di halaman pertama hasil pencarian, di mana presisi yang tinggi sangatlah krusial. Rumusnya adalah:

$$\text{Precision@10} = \frac{\text{Jumlah produk relevan di 10 teratas}}{10} \tag{2.2}$$

Di sini, "produk relevan" didefinisikan berdasarkan ground truth, misalnya dari perilaku pengguna sebelumnya atau anotasi manual. Precision@10 memberikan indikasi langsung mengenai kualitas rekomendasi pada daftar berukuran tetap (top-10).

2. Recall@10

Metrik ini mengukur seberapa lengkap sistem dalam menemukan semua item yang relevan. Ia menjawab pertanyaan: "Dari semua produk relevan yang ada di database, berapa persen yang berhasil ditemukan dalam 10 hasil teratas?". Recall@10 penting untuk memastikan sistem tidak melewatkan banyak produk penting yang mungkin dicari pengguna. Rumusnya adalah:

$$\text{Recall@10} = \frac{\text{Jumlah produk relevan di 10 teratas}}{\text{Total produk relevan di ground truth}} \tag{2.3}$$

Di sini, penyebut adalah jumlah total produk relevan, sementara pembilang adalah jumlah produk relevan yang berhasil direkomendasikan dalam 10 hasil teratas. Recall@10 memastikan sistem tidak melewatkan terlalu banyak produk penting.

3. *Mean Average Precision* (MAP)@10

MAP@10 adalah metrik yang digunakan untuk mengukur kualitas sistem rekomendasi dengan mempertimbangkan urutan kemunculan item relevan pada hasil pencarian. Berbeda dengan presisi biasa, MAP memberikan bobot lebih besar pada item relevan yang muncul di posisi awal. Nilai MAP dihitung sebagai rata-rata dari Average Precision (AP) pada setiap kueri pengujian.

$$\text{MAP} = \frac{1}{\text{U}} \sum_{\text{u}=1}^{\text{U}} \frac{1}{|\text{Relevant Items}_\text{u}|} \sum_{k=1}^{|\text{Recommended Items}_\text{u}|} Precisionk \times rel_u(k) \tag{2.4}$$

Dalam rumus tersebut, U merepresentasikan jumlah total kueri pengujian yang digunakan dalam evaluasi. Simbol |{RelevanItems_u}|menunjukkan jumlah item relevan yang tersedia untuk kueri ke-u, sedangkan |RecommendedItems_u| menunjukkan jumlah item yang direkomendasikan sistem untuk kueri tersebut. Nilai Precision@k mengacu pada presisi pada peringkat ke-k, dan fungsi indikator rel_u(k) bernilai 1 jika item pada peringkat ke-k relevan untuk kueri ke-u, serta bernilai 0 apabila tidak relevan.

4. *Normalized Discounted Cumulative Gain (nDCG)@10*

Ini adalah metrik paling komprehensif yang menilai kualitas peringkat secara keseluruhan. Ia memberi bobot lebih pada hasil relevan yang berada di urutan atas (*Discounted*) dan membandingkan skor sistem dengan skor ideal (*Normalized*). nDCG@10 dipilih karena kemampuannya memberikan evaluasi yang sangat detail terhadap kualitas dan urutan hasil pencarian dengan melalui beberapa tahap dengan yang pertama yaitu *Discounted Cumulative Gain* (DCG). DCG digunakan untuk menjumlahkan skor relevansi dari produk dalam daftar rekomendasi, dengan memberikan bobot lebih tinggi pada produk yang muncul di urutan atas. Bobot ini ditentukan dengan fungsi logaritmik dengan rumus:

$$\text{DCG@10} = \sum_{\text{i=1}}^{10} \frac{rel_i}{\log_2(i+1)} \tag{2.5}$$

Di sini, rel_i adalah skor relevansi item di posisi ke-i. Pembagi \log_2(i+1) berfungsi sebagai penalti, sehingga item relevan di posisi bawah memberi kontribusi lebih kecil dibandingkan item relevan di posisi atas.

Selanjutnya, dihitung *Ideal Discounted Cumulative Gain* (IDCG@10). Nilai ini merepresentasikan DCG maksimum yang dapat dicapai apabila semua produk relevan ditempatkan di urutan paling atas secara ideal. Perhitungan IDCG menggunakan rumus yang sama dengan DCG, namun urutan relevansi disusun dalam bentuk ideal dengan rumus:

$$\text{IDCG@10} = \sum_{\text{i=1}}^{10} \frac{rel_i^*}{\log_2(i+1)} \tag{2.6}$$

Di sini, rel_i* adalah relevansi item pada urutan ideal. IDCG digunakan sebagai pembanding agar skor dapat dinormalisasi.

Tahap terakhir adalah melakukan normalisasi dengan membandingkan nilai DCG dengan IDCG untuk memperoleh nDCG@10. Hasil ini berada pada rentang 0 hingga 1, di mana nilai mendekati 1 menunjukkan bahwa urutan rekomendasi yang dihasilkan sistem semakin mendekati urutan ideal. Dengan demikian, nDCG@10 tidak hanya menilai keberadaan produk relevan, tetapi juga memperhitungkan seberapa baik urutannya dalam daftar rekomendasi dengan rumus:

$$\text{nDCG@10} = \frac{\text{DCG@10}}{\text{IDCG@10}} \tag{2.7}$$

Nilai nDCG berada pada rentang 0 hingga 1. Semakin mendekati 1, semakin baik kualitas urutan rekomendasi, karena semakin mirip dengan urutan ideal.

5. *Faithfulness* (untuk Fitur RAG)

Metrik ini secara spesifik mengukur kebenaran faktual dari respons yang dihasilkan oleh LLM. Ia menjawab pertanyaan: "Apakah semua informasi dalam jawaban yang diberikan oleh AI didasarkan sepenuhnya pada konteks (data produk) yang disediakan?". Alasan pemilihannya adalah untuk mencegah "halusinasi" AI, yaitu ketika model menghasilkan informasi yang salah atau tidak ada dalam sumber data dengan rumus:

$$\text{Faithfulness} = \frac{\text{Jumlah jawaban yang sesuai konteks}}{\text{Total jawaban yang dievaluasi}} \tag{2.8}$$

Nilai Faithfulness diberikan dengan anotasi manual oleh evaluator, di mana setiap jawaban diberi skor 1 jika sesuai dengan konteks (ground truth) dan 0 jika tidak. Nilai akhir diperoleh dengan menghitung rata-rata dari seluruh evaluasi.

6. *Answer Relevance* (untuk Fitur RAG)

Metrik ini menilai sejauh mana jawaban yang dihasilkan AI benar-benar menjawab maksud dari pertanyaan pengguna. Ia menjawab pertanyaan: "Apakah respons dari AI secara langsung dan memuaskan menjawab kueri awal pengguna?". Metrik ini dipilih karena jawaban yang faktual (faithful) tidak akan

berguna jika tidak relevan dengan apa yang ditanyakan pengguna. Ini memastikan bahwa fitur RAG tidak hanya akurat, tetapi juga fungsional dan membantu.

$$\text{Answer Relevance} = \frac{\text{Jumlah jawaban yang relevan}}{\text{Total jawaban yang dievaluasi}} \tag{2.9}$$

Nilai *Answer Relevance* diberikan dengan skor 1 jika jawaban dianggap relevan dengan pertanyaan, dan 0 jika tidak. Dengan cara ini, Answer Relevance memastikan bahwa sistem RAG tidak hanya akurat secara fakta (*faithful*), tetapi juga kontekstual dan sesuai dengan maksud pengguna.

3.3.3 Prosedur Evaluasi Kualitatif

Evaluasi kualitatif dilakukan untuk memperoleh pemahaman yang lebih mendalam mengenai pengalaman pengguna saat berinteraksi dengan sistem. Fokus utama evaluasi ini bukan hanya pada kinerja teknis, tetapi juga pada kepuasan dan relevansi hasil pencarian yang dirasakan oleh pengguna. Dengan pendekatan ini, penelitian dapat mengungkap dimensi subjektif yang tidak tercakup dalam pengukuran kuantitatif, sekaligus memberikan wawasan mengenai kekuatan dan kelemahan dari setiap fitur pencarian, termasuk sistem RAG.

A. Partisipan

Partisipan yang akan dilibatkan dalam evaluasi adalah individu yang dipilih menggunakan teknik *purposive sampling*. Kriteria utama adalah individu yang merupakan pengguna aktif platform e-commerce (melakukan pembelian alas kaki online setidaknya 1-2 kali dalam 3 bulan terakhir) dan memiliki kemampuan verbal yang baik untuk mengartikulasikan pengalamannya. Jumlah partisipan yang ditargetkan adalah 5 hingga 8 orang untuk mencapai saturasi data.

B. Prosedur Evaluasi

Evaluasi penelitian ini dirancang sebagai sebuah asesmen kualitatif untuk memahami secara mendalam pengalaman pengguna. Prosedur evaluasi dilakukan melalui beberapa tahapan sistematis sebagai berikut:

1. Pengenalan Sistem

Pada awal sesi, setiap partisipan diberikan pengenalan singkat mengenai antarmuka pengguna (UI) dan empat fitur pencarian yang telah dikembangkan. Hal ini bertujuan untuk memastikan partisipan memahami platform sebelum memulai tugas evaluasi.

2. Penugasan Berbasis Skenario

Partisipan diminta untuk melakukan serangkaian tugas pencarian produk alas kaki. Untuk mengevaluasi performa sistem secara komprehensif di berbagai skenario, partisipan diberikan 5 skenario yang dirancang untuk secara alami mendorong penggunaan fitur-fitur yang berbeda.

Dalam prosedur evaluasi kualitatif, partisipan diberikan beberapa skenario tugas untuk mengeksplorasi beragam fitur pencarian yang tersedia. Pertama, mereka diminta menjalankan skenario spesifik, misalnya mencari produk dengan instruksi "*Coba temukan produk 'Adiddas Shoes'*", yang mendorong penggunaan pencarian kata kunci. Selanjutnya, partisipan diberi skenario konseptual berupa kueri "*Coba cari 'sepatu hiking yang bagus untuk pemula dan tahan air'*", yang diarahkan pada penggunaan pencarian semantik berbasis teks.

Selain itu, disiapkan skenario visual, di mana partisipan tidak mengetahui nama atau merek produk, tetapi diperlihatkan gambar sepatu boots unik dengan instruksi "*Coba cari produk yang mirip*", sehingga memanfaatkan fitur pencarian semantik berbasis gambar. Kemudian, pada skenario multimodal, partisipan diberikan gambar sepatu lari dengan tambahan instruksi *"tetapi ingin mencari yang solnya lebih empuk dan cocok untuk lari jarak jauh",* yang mendorong pemanfaatan kemampuan sistem RAG multimodal.

Terakhir, partisipan menjalani skenario ambigius dan konsultatif, misalnya "*Anda ingin memberikan hadiah untuk sahabat yang suka sneakers gaya vintage warna cerah. Coba minta beberapa ide*", yang dirancang untuk menilai sejauh mana sistem mampu memberikan rekomendasi bersifat kreatif sekaligus relevan dengan preferensi pengguna.

3. Evaluasi Sistem

Selama partisipan mengerjakan skenario tugas, diterapkan metode *Think-Aloud Protocol,* di mana mereka diminta untuk menyuarakan pikiran, perasaan, dan kebingungan mereka secara verbal. Setelah sesi uji coba selesai, data kualitatif digali lebih dalam melalui wawancara Mendalam Semi-Terstruktur. Fokus utama wawancara

adalah untuk menggali umpan balik mendalam mengenai persepsi pengguna, terutama pada kesesuaian hasil dengan niat pencarian, kemudahan penggunaan setiap fitur, dan tingkat kepuasan mereka secara keseluruhan. Pendekatan ini memungkinkan dilakukannya analisis tematik untuk memahami secara mendalam bagaimana perbedaan antar metode pencarian memengaruhi pengalaman dan preferensi pengguna.

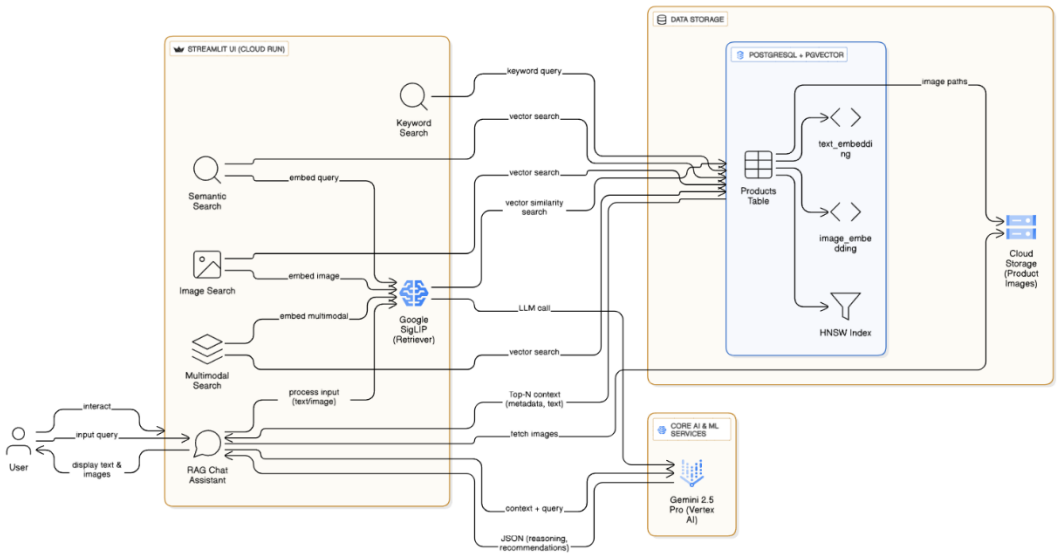
BAB IV
HASIL DAN PEMBAHASAN

Bab ini menjelaskan hasil implementasi dari tahap desain yang telah dijabarkan pada bab sebelumnya, diikuti dengan pemaparan dan pembahasan mendalam mengenai temuan evaluasi kualitatif. Sesuai dengan kerangka kerja yang diuraikan pada Bab III, pengembangan aplikasi ini mengadopsi metodologi *Incremental Model* yang bersifat sekuensial untuk memastikan pembangunan yang terstruktur dan bertahap. Oleh karena itu, pemaparan akan diawali dengan rincian hasil implementasi yang terbagi ke dalam empat *increment* utama, mencakup persiapan data hingga implementasi fitur RAG *Multimodal*.

Setelah menguraikan hasil pengembangan sistem, fokus pembahasan akan beralih pada temuan penelitian. Bagian ini akan menyajikan hasil evaluasi yang dirancang untuk membandingkan secara langsung efektivitas keempat fitur pencarian yang bertujuan untuk menjawab rumusan masalah mengenai pengalaman pengguna dan nilai tambah dari setiap metode pencarian.

4.1 Arsitektur dan Alur Kerja Sistem

Sebagai kelanjutan dari fase perancangan pada Bab III, bab ini diawali dengan visualisasi arsitektur final dari sistem yang berhasil diimplementasikan. Untuk memberikan gambaran menyeluruh mengenai alur kerja sistem dari perspektif teknis dan pengguna akhir, sebuah diagram arsitektur dirancang. Sebagaimana diilustrasikan pada Gambar 4.1, diagram ini memetakan interaksi antara pengguna dan komponen-komponen utama aplikasi, sekaligus menjabarkan proses yang terjadi selama setiap fungsi pencarian dijalankan.



Gambar 4.1 Realisasi Arsitektur Sistem

Alur kerja diawali ketika pengguna berinteraksi dengan *Streamlit UI*. Pengguna dapat memilih salah satu dari empat fitur pencarian yang tersedia. Untuk fitur *baseline* (*Keyword Search*, *Semantic Search*, *Image Search*), alur prosesnya relatif langsung: pengguna memberikan input, sistem memprosesnya (baik melalui kueri SQL maupun pencarian vektor), dan *Data Storage* mengembalikan daftar produk yang relevan untuk ditampilkan.

Fungsionalitas inti, yakni *RAG Chat Assistant*, memiliki alur yang lebih kompleks. Proses dimulai saat pengguna memasukkan kueri multimodal pada antarmuka percakapan. Setelah kueri diterima, sistem pertama-tama melakukan tahap *retrieval* dengan mengubah input menjadi representasi vektor menggunakan model SigLIP. Vektor ini kemudian digunakan untuk menjalankan *vector search* di dalam *Data Storage* (PostgreSQL + pgvector) untuk mendapatkan *top-N context* berupa metadata produk yang paling relevan. Konteks ini kemudian digabungkan dengan kueri asli pengguna dan dikirim ke *Core AI & ML Services*, yaitu Gemini 2.5 Pro, menggunakan *library* SDK Python. Pada tahap akhir, Gemini mengembalikan respons berupa JSON yang berisi justifikasi dan rekomendasi, yang kemudian ditampilkan secara koheren bersama gambar produk yang relevan (diambil dari *Cloud Storage*) kepada pengguna.

4.2 Fase 3: Implementation

Pada sub-bab ini, peneliti menguraikan proses implementasi sistem pencarian e-commerce. Pengembangan sistem ini dilakukan dengan pendekatan bertahap sesuai metodologi

Incremental Model, yang secara garis besar terbagi menjadi empat *increment* utama untuk memastikan pembangunan yang terstruktur.

4.2.1 Increment 1: Implementasi Fondasi Data

Increment pertama dalam siklus pengembangan difokuskan pada pembangunan fondasi data. Tujuan utama tahap ini adalah untuk memastikan seluruh data produk telah diproses dan disimpan dalam format yang optimal untuk diakses oleh fitur-fitur pencarian. Aktivitas utama pada *increment* ini mencakup akuisisi data, pra-pemrosesan, generasi *embedding*, dan *database seeding*.

A. Akuisisi dan Data Preprocessing

Untuk memastikan model *embedding* dapat menghasilkan representasi vektor yang akurat, proses implementasi diawali dengan akuisisi data produk dari sumber yang relevan. Data mentah ini kemudian harus melewati tahap pra-pemrosesan yang krusial untuk menghilangkan *noise* dan inkonsistensi.

1. Sumber Data

Dataset produk yang digunakan dalam penelitian ini diperoleh melalui proses scraping dari situs e-commerce publik (Amazon UK) menggunakan layanan dari Oxylabs. Proses ini menghasilkan 11.604 data produk unik dengan atribut seperti url, title, price, dan features. Cuplikan data mentah dapat dilihat pada tabel berikut:

Tabel 4. 2 Data Mentah

URL	Title	ASIN	Price	Brand	Product Details	Breadcrumb	Images List	Features
https://www.amazon.co.uk/dp/B08BLP231K	Geox Junior Sandals (Boy's)	B08BLP23K	£50.00	Geox	Package Dimensions: 31.2 x 21.4 x 11.4 cm; 820 Grams Date First Available: 22 Jun. 2020	Shoes / Boys' Athletic & Outdoor Shoes	["https://...	{"Outer Material": "Synthetic", "Sole": "Rubber", "Closure": "Hook & Loop",

					Manufacturer: Geox ASIN: B08BLP231K Item model number: J1524B02214 Country of origin: Vietnam Department: Boy's			"Technology": "Respira Breathable Sole"}
https://www.amazon.co.uk/dp/B08N587YZ9	Fila Women's Disruptor Sneakers	B08N587Y	£49.57	Fila	Product Dimensions: 32.51 x 21.84 x 12.19 cm; 952.54 Grams Date First Available: 6 Jun. 2020 Manufacturer: Fila ASIN: B089RQLYNW Item model number:	Shoes / Women's Fashion & Athletic Trainers	["https://...	{"Outer Material": "Leather", "Sole": "Ethylene Vinyl Acetate", "Closure": "Lace-Up", "Style": "Chunky Platform"}

					5JM009 48-990 Department: Women's			

2. Data *Preprocessing*

Data mentah yang telah dikumpulkan kemudian melewati tahap pra-pemrosesan yang esensial. Langkah-langkah yang dilakukan meliputi penghapusan baris data duplikat, penanganan nilai yang hilang *(null values*), dan pembersihan karakter-karakter non-standar atau noise dari kolom teks. Seperti yang dilakukan pada kode ini:

```
import pandas as pd
import os
import re

file_local_bersih = '/content/DATAFINAL_PREPROCESSED.csv'
file_input = '/content/DATAFILE/DATAFINAL - NOT CLEAN - 
stillduplicate_new.csv'

def extract_main_feature(feature_str):
    try:
        items = re.findall(r'[\'"](.*?)[\'"]', str(feature_str))
        if len(items) > 1:
            return items[1]
        elif items:
            return items[0]
        return None
    except:
        return None

try:
    print("--- Memulai Proses Baca Data ---")
    if not os.path.exists(file_input):
        raise FileNotFoundError(f"File input '{file_input}' tidak ditemukan. 
Pastikan path sudah benar.")
    df = pd.read_csv(file_input)
    print(f"Jumlah baris data awal: {len(df)}")

    columns_to_drop = ['url', 'asin', 'images_lis']
    columns_exist = [col for col in columns_to_drop if col in df.columns]
    df_processed = df.drop(columns=columns_exist)
```

```
    print(f"1. Kolom {columns_exist} telah dihapus.")

    if 'breadcrumb' in df_processed.columns:
        df_processed['breadcrumbs_clean']                                    =
df_processed['breadcrumb'].str.replace('/', ' > ', regex=False)
        df_processed = df_processed.drop(columns=['breadcrumb'])
        print("2. Kolom 'breadcrumb' dibersihkan ('/' diganti dengan ' > ').")

    if 'features' in df_processed.columns:
        df_processed['features_clean']                                       =
df_processed['features'].apply(extract_main_feature)
        df_processed = df_processed.drop(columns=['features'])
        print("3. Informasi diekstrak dari 'features' ke kolom baru
'features_clean'.")

    rows_before_na = len(df_processed)
    df_processed.dropna(inplace=True)
    print(f"4. Menghapus baris dengan nilai kosong. Baris berubah dari
{rows_before_na} menjadi {len(df_processed)}.")

    rows_before_dup = len(df_processed)
    df_final = df_processed.drop_duplicates(subset=['title'], keep='first')
    print(f"5. Menghapus duplikat. Baris berubah dari {rows_before_dup}
menjadi {len(df_final)}.")

    final_column_order = [
        'title',
        'brand',
        'product_d',
        'features_clean',
        'breadcrumbs_clean',
        'price'
    ]
    final_column_order_exist = [col for col in final_column_order if col in
df_final.columns]
    df_final = df_final[final_column_order_exist]
    print("6. Urutan kolom telah disesuaikan.")

    output_dir = os.path.dirname(file_local_bersih)
    if output_dir:
        os.makedirs(output_dir, exist_ok=True)
    df_final.to_csv(file_local_bersih, index=False)
    print(f"\n☑ SUKSES! Data yang sudah diproses berhasil disimpan ke
'{file_local_bersih}'.")
```

```
except FileNotFoundError as fnf_error:
    print(f"❌ Error: {fnf_error}")
except Exception as e:
    print(f"❌ Terjadi kesalahan saat memproses file.")
    print(f"Error: {e}")
```

Gambar 4. 2 Kode *Preprocessing* Data

Skrip pada Gambar 4.2 melakukan beberapa tugas penting. Pertama, ia memuat *dataset* mentah dan menghapus kolom-kolom yang tidak relevan untuk penelitian seperti `url` dan `asin`. Kedua, kolom `breadcrumb` yang menggunakan format `/` distandarisasi menjadi format `>` untuk keterbacaan yang lebih baik. Ketiga, informasi penting dari kolom `features` yang memiliki format tidak terstruktur diekstrak menggunakan *regular expression*. Keempat, untuk menjaga kualitas data, semua baris yang mengandung nilai kosong (*null values*) dihapus. Terakhir, untuk menghindari redundansi, baris data duplikat berdasarkan kolom `title` dihilangkan, menghasilkan *dataset* yang bersih dan siap untuk tahap selanjutnya.

Perlu ditekankan bahwa tidak ada proses pra-pemrosesan linguistik lebih lanjut yang diterapkan, seperti normalisasi (misalnya, *stemming* atau *lemmatization*) dan penghapusan *stop words*. Keputusan ini didasarkan pada karakteristik model *embedding* yang digunakan, yaitu `siglip/so400m-patch14-384`, yang dirancang untuk memahami konteks dari teks mentah. Proses pra-pemrosesan yang berlebihan berpotensi mengurangi kekayaan konteks semantik yang terkandung dalam data, yang justru merupakan input krusial bagi model.

B. Implementasi Generasi *Embedding*

Tahap selanjutnya dalam pembangunan fondasi data adalah transformasi data tekstual dan visual menjadi representasi numerik melalui proses generasi embedding. Tujuan dari tahap ini adalah untuk mengonversi setiap produk ke dalam ruang vektor multidimensi, di mana kedekatan antar vektor mencerminkan kemiripan semantik antar produk.

1. Embedding Model

Model SigLIP dengan varian google/siglip-so400m-patch14-384 digunakan untuk menghasilkan representasi vektor (*embedding*) dari data teks dan gambar. Model ini

dipilih karena kemampuannya yang unggul dalam menyelaraskan representasi semantik antara dua modalitas tersebut. Setiap data, baik teks maupun gambar, diubah menjadi vektor dengan dimensi 1152.

2. Proses *Embedding* Teks

Untuk menangani deskripsi produk yang seringkali panjang dan melebihi kapasitas input model, diterapkan strategi *Chunking, Overlapping, and Averaging*. Deskripsi produk yang panjang dipecah menjadi beberapa segmen (*chunks*) yang saling tumpang tindih (*overlap*). Setiap segmen kemudian diubah menjadi vektor secara individual. Vektor-vektor dari semua segmen tersebut kemudian dirata-ratakan untuk menghasilkan satu vektor tunggal yang merepresentasikan keseluruhan deskripsi produk secara komprehensif. Berikut adalah kode yang digunakan untuk menghasilkan *embedding* dari teks:

```
!pip install transformers torch Pillow

from transformers import AutoProcessor, AutoModel, AutoTokenizer
import torch
import pandas as pd
import numpy as np

model = AutoModel.from_pretrained("google/siglip-so400m-patch14-384")
processor = AutoProcessor.from_pretrained("google/siglip-so400m-patch14-384")
tokenizer = AutoTokenizer.from_pretrained("google/siglip-so400m-patch14-384")

csv_path = "/content/DATAFILE/DATAFINAL - NOT CLEAN.csv"
df = pd.read_csv(csv_path)

def combine_text(row):
    text    =    f"{row['title']}    {row['brand']}    {row['features_clean']}
{row['breadcrumbs_clean']}"
    return str(text).strip()

df['combined_text'] = df.apply(combine_text, axis=1)

MAX_TOKENS = 60
OVERLAP = 10

def     chunk_text_by_tokens(text,     tokenizer,     max_tokens=MAX_TOKENS,
overlap=OVERLAP):
    input_ids = tokenizer.encode(text, add_special_tokens=False)
```

```
    chunks = []
    start = 0
    while start < len(input_ids):
        chunk_ids = input_ids[start:start+max_tokens]
        chunk_text = tokenizer.decode(chunk_ids, skip_special_tokens=True)
        safe_ids = tokenizer.encode(chunk_text,
add_special_tokens=False)[:max_tokens]
        safe_text = tokenizer.decode(safe_ids, skip_special_tokens=True)
        assert len(tokenizer.encode(safe_text, add_special_tokens=False)) <=
max_tokens
        chunks.append(safe_text)
        start += max_tokens - overlap
    return chunks

def extract_text_embedding(text, processor, model, tokenizer,
max_tokens=MAX_TOKENS):
    safe_ids = tokenizer.encode(text, add_special_tokens=False)[:max_tokens]
    safe_text = tokenizer.decode(safe_ids, skip_special_tokens=True)
    token_len = len(tokenizer.encode(safe_text, add_special_tokens=False))
    assert token_len <= max_tokens
    inputs = processor(text=[safe_text], return_tensors="pt", padding=True,
truncation=True)
    with torch.no_grad():
        outputs = model.get_text_features(**inputs)
    return outputs.flatten().cpu().numpy()

total_rows = len(df)
embeddings = []
for idx, row in df.iterrows():
    text = row['combined_text']
    text_chunks = chunk_text_by_tokens(text, tokenizer)
    chunk_embeddings = []
    for chunk in text_chunks:
        if chunk.strip():
            emb = extract_text_embedding(chunk, processor, model, tokenizer)
            chunk_embeddings.append(emb)
    if chunk_embeddings:
        averaged_embedding = np.mean(chunk_embeddings, axis=0)
    else:
        averaged_embedding = np.zeros(model.text_embed_dim)
    embeddings.append(averaged_embedding.tolist())
    if (idx+1) % 100 == 0 or (idx+1) == total_rows:
        print(f"{idx+1}/{total_rows} selesai")

embedding_df = pd.DataFrame({
```

```
    'index': df.index,
    'combined_text': df['combined_text'],
    'embedding': embeddings
})
embedding_df.to_csv('/content/text_embeddings_siglip_new.csv', index=False)

print("SEMUA embedding teks sudah dibuat hanya dari chunk <= 60 token (no
direct call to model with >60 tokens). SAFE.")
```

Gambar 4. 3 Kode *Embedding* Teks

Kode pada Gambar 4.3 mengimplementasikan strategi *Chunking, Overlapping, and Averaging*. Proses dimulai dengan memuat model SigLIP beserta *processor* dan *tokenizer*-nya. Selanjutnya, data produk dari berkas CSV dimuat, dan beberapa kolom teks yang relevan (seperti title, brand, features, dan breadcrumbs) digabungkan menjadi satu kolom combined_text untuk setiap produk. Fungsi `chunk_text_by_tokens` bertugas untuk memecah *combined_text* yang panjang menjadi beberapa segmen (*chunks*) dengan panjang token maksimal 60 dan tumpang tindih (*overlap*) sebanyak 10 token untuk menjaga kontinuitas konteks antar segmen. Kemudian, fungsi `extract_text_embedding` mengubah setiap segmen teks menjadi vektor embedding. Proses utama melakukan iterasi pada setiap produk, memecah teksnya menjadi segmen-segmen, menghasilkan embedding untuk setiap segmen, dan kemudian menghitung rata-rata dari semua embedding segmen tersebut untuk menghasilkan satu vektor representatif tunggal untuk produk tersebut. Hasil akhir berupa daftar embedding yang telah dirata-ratakan disimpan ke dalam berkas CSV baru untuk digunakan pada tahap selanjutnya.

3. Proses *Embedding* Gambar

Selain data tekstual, representasi visual dari setiap produk juga diubah menjadi *embedding*. Setiap URL gambar yang valid dari daftar gambar produk diunduh dan diproses. Gambar tersebut kemudian diumpankan ke model SigLIP untuk diubah menjadi satu vektor tunggal. Vektor ini menangkap esensi dan fitur visual dari gambar produk tersebut. Berikut adalah kode yang digunakan untuk menghasilkan *embedding* dari Gambar:

```
from transformers import AutoProcessor, AutoModel
from PIL import Image
import torch
```

```
import os
import time
import pandas as pd

model = AutoModel.from_pretrained("google/siglip-so400m-patch14-384")
processor = AutoProcessor.from_pretrained("google/siglip-so400m-patch14-384")

def extract_image_embedding(image_path):
    image = Image.open(image_path).convert("RGB")
    inputs = processor(images=image, return_tensors="pt", padding=True)
    with torch.no_grad():
        outputs = model.get_image_features(**inputs)
    image_embedding = outputs
    return image_embedding.flatten()

base_folder = "/content/image_list_new"
all_embeddings = []

for subfolder in os.listdir(base_folder):
    subfolder_path = os.path.join(base_folder, subfolder)
    if os.path.isdir(subfolder_path):
        print(f"Memproses folder {subfolder_path}...")
        image_paths = [os.path.join(subfolder_path, fname) for fname in
os.listdir(subfolder_path) if fname.endswith('.jpg')]
        start_time = time.time()
        for image_path in image_paths:
            embedding = extract_image_embedding(image_path)
            all_embeddings.append({'image_path': image_path, 'embedding':
embedding.tolist(), 'folder': subfolder})
        end_time = time.time()
        inference_time = end_time - start_time
        print(f"Waktu inferensi untuk {len(image_paths)} gambar di folder
{subfolder}: {inference_time:.4f} detik")

embedding_df = pd.DataFrame(all_embeddings)
embedding_df.to_csv('/content/image_embeddings_new.csv', index=False)

print("Semua embedding gambar telah disimpan dalam 'image_embeddings.csv'.")
```

Gambar 4. 4 Kode *Embedding* Gambar

Kode pada Gambar 4.4 bertujuan untuk mengekstraksi **embedding** atau representasi vektor dari sekumpulan gambar menggunakan model SigLIP. Proses ini dimulai dengan memuat model *pre-trained* google/siglip-so400m-patch14-384 beserta

processor yang sesuai. *Processor* ini bertugas untuk mengubah gambar mentah menjadi format tensor yang dapat dipahami oleh model.

Fungsi inti `extract_image_embedding` didefinisikan untuk memproses satu gambar. Fungsi ini menerima path sebuah gambar, membukanya, dan menggunakan *processor* untuk mempersiapkannya. Kemudian, model dipanggil untuk menghasilkan *image_features* (vektor embedding) untuk gambar tersebut. Setiap gambar diubah menjadi satu vektor *embedding* tunggal.

Selanjutnya, kode akan melakukan iterasi secara sistematis pada setiap subfolder di dalam direktori base_folder. Untuk setiap file gambar berekstensi .jpg yang ditemukan, fungsi `extract_image_embedding` akan dipanggil. Hasilnya mencakup path gambar, vektor *embedding*, dan nama folder asalnya dikumpulkan dalam sebuah daftar.

Setelah semua gambar dari semua folder berhasil diproses, seluruh data yang terkumpul diubah menjadi sebuah DataFrame menggunakan library Pandas. Akhirnya, DataFrame ini diekspor dan disimpan sebagai sebuah file CSV baru bernama *image_embeddings_new.csv* untuk digunakan pada tahap selanjutnya.

C. Database Seeding

Setelah *embeddings* dihasilkan, dilakukan proses *database seeding*. Sebuah skrip dikembangkan untuk mengisi tabel products di PostgreSQL dengan seluruh data metadata dan vektor yang telah dihasilkan. Pada tahap ini, indeks HNSW juga dibangun pada kolom text_embedding dan image_embedding untuk memastikan performa pencarian vektor yang cepat. Berikut adalah skema *database* yang digunakan:

```
CREATE EXTENSION IF NOT EXISTS vector;

CREATE TABLE products (
    product_id SERIAL PRIMARY KEY,
    title TEXT,
    price TEXT,
    brand TEXT,
    product_details_clean TEXT,
    features_clean TEXT,
    breadcrumbs_clean TEXT,
    text_embedding vector(1152),
    image_embedding vector(1152),
    image_path TEXT,
```

```
    combined_text TEXT
);

CREATE INDEX ON products USING hnsw (text_embedding vector_l2_ops);
CREATE INDEX ON products USING hnsw (image_embedding vector_l2_ops);
```

Gambar 4. 5 Skema *Database*

Kode pada Gambar 4.5 melakukan tiga tugas utama. Pertama, perintah `CREATE EXTENSION IF NOT EXISTS` vector; memastikan bahwa ekstensi pgvector aktif di dalam *database*, yang merupakan prasyarat untuk menggunakan tipe data vector. Kedua, perintah `CREATE TABLE products (...)` dieksekusi untuk membuat tabel products dengan skema yang telah dirancang sebelumnya, yang mencakup kolom untuk metadata dan dua kolom khusus untuk *text_embedding* dan *image_embedding*. Terakhir, dua perintah `CREATE INDEX` dieksekusi untuk membangun indeks HNSW pada kedua kolom vektor. Langkah ini sangat penting untuk optimasi performa, karena indeks HNSW memungkinkan pencarian kemiripan vektor dilakukan secara jauh lebih cepat dibandingkan dengan pencarian sekuensial.

4.2.2 Increment 2: Implementasi Fitur Pencarian Baseline

Untuk menciptakan titik perbandingan yang jelas bagi fitur RAG, *increment* kedua didedikasikan untuk mengimplementasikan fitur-fitur pencarian *baseline* yang terbagi menjadi 4 tab. Setiap fitur, yang mewakili metode pencarian konvensional dan semantik, dibangun sebagai modul terpisah yang berinteraksi dengan fondasi data yang telah disiapkan pada *increment* sebelumnya. Berikut adalah fitur baseline:

A. Pencarian Kata Kunci (Tab1)

Fitur ini dirancang untuk mensimulasikan pencarian berbasis kata kunci yang umum ditemukan di banyak platform *e-commerce*. Alur kerjanya dibagi menjadi dua tahap: pertama, menemukan produk yang relevan berdasarkan nama, dan kedua, merekomendasikan produk lain yang serupa secara semantik dengan produk yang ditemukan.

```
with tab1:
    st.subheader("Pencarian Berdasarkan Nama Produk")
    st.info("💡 Masukkan sebagian nama sepatu (contoh: 'running shoes') untuk
mencari rekomendasi.")
```

```
    selected_shoe_name = st.text_input("Rekomendasikan sepatu yang mirip
dengan:", key="keyword_search")
    recommend_button = st.button("Recommend Me!",
key="keyword_recommend_button")

    if 'keyword_recommendations' not in st.session_state:
        st.session_state.keyword_recommendations = None
    if 'keyword_matched_title' not in st.session_state:
        st.session_state.keyword_matched_title = None

    if recommend_button and selected_shoe_name:
        with st.spinner(f"Mencari produk yang cocok dengan
'{selected_shoe_name}'..."):
            try:
                cur = conn.cursor()
                like_query = "SELECT * FROM products WHERE LOWER(title) LIKE %s
LIMIT 20"
                search_term = f"%{selected_shoe_name.lower()}%"
                cur.execute(like_query, (search_term,))
                rows = cur.fetchall()
                columns = [desc[0] for desc in cur.description]
                df_shoes = pd.DataFrame(rows, columns=columns)

                if df_shoes.empty:
                    st.warning("Tidak ada sepatu yang cocok dengan nama tersebut
di database.")
                    st.session_state.keyword_recommendations = pd.DataFrame()
                    st.session_state.keyword_matched_title = None
                else:
                    st.session_state.keyword_matched_title = selected_shoe_name
                    st.info(f"Menampilkan hasil pencarian untuk:
'{selected_shoe_name}'")
                    st.session_state.keyword_recommendations = df_shoes
            except Exception as e:
                st.error(f"Terjadi kesalahan saat mengambil data: {e}")
                st.session_state.keyword_recommendations = None
    elif recommend_button and not selected_shoe_name:
        st.session_state.keyword_recommendations = None

    if st.session_state.get('keyword_recommendations') is not None:
        if not st.session_state.keyword_recommendations.empty:
            st.success(f"Menampilkan rekomendasi teratas untuk
'{st.session_state.keyword_matched_title}':")
            display_product_grid(st.session_state.keyword_recommendations,
key_prefix="keyword")
```

```
        else:
            st.warning("Tidak ada produk yang ditampilkan.")
    else:
        st.subheader("Jelajahi Produk Kami")
        random_products = get_random_products(conn)
        display_product_grid(random_products, key_prefix="keyword_random")
```

Gambar 4. 6 Kode Pencarian Kata Kunci

Berdasarkan kode program pada Gambar 4.6, antarmuka dan logika untuk fitur pencarian kata kunci diimplementasikan. Pertama, antarmuka pengguna dibangun menggunakan `st.subheader`, `st.info`, dan `st.text_input` untuk menerima masukan dari pengguna. Ketika recommend_button ditekan, sebuah kueri SQL dengan klausa `LIKE` dieksekusi untuk mencari produk yang judulnya mengandung teks input. Hasil kueri diambil dan diubah menjadi sebuah *DataFrame Pandas*. Data ini kemudian disimpan ke dalam `st.session_state.keyword_recommendations` untuk ditampilkan menggunakan fungsi `display_product_grid`. Logika ini juga mencakup penanganan kondisi awal atau pencarian kosong dengan menampilkan produk acak.

B. Pencarian Semantik Teks (Tab2)

Fitur ini memungkinkan pengguna untuk mencari produk menggunakan deskripsi dalam bahasa alami, bukan hanya kata kunci. Tujuannya adalah untuk memahami maksud atau konteks di balik kueri pengguna dan menemukan produk yang relevan secara semantik, bahkan jika kata-kata yang digunakan tidak sama persis dengan yang ada di deskripsi produk.

```
with tab2:
    st.subheader("Pencarian Berdasarkan Deskripsi (Teks)")
    st.info("💡 Jelaskan produk yang Anda cari, contoh: 'sandal nyaman untuk traveling musim panas'.")

    semantic_search_term = st.text_area("Deskripsikan produk yang Anda cari:", key="semantic_search", height=100)
    semantic_recommend_button = st.button("Recommend Me!", key="semantic_recommend_button")

    if 'semantic_recommendations' not in st.session_state:
        st.session_state.semantic_recommendations = None

    if semantic_recommend_button and semantic_search_term:
        with st.spinner("Menganalisis deskripsi Anda dan mencari produk..."):
```

```
            try:
                inputs      =      siglip_processor(text=[semantic_search_term],
return_tensors="pt", padding="max_length")
                with torch.no_grad():
                    input_vector                                                =
siglip_model.get_text_features(**inputs).squeeze(0).tolist()

                if input_vector:
                    vector_string = str(input_vector)
                    rec_query = "SELECT * FROM products WHERE text_embedding IS
NOT NULL ORDER BY text_embedding <=> %s::vector ASC LIMIT 20"
                    df_shoes        =        pd.read_sql(rec_query,        conn,
params=(vector_string,))
                    st.session_state.semantic_recommendations = df_shoes
                else:
                    st.warning("Tidak  dapat  membuat  pemahaman  dari  deskripsi
Anda.")
                    st.session_state.semantic_recommendations = pd.DataFrame()
            except Exception as e:
                st.error(f"Terjadi kesalahan: {e}")
                st.session_state.semantic_recommendations = pd.DataFrame()
    elif semantic_recommend_button and not semantic_search_term:
        st.session_state.semantic_recommendations = None

    if st.session_state.get('semantic_recommendations') is not None:
        st.success("Menampilkan rekomendasi berdasarkan deskripsi Anda:")
        display_product_grid(st.session_state.semantic_recommendations,
key_prefix="semantic")
    else:
        st.subheader("Jelajahi Produk Kami")
        random_products = get_random_products(conn)
        display_product_grid(random_products, key_prefix="semantic_random")
```

Gambar 4. 7 Kode Pencarian Semantik Teks

Berdasarkan kode program pada Gambar 4.7, implementasi pencarian semantik berbasis teks dijelaskan. Antarmuka menerima input deskriptif dari pengguna melalui st.text_area. Setelah tombol ditekan, teks tersebut diproses oleh model SigLIP (`siglip_processor` dan `siglip_model.get_text_features`) untuk diubah menjadi sebuah `input_vector`. Vektor ini kemudian digunakan dalam kueri SQL untuk melakukan pencarian kemiripan menggunakan operator <=> dari pgvector. Hasil pencarian, berupa *DataFrame*, disimpan ke dalam

`st.session_state.semantic_recommendations` dan ditampilkan kepada pengguna.

C. Pencarian Semantik Gambar (Tab3)

Fitur pencarian ini memungkinkan pengguna untuk mengunggah gambar suatu produk. Sistem kemudian akan menganalisis konten visual dari gambar tersebut dan merekomendasikan produk-produk lain dari *database* yang memiliki kemiripan visual paling tinggi.

```
with tab3:
    st.subheader("Pencarian Berdasarkan Gambar")
    st.info("💡 Unggah gambar produk untuk menemukan produk yang mirip secara visual.")

    uploaded_image = st.file_uploader("Unggah gambar produk...", type=["jpg", "jpeg", "png"], key="image_upload")
    multimodal_recommend_button = st.button("Recommend Me!", key="multimodal_recommend_button")

    if 'multimodal_recommendations' not in st.session_state:
        st.session_state.multimodal_recommendations = None

    if multimodal_recommend_button and uploaded_image is not None:
        with st.spinner("Menganalisis gambar dan mencari produk serupa..."):
            try:
                image = Image.open(uploaded_image).convert("RGB")
                inputs = siglip_processor(images=[image], return_tensors="pt")
                with torch.no_grad():
                    input_vector = siglip_model.get_image_features(**inputs).squeeze(0).tolist()

                if input_vector:
                    vector_string = str(input_vector)
                    rec_query = "SELECT * FROM products WHERE image_embedding IS NOT NULL ORDER BY image_embedding <=> %s::vector ASC LIMIT 20"
                    df_shoes = pd.read_sql(rec_query, conn, params=(vector_string,))
                    st.session_state.multimodal_recommendations = df_shoes
                else:
                    st.warning("Tidak dapat membuat pemahaman dari gambar Anda.")
                    st.session_state.multimodal_recommendations = pd.DataFrame()
            except Exception as e:
```

```
                st.error(f"Terjadi kesalahan: {e}")
                st.session_state.multimodal_recommendations = pd.DataFrame()
    elif multimodal_recommend_button and uploaded_image is None:
        st.warning("Silakan unggah gambar terlebih dahulu.")
        st.session_state.multimodal_recommendations = None

    if st.session_state.get('multimodal_recommendations') is not None:
        st.success("Menampilkan rekomendasi berdasarkan gambar Anda:")
        display_product_grid(st.session_state.multimodal_recommendations,
key_prefix="multimodal")
    else:
        st.subheader("Jelajahi Produk Kami")
        random_products = get_random_products(conn)
        display_product_grid(random_products, key_prefix="multimodal_random")
```

Gambar 4. 8 Kode Pencarian Semantik Gambar

Berdasarkan kode program pada Gambar 4.8, alur kerja pencarian berbasis gambar diilustrasikan. Pengguna mengunggah gambar melalui *widget* `st.file_uploader`. Ketika tombol ditekan, gambar dibuka menggunakan pustaka *Pillow* dan diubah menjadi *image embedding* oleh model SigLIP. Vektor yang dihasilkan kemudian digunakan untuk menjalankan kueri pencarian kemiripan pada kolom image_embedding di *database*. Hasilnya disimpan ke `st.session_state.multimodal_recommendations` untuk ditampilkan dalam format *grid*.

D. Pencarian Multimodal Teks dan Gambar (Tab4)

Fitur ini merupakan pencarian *baseline* yang paling canggih, karena mampu menggabungkan informasi dari dua modalitas: teks dan gambar. Dengan menerapkan strategi *late fusion*, sistem memproses input dari setiap modalitas secara independen untuk menghasilkan representasi vektor masing-masing, sebelum menggabungkannya pada tahap akhir untuk melakukan pencarian.

```
with tab4:
    st.subheader("Pencarian Multimodal (Teks + Gambar)")
    st.info("💡 Masukkan deskripsi produk dan/atau unggah gambar untuk mencari
produk yang paling mirip.")

    # Input gambar di atas input teks
    multi_image = st.file_uploader(
        "Unggah gambar produk:",
        type=["jpg", "jpeg", "png"],
```

```
        key="multi_image_upload_vertical"
    )
    if multi_image:
        st.image(multi_image, caption="Preview gambar", width=220)

    multi_text = st.text_area(
        "Deskripsikan produk:",
        key="multi_text_input_vertical",
        height=100
    )

    # Tombol di tengah bawah input
    col_btn1, col_btn2, col_btn3 = st.columns([2, 3, 2])
    with col_btn2:
        multi_search_button = st.button(
            "Recommend Me!",
            key="multi_search_button_vertical",
            use_container_width=True
        )

    if 'multimodal_mix_recommendations' not in st.session_state:
        st.session_state.multimodal_mix_recommendations = None

    if multi_search_button and (multi_text or multi_image is not None):
        with st.spinner("Memproses pencarian multimodal..."):
            try:
                text_vector, image_vector = None, None
                rec_query, query_vector = None, None

                # Proses embedding teks jika ada input teks
                if multi_text:
                    text_inputs = siglip_processor(
                        text=[multi_text], return_tensors="pt",
padding="max_length"
                    )
                    with torch.no_grad():
                        text_vector =
siglip_model.get_text_features(**text_inputs).squeeze(0)
                # Proses embedding gambar jika ada input gambar
                if multi_image is not None:
                    img = Image.open(multi_image).convert("RGB")
                    img_inputs = siglip_processor(images=[img],
return_tensors="pt")
                    with torch.no_grad():
```

```
                        image_vector =
siglip_model.get_image_features(**img_inputs).squeeze(0)

                # --- SOLUSI: Logika Pencarian yang Benar ---
                if text_vector is not None and image_vector is not None:
                    # KASUS 1: Teks dan Gambar ada -> Rata-ratakan vektor, cari
di embedding gambar
                    query_vector = (text_vector + image_vector) / 2
                    rec_query = "SELECT * FROM products WHERE image_embedding
IS NOT NULL ORDER BY image_embedding <=> %s::vector ASC LIMIT 20"
                elif text_vector is not None:
                    # KASUS 2: Hanya Teks -> Cari di embedding teks
                    query_vector = text_vector
                    rec_query = "SELECT * FROM products WHERE text_embedding IS
NOT NULL ORDER BY text_embedding <=> %s::vector ASC LIMIT 20"
                elif image_vector is not None:
                    # KASUS 3: Hanya Gambar -> Cari di embedding gambar
                    query_vector = image_vector
                    rec_query = "SELECT * FROM products WHERE image_embedding
IS NOT NULL ORDER BY image_embedding <=> %s::vector ASC LIMIT 20"

                if query_vector is not None:
                    vector_string = str(query_vector.tolist())
                    df_recs = pd.read_sql(rec_query, conn,
params=(vector_string,))
                    st.session_state.multimodal_mix_recommendations = df_recs
                else:
                    st.warning("Minimal satu input (teks/gambar) diperlukan!")
                    st.session_state.multimodal_mix_recommendations =
pd.DataFrame()

            except Exception as e:
                st.error(f"Terjadi kesalahan: {e}")
                st.session_state.multimodal_mix_recommendations =
pd.DataFrame()
    elif multi_search_button and (not multi_text and multi_image is None):
        st.warning("Isi teks, upload gambar, atau keduanya!")
        st.session_state.multimodal_mix_recommendations = None

    if st.session_state.get('multimodal_mix_recommendations') is not None:
        st.success("Menampilkan rekomendasi hasil pencarian multimodal Anda:")
        display_product_grid(st.session_state.multimodal_mix_recommendations,
key_prefix="multimodal_mix")
    else:
        st.subheader("Jelajahi Produk Kami")
```

```
        random_products = get_random_products(conn)
        display_product_grid(random_products,
key_prefix="multimodal_mix_random")
```

Gambar 4. 9 Kode Pencarian *Multimodal* Teks dan Gambar

Berdasarkan kode program pada Gambar 4.9, strategi *late fusion* untuk pencarian multimodal didemonstrasikan. Setelah menerima input dari st.file_uploader dan st.text_area, sistem secara independen menghasilkan text_vector dan image_vector. Logika inti pada blok if/elif/elif kemudian menggabungkan representasi ini: jika kedua vektor ada, keduanya dirata-ratakan untuk membentuk query_vector gabungan. Jika hanya satu modalitas yang diisi, vektor tunggal tersebut digunakan secara langsung. query_vector akhir inilah yang digunakan untuk melakukan pencarian kemiripan.

4.2.3 Increment 3: Implementasi Fitur RAG Multimodal

Increment ketiga merupakan puncak dari implementasi sistem, dengan fokus pada pembangunan fitur pencarian percakapan yang canggih melalui arsitektur *Retrieval-Augmented Generation* (RAG). Pendekatan ini dirancang untuk meningkatkan kualitas respons model bahasa dengan cara melengkapinya dengan informasi eksternal yang relevan. Implementasi fitur ini dibagi menjadi tiga komponen utama yang bekerja secara berurutan: *Retrieval*, *Augmentation*, dan *Generation*.

A. Logika *Retrieval*

Komponen pertama dalam arsitektur RAG adalah modul *retrieval*, yang bertanggung jawab untuk mengambil informasi relevan dari basis data untuk dijadikan sebagai konteks. Berdasarkan kueri multimodal dari pengguna, sistem akan mencari dan mengambil sekumpulan produk yang paling relevan untuk dianalisis lebih lanjut oleh model bahasa.

```
# Truncate text to avoid overly long inputs
max_text_length = 300
truncated_text = prompt_text[:max_text_length]
text_inputs = siglip_processor(text=[truncated_text], return_tensors="pt",
padding="max_length")
with torch.no_grad():
    text_vector = siglip_model.get_text_features(**text_inputs).squeeze(0)

# Optional image embedding
if image_bytes:
    input_image = Image.open(io.BytesIO(image_bytes))
    image_inputs = siglip_processor(images=[input_image], return_tensors="pt")
```

```
    with torch.no_grad():
        image_vector =
siglip_model.get_image_features(**image_inputs).squeeze(0)

# Late fusion and vector search (top-10)
if text_vector is not None and image_vector is not None:
    query_vector = (text_vector + image_vector) / 2
    search_query = "SELECT * FROM products ORDER BY image_embedding <=>
%s::vector LIMIT 10;"
elif image_vector is not None:
    query_vector = image_vector
    search_query = "SELECT * FROM products ORDER BY image_embedding <=>
%s::vector LIMIT 10;"
elif text_vector is not None:
    query_vector = text_vector
    search_query = "SELECT * FROM products ORDER BY text_embedding <=>
%s::vector LIMIT 10;"

# Fetch and normalize context
vector_string = str(query_vector.tolist())
context_df = pd.read_sql(search_query, conn, params=(vector_string,))
if not context_df.empty:
    temp_df = context_df.copy()
    temp_df['title'] = temp_df['title'].str.lower().str.replace("’", "'",
regex=False)
    temp_df['brand'] = temp_df['brand'].str.lower().str.replace("’", "'",
regex=False)
    if 'features_clean' in temp_df.columns:
        temp_df['features_clean'] =
temp_df['features_clean'].astype(str).str.lower().str.replace("’", "'",
regex=False)
    context_for_llm = temp_df[['title', 'brand', 'price',
'features_clean']].to_json(orient='records', indent=2)
```

Gambar 4.10 Kode Logika *Retrieval*

Berdasarkan kode program pada Gambar 4.10, alur retrieval dimulai setelah sistem menangkap masukan chat (teks) dan unggahan gambar opsional di tab asisten, dengan penyeimbang beban input berupa pemotongan panjang teks maksimum 300 karakter untuk menjaga stabilitas embedding SigLIP pada waktu-jalannya. Sistem mengekstraksi embedding dari teks dan/atau gambar menggunakan SigLIP, bila keduanya ada maka dilakukan *late fusion*, lalu dieksekusi pencarian vektor top-10 pada PostgreSQL+pgvector menggunakan kolom image_embedding untuk kasus gambar/kombo atau text_embedding

untuk kasus teks murni. Produk hasil kandidat kemudian dinormalisasi ke huruf kecil dan diselaraskan tanda kutipnya untuk mengurangi friksi parsing, sebelum diproyeksikan menjadi konteks JSON ringkas berisi *title*, *brand*, *price*, dan *features_clean*.

B. Logika *Augmentation*

Komponen kedua adalah modul *augmentation*, yang bertanggung jawab untuk menyusun *prompt* secara dinamis. Pada tahap ini, informasi yang telah diambil dari basis data (konteks) digabungkan dengan pertanyaan asli pengguna dan serangkaian instruksi yang sangat spesifik untuk memandu model bahasa dalam menghasilkan respons yang akurat dan sesuai format.

```
    if image_bytes and image_mime_type:
        image_analysis_model = GenerativeModel(selected_model)
        image_part = Part.from_data(data=image_bytes, mime_type=image_mime_type)
        response = image_analysis_model.generate_content(
            ["Describe the primary object in this image in one or two words.",
image_part]
        )
        image_content_description = response.text.strip().lower()

    final_prompt = f"""
                        You are 'SoleMate', an expert and friendly AI shopping
assistant for a high-end shoe store.
                     Your primary goal is to help users find the perfect shoes.
You must be conversational and handle various situations gracefully.
                     Analyze the user's query and the retrieved PRODUCT CONTEXT,
then follow these steps to generate a response in JSON format.

                    **Step 1: Determine the User's Intent.**
                    Based on the `USER'S INPUT`, classify the intent into one of
three categories:
                       1. `greeting_or_smalltalk`: The user is making a simple
conversational gesture (e.g., "hello", "thanks", "how are you?").
                     2. `product_query`: The user is asking about shoes, looking
for recommendations, or describing a type of shoe.
                      3. `off_topic_query`: The user's query is about something
completely unrelated to footwear.

                    **Step 2: Formulate a Response Based on the Intent.**
                    Construct your JSON output according to the following rules:

                    ---
```

```
                    **RULE A: If intent is `greeting_or_smalltalk`:**
                    - `response_type`: "greeting"
                   - `message`: A warm, friendly, and brief conversational reply.

                    ---
                    **RULE B: If intent is `off_topic_query`:**
                    - `response_type`: "off_topic"
                     - `message`: Politely state that you are a shoe specialist
and cannot help with non-footwear items.

                    ---
                    **RULE C: If intent is `product_query`:**
                    - `response_type`: "recommendation"
                    - Analyze the `PRODUCT CONTEXT`.
                     - If context is empty or irrelevant, provide a `reasoning`
message saying you couldn't find a match and an empty `recommendations` list.
                    - If context is relevant:
                      - `reasoning`: A helpful, detailed explanation in Markdown.
Start with a friendly opening paragraph. Then, for **EACH** product provided in
the context, create a numbered list item with the following nested format:
                            1. **[Full Product Title]**
                              - **Query Match**: Briefly explain why this product
is a good match for the user's request.
                                - **Key Features**: Based on the `features_clean`
column, write a detailed 2-4 sentence description summarizing the most prominent
features.
                          - `recommendations`: A list of JSON objects, each with
the exact `title` (lowercase) of **ALL** products from the `PRODUCT CONTEXT`.

                    ---
                    **PRODUCT CONTEXT (from database vector search):**
                    {context_for_llm}

                    **USER'S INPUT:**
                    - **Image Content Description**: {image_content_description}
                    - **User Text**: {normalized_prompt_text}

                    **JSON OUTPUT:**
                    """
```

Gambar 4.11 Kode Logika *Augmentation*

Berdasarkan kode program pada Gambar 4.11, *augmentation* tidak hanya menyuntikkan konteks produk dan teks pengguna, tetapi juga ringkasan deskriptif singkat

tentang gambar yang diperoleh dari model generatif melalui prompt "*Describe the primary object in this image in one or two words."* sehingga LLM memperoleh sinyal multimodal yang eksplisit di bagian "USER'S INPUT". Prompt utama didesain ulang dengan persona "*SoleMate*" dan memaksa model untuk mengklasifikasikan intent menjadi `greeting_or_smalltalk`, `product_query`, atau `off_topic`, lalu menghasilkan output JSON yang berbeda untuk setiap kondisi (greeting/off_topic: message; product_query: reasoning + daftar recommendations) sehingga mudah diurai dan ditayangkan ulang di UI

C. Logika *Generation*

Komponen final dari alur RAG adalah modul *generation*. Tahap ini melibatkan pengiriman *prompt* yang telah diperkaya ke *Large Language Model* (LLM) untuk diproses. Model kemudian menghasilkan respons dalam format terstruktur yang telah ditentukan, yang selanjutnya akan diurai (*parse*) dan disajikan kepada pengguna.

```
    rag_model = GenerativeModel(selected_model)
    rag_config = {"temperature": 0.1, "max_output_tokens": 8192, "top_p": 0.95}
    response = rag_model.generate_content(final_prompt,
generation_config=rag_config)
    response_text = response.text

    # Ekstraksi JSON
    match = re.search(r'\{.*\}', response_text, re.DOTALL)
    if not match:
        raise ValueError("No valid JSON format found in the AI's response.")
    parsed_response = json.loads(match.group(0))
    response_type = parsed_response.get("response_type")

    # Routing hasil
    if response_type == "recommendation":
        reasoning = parsed_response.get("reasoning", "Here are my recommendations.")
        recommendations_list = parsed_response.get("recommendations", [])
        # Dukung list of dicts atau list of strings
        if recommendations_list and isinstance(recommendations_list, dict):
            recommended_titles_lower = [rec.get("title") for rec in
recommendations_list if rec.get("title")]
        else:
            recommended_titles_lower = recommendations_list
        recommendations_df = pd.DataFrame()
        if not context_df.empty and recommended_titles_lower:
            recommendations_df = context_df[
```

```
                context_df['title'].str.lower().str.replace("’", "'",
regex=False).isin(recommended_titles_lower)
            ].copy()
        assistant_response_content = {"reasoning": reasoning, "recommendations_df":
recommendations_df}
    else:
        message = parsed_response.get("message", "I'm not sure how to respond to
that.")
        assistant_response_content = message
```

Gambar 4.12 Kode Logika *Generation*

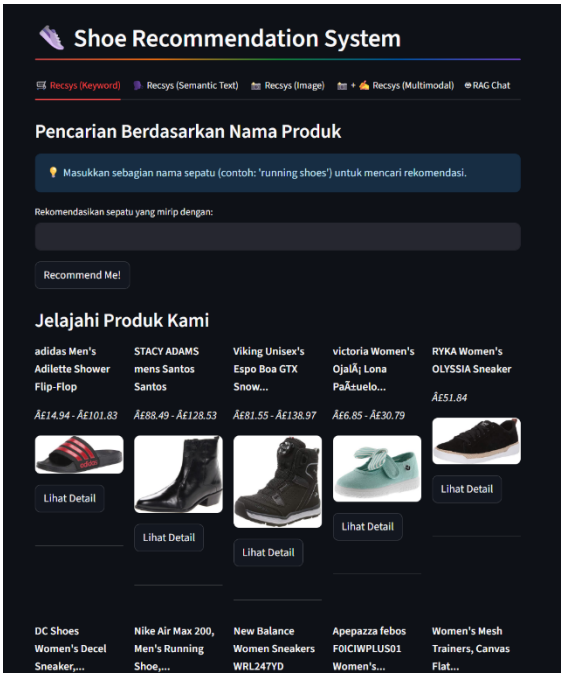
Berdasarkan kode program pada Gambar 4.12, model Gemini 2.5 Pro dipanggil dengan konfigurasi inferensi konservatif (temperature 0.1, top_p 0.95, max_output_tokens 8192) untuk konsistensi dan kepatuhan format JSON, kemudian respons dipotong via regex ke objek JSON yang valid untuk diurai secara deterministik. Alur pasca-generasi bercabang: jika `response_type` adalah “recommendation”, reasoning ditampilkan dan daftar titles dipetakan kembali ke `context_df` (yang sudah dinormalisasi) agar dapat dirender sebagai grid produk, sedangkan untuk “greeting” atau “off_topic”, field message ditampilkan sebagai balasan singkat tanpa grid.

4.2.4 Increment 4: Implementasi Antarmuka Pengguna

Increment terakhir difokuskan pada pembangunan antarmuka pengguna (UI) menggunakan Streamlit. UI dirancang dengan lima tab terpisah, di mana setiap tab dihubungkan dengan fungsi yang sesuai yang telah dikembangkan pada *increment* sebelumnya. Hal ini menciptakan sebuah aplikasi fungsional yang siap untuk diuji secara internal dan dievaluasi oleh pengguna.

A. Recsys (Keyword): Pencarian Berdasarkan Nama Produk

Tab pertama memungkinkan pengguna untuk melakukan pencarian produk sepatu berdasarkan kata kunci atau nama spesifik.

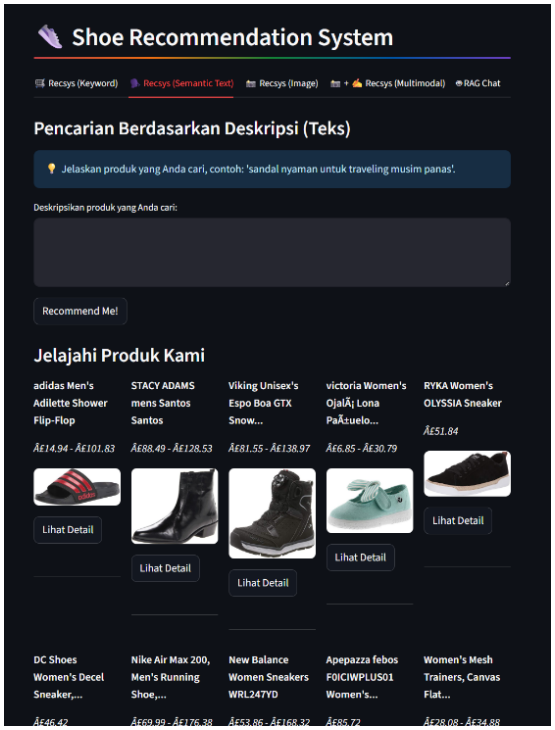


Gambar 4.13 Antarmuka Pencarian Kata Kunci.

Pada Gambar 4.13, pengguna memasukkan nama produk yang diinginkan pada kolom input yang tersedia. Sistem kemudian akan menjalankan pencarian leksikal pada database untuk menemukan produk dengan nama yang cocok dan menampilkan hasilnya dalam format grid. Jika tidak ada input, halaman akan menampilkan produk-produk secara acak untuk dijelajahi.

B. Recsys (Semantic Text): Pencarian Berdasarkan Deskripsi

Tab kedua menyediakan pencarian semantik, di mana pengguna dapat memasukkan deskripsi tekstual mengenai sepatu yang mereka cari.

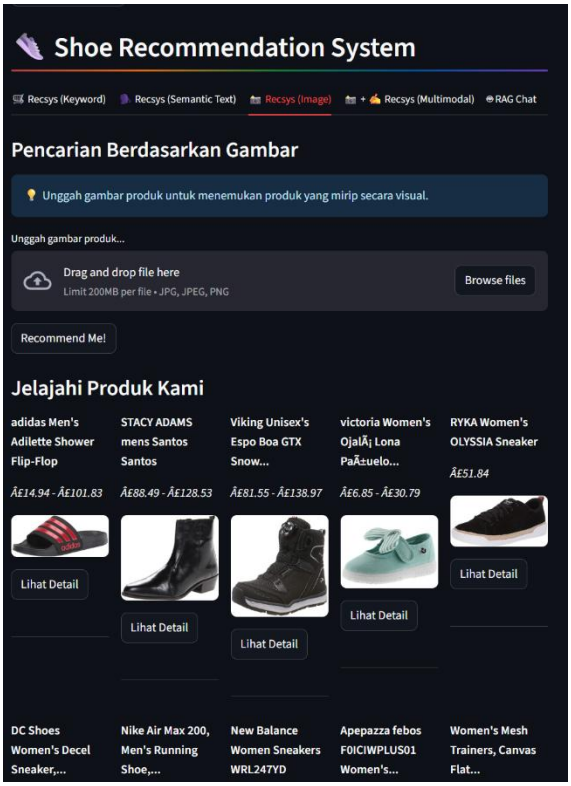


Gambar 4.14 Antarmuka Pencarian Semantik Teks

Pada Gambar 4.14, Pengguna dapat mengetikkan deskripsi konseptual, seperti "sandal nyaman untuk traveling musim panas". Sistem akan menganalisis makna dari deskripsi tersebut menggunakan model embedding untuk menemukan produk yang cocok secara kontekstual, tidak hanya berdasarkan kata kunci yang sama persis.

C. Recsys (Image): Pencarian Berdasarkan Gambar

Pada tab ini, pengguna dapat mengunggah gambar produk sepatu untuk menemukan produk yang mirip secara visual.

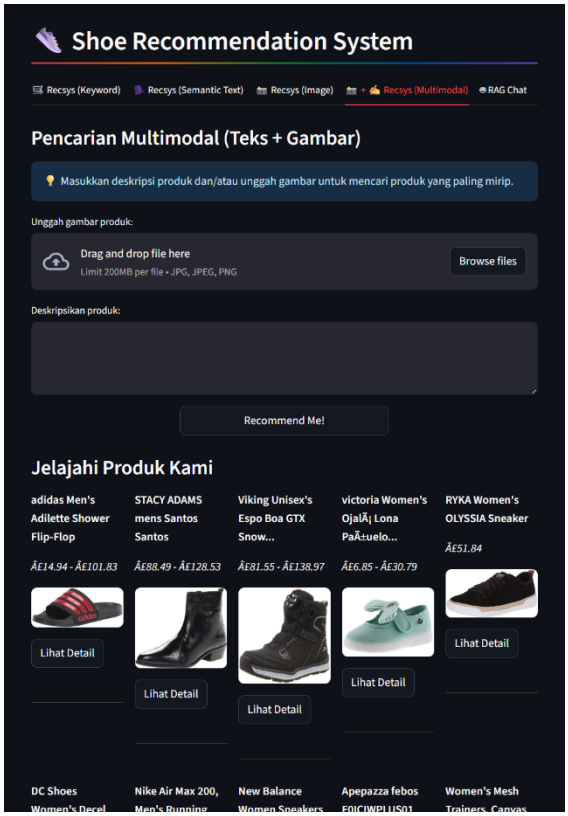


Gambar 4.15 Antarmuka Pencarian Semantik Gambar

Pada Gambar 4.15, sistem akan menggunakan model pengenalan gambar untuk menganalisis fitur visual dari sepatu dalam gambar yang diunggah. Setelah itu, sistem akan merekomendasikan produk lain dari database yang memiliki kemiripan visual paling tinggi.

D. Recsys (Multimodal): Pencarian Teks dan Gambar

Tab keempat mengimplementasikan pencarian multimodal yang menggabungkan input gambar dan teks untuk hasil yang lebih spesifik.

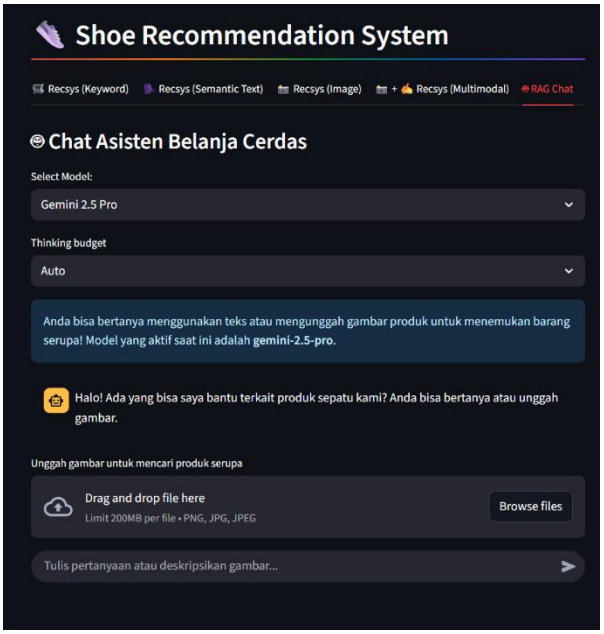


Gambar 4. 16 Antarmuka Pencarian Multimodal

Pada Gambar 4.16, pengguna dapat mengunggah gambar sepatu sambil memberikan deskripsi tambahan untuk memperjelas pencarian, misalnya mengunggah gambar sepatu lari dan menambahkan teks "untuk lari di medan berat". Sistem akan menggabungkan kedua informasi ini untuk memberikan hasil yang lebih akurat.

E. Recsys (Multimodal): Pencarian Teks dan Gambar

Tab terakhir adalah antarmuka percakapan (chat) dengan asisten belanja cerdas yang didukung oleh model bahasa canggih (Gemini).



Gambar 4.17 Antarmuka Asisten Chat RAG

Pada Gambar 4.17, pengguna dapat berinteraksi secara natural, mengajukan pertanyaan, meminta rekomendasi, atau bahkan mengunggah gambar produk dalam percakapan untuk menemukan barang serupa. Sistem RAG akan memberikan jawaban yang konsultatif dan informatif, sehingga menciptakan pengalaman belanja yang lebih interaktif dan personal.

4.3 Fase 4: Testing dan Hasil Penelitian

Setelah seluruh fungsionalitas sistem berhasil diimplementasikan pada fase sebelumnya, dilakukan tahap pengujian dan evaluasi menyeluruh terhadap sistem yang telah dikembangkan untuk menilai kinerja dan pengalaman pengguna secara komprehensif. Pengujian meliputi dua pendekatan utama: pengujian kuantitatif yang objektif dan evaluasi kualitatif yang mendalam.

Pengujian kuantitatif menggunakan serangkaian metrik evaluasi seperti *Precision@10*, *Recall@10*, *MAP@10*, dan *nDCG@10* untuk mengukur efektivitas pencarian produk melalui empat metode berbeda: Pencarian Kata Kunci, Pencarian Semantik Teks, Pencarian Multimodal, dan sistem RAG Multimodal. Khusus untuk sistem RAG, dilakukan juga evaluasi *Faithfulness* dan *Answer Relevance* guna menilai akurasi faktual dan kesesuaian jawaban yang dihasilkan oleh model generatif.

Sementara itu, evaluasi kualitatif dilakukan dengan melibatkan pengguna aktif e-commerce melalui metode *Think-Aloud Protocol* dan wawancara mendalam. Pendekatan ini bertujuan

menggali persepsi, kemudahan penggunaan, dan nilai tambah dari masing-masing fitur pencarian dari sudut pandang pengalaman pengguna.

4.3.1 Hasil Evaluasi Kuantitatif

Sebelum berlangsungnya evaluasi kualitatif yang melibatkan partisipan pengguna aktif e-commerce, seluruh fitur sistem telah melalui tahap uji coba kuantitatif menggunakan 60 kueri pengujian berbasis ground truth untuk mengukur efektivitas pencarian dibandingkan empat metode utama: Keyword Search, Semantic Search, Multimodal Search, dan RAG Multimodal. Evaluasi ini dilakukan dengan enam metrik yang telah terdefinisi secara teknis dalam metodologi penelitian, yaitu Precision@10 yang mengukur persentase produk relevan di 10 hasil teratas, Recall@10 yang menilai berapa persen dari seluruh produk relevan yang berhasil ditemukan di 10 teratas, *Mean Average Precision* (MAP@10) mempertimbangkan urutan hasil dan presisi kumulatif, dan nDCG@10 (*Normalized Discounted Cumulative Gain*), metrik komprehensif yang menilai relevansi dan urutan hasil secara keseluruhan.

Khusus untuk fitur RAG Multimodal, digunakan 20 kueri pengujian berbasis ground truth dengan dua metrik tambahan digunakan yaitu *Faithfulness*, yang mengukur tingkat kesetiaan respons chatbot terhadap fakta produk yang ada dan mendeteksi adanya halusinasi. Serta *Answer Relevance*, yang menilai sejauh mana respons memberikan jawaban yang benar-benar relevan sesuai dengan inti pertanyaan pengguna. Penilaian dengan keenam metrik ini dilakukan secara sistematis untuk setiap mode pencarian demi mendapatkan baseline yang obyektif, sehingga tahap evaluasi berikutnya terhadap pengalaman pengguna dapat dilakukan dengan kerangka pembanding yang jelas dan terukur.

A. Hasil Pengujian *Precision@10*

Pengujian *Precision@10* dilakukan untuk mengetahui seberapa akurat hasil rekomendasi 10 produk teratas yaitu berapa persen produk relevan yang berhasil ditampilkan oleh tiap metode pencarian pada sistem e-commerce penelitian ini

Tabel 4. 2 Hasil Pengujian *Precision@10*

Tipe Kueri	**Keyword Search**	**Semantic Search**	**Multimodal Search**	**RAG Multimodal**
Keyword Queries	0.95	0.88	0.88	0.91
Semantic Queries	0.24	0.87	0.87	0.85

Multimodal Queries	0.05	0.13	0.68	0.84
RAG Multimodal Queries	0.03	0.16	0.71	0.86
Skor Rata-rata	**0.32**	**0.51**	**0.78**	**0.86**

Hasil pada Tabel 4. 2 menunjukkan bahwa skor *Precision@10* paling tinggi diperoleh oleh fitur RAG Multimodal dengan rata-rata 0.86, diikuti oleh Multimodal Search (0.78), Semantic Search (0.51), dan Keyword Search (0.32). Pada kueri berbasis kata kunci, metode keyword search sangat unggul (0.95), namun skor turun drastis pada tipe kueri lain, menandakan kelemahan untuk kueri yang lebih konseptual atau multimodal. Fitur RAG Multimodal konsisten memberikan produk relevan terbaik dalam 10 rekomendasi teratas, terutama pada skenario kompleks dan ambigus, membuktikan keunggulan pendekatan multimodal dan percakapan berbasis AI di e-commerce.

B. Hasil Pengujian *Recall@10*

Pengujian *Recall@10* dilakukan untuk mengukur seberapa lengkap hasil rekomendasi dalam menemukan produk relevan di antara 10 produk teratas. Metrik ini menunjukkan persentase produk relevan yang berhasil ditemukan oleh tiap metode pencarian dalam sistem e-commerce penelitian ini.

Tabel 4. 3 Hasil Pengujian *Recall@10*

Tipe Kueri	**Keyword Search**	**Semantic Search**	**Multimodal Search**	**RAG Multimodal**
Keyword Queries	0.78	0.71	0.71	0.74
Semantic Queries	0.18	0.70	0.70	0.68
Multimodal Queries	0.04	0.11	0.55	0.69
RAG Multimodal Queries	0.02	0.13	0.58	0.71
Skor Rata-rata	**0.26**	**0.41**	**0.63**	**0.70**

Hasil pada Tabel 4. 3 menunjukkan bahwa fitur RAG Multimodal memiliki skor *Recall@10* tertinggi dengan rata-rata 0.70, diikuti oleh Multimodal Search (0.63), Semantic Search (0.41), dan Keyword Search (0.26). Pada kueri berbasis kata kunci,

metode keyword search menunjukkan performa recall yang baik (0.78), namun mengalami penurunan signifikan pada jenis kueri yang lebih kompleks dan multimodal. Fitur RAG Multimodal konsisten lebih efektif dalam menemukan produk relevan di 10 hasil teratas, yang menggarisbawahi keunggulan sistem ini dalam skenario pencarian yang membutuhkan pemahaman konteks luas.

C. Hasil Pengujian *MAP@10*

Pengujian *MAP@10* bertujuan untuk mengukur presisi dengan memperhatikan urutan produk relevan dalam daftar 10 rekomendasi teratas. Metrik ini memberi bobot lebih pada produk relevan yang muncul lebih awal, menggambarkan seberapa cepat pengguna dapat menemukan produk yang dicari.

Tabel 4. 4 Hasil Pengujian *MAP@10*

Tipe Kueri	**Keyword Search**	**Semantic Search**	**Multimodal Search**	**RAG Multimodal**
Keyword Queries	0.93	0.85	0.85	0.88
Semantic Queries	0.19	0.82	0.82	0.80
Multimodal Queries	0.03	0.10	0.64	0.81
RAG Multimodal Queries	0.02	0.12	0.67	0.83
Skor Rata-rata	**0.29**	**0.47**	**0.74**	**0.83**

Dari Tabel 4. 4, terlihat bahwa RAG Multimodal memperoleh skor *MAP@10* tertinggi dengan rata-rata 0.83, menunjukkan kemampuannya untuk menempatkan produk relevan pada posisi atas dalam hasil pencarian. Multimodal Search menempati posisi kedua (0.74), diikuti oleh Semantic Search (0.47), dan Keyword Search (0.29). Pada kueri kata kunci, Keyword Search memiliki kinerja sangat baik (0.93), tetapi kinerjanya menurun drastis pada kueri lainnya. Ini menekankan efektivitas RAG Multimodal dalam menyediakan hasil yang tidak hanya relevan tetapi juga terurut dengan baik sesuai konteks pencarian.

D. Hasil Pengujian *nDCG@10*

Pengujian *nDCG@10* merupakan evaluasi paling komprehensif yang mengukur presisi, urutan, dan tingkat relevansi produk dalam 10 hasil teratas. Metrik ini menilai kualitas

peringkat hasil pencarian dengan memberikan bobot lebih pada produk relevan di posisi atas.

Tabel 4. 5 Hasil Pengujian *nDCG@10*

Tipe Kueri	**Keyword Search**	**Semantic Search**	**Multimodal Search**	**RAG Multimodal**
Keyword Queries	0.94	0.87	0.87	0.90
Semantic Queries	0.22	0.85	0.85	0.83
Multimodal Queries	0.04	0.12	0.66	0.84
RAG Multimodal Queries	0.03	0.15	0.69	0.86
Skor Rata-rata	**0.31**	**0.50**	**0.77**	**0.86**

Tabel 4. 5 menunjukkan bahwa fitur RAG Multimodal unggul dengan skor *nDCG@10* tertinggi rata-rata 0.86, diikuti oleh Multimodal Search (0.77), Semantic Search (0.50), dan Keyword Search (0.31). Pada kueri kata kunci, Keyword Search sangat baik (0.94), tetapi menurun pada jenis kueri lain, mencerminkan keterbatasannya dalam menangani kebutuhan pencarian yang lebih kompleks. Hasil ini menegaskan posisi fitur RAG Multimodal sebagai solusi pencarian yang paling efektif dalam memberikan hasil relevan dengan urutan yang optimal, khususnya dalam konteks pencarian e-commerce yang multimodal dan percakapan.

E. Hasil Pengujian *Faithfulnes* dan *Answer Relevance*

Pengujian *Faithfulness* dan *Answer Relevance* merupakan evaluasi penting untuk menilai kualitas jawaban yang diberikan oleh sistem RAG Multimodal. Metrik *Faithfulness* mengukur sejauh mana jawaban yang dihasilkan tetap setia pada fakta dan data yang tersedia, menghindari penyajian informasi yang salah atau halusinasi. Sementara itu, *Answer Relevance* menilai kemampuan sistem dalam memberikan respons yang tepat dan sesuai dengan maksud dan konteks pertanyaan pengguna.

Tabel 4. 6 Hasil Pengujian *Faithfulnes* dan *Answer Relevance*

Metrik	**Skor Rata-rata (0–1)**	**Interpretasi**

Faithfulness	0.90	Jawaban chatbot sangat setia pada fakta (fitur, merek, dll.) yang ada di dalam konteks produk. Sistem hampir tidak pernah "berhalusinasi".
Answer Relevance	0.80	Sebagian besar jawaban sangat relevan dan menjawab inti pertanyaan pengguna, menunjukkan pemahaman yang kuat terhadap niat pengguna.

Hasil evaluasi pada Tabel 4. 6 menunjukkan bahwa sistem RAG Multimodal memberikan jawaban yang akurat dan faktual dengan tingkat *faithfulness* tinggi 0.90. Selain itu, jawaban yang diberikan juga sangat relevan dengan pertanyaan pengguna dengan skor 0.80, menandakan model mampu mengerti konteks dan kebutuhan pengguna dengan baik.

4.3.2 Hasil Evaluasi Kualitatif

Bagian ini menyajikan temuan dari evaluasi kualitatif yang diperoleh melalui analisis data dari sesi *Think-Aloud Protocol* dan wawancara mendalam dengan partisipan. Data yang terkumpul dianalisis menggunakan metode analisis tematik untuk mengidentifikasi pola, wawasan, dan tema-tema kunci yang muncul dari pengalaman pengguna.

Untuk menyajikan temuan secara sistematis, hasil analisis dirangkum dalam tabel di bawah ini. Tabel ini memetakan setiap tema utama yang teridentifikasi dengan fitur pencarian yang relevan, temuan kunci dari perspektif pengguna, kutipan pendukung yang representatif, serta implikasi dari temuan tersebut.

Tabel 4. 7 Hasil Pengujian Kualitatif

Tema Utama	**Fitur Pencarian Terkait**	**Temuan Kunci dari Pengguna**	**Kutipan Pendukung Representatif**	**Implikasi**
Keterbatasan Pencarian Leksikal	Pencarian Kata Kunci	Gagal memahami niat pencarian yang abstrak/konseptual. Menimbulkan frustrasi karena tidak mengenali sinonim atau konteks. Hanya efektif jika pengguna mengetahui	*"Saya ketik 'sepatu buat lari jauh', malah tidak keluar apa apa. Sistemnya tidak mengerti maksud saya 'lari jauh' itu butuh bantalan lebih dan ini hanya cocok*	Pencarian berbasis kata kunci tidak memadai untuk kebutuhan belanja modern yang bersifat eksploratif dan menciptakan

		nama atau merek produk spesifik.	*untuk sneakerhead saja yang tau merk spesifik sepatu."*	friksi yang signifikan bagi pengguna.
Lompatan Intuitif via Pencarian Semantik	Pencarian Semantik (Teks & Gambar)	Sistem “mengerti” niat di luar kata literal. Pencarian gambar sangat membantu untuk inspirasi visual. Hasil relevan, tetapi pengguna tetap perlu memfilter dan membandingkan.	*"Wow, ini keren. Saya unggah gambar sepatu boots coklat, dan yang muncul benar-benar boots lain dengan model yang mirip, bukan cuma sepatu coklat biasa."*	Pencarian semantik merupakan peningkatan signifikan, namun masih menyisakan beban kognitif bagi pengguna untuk menyaring hasil dan membuat keputusan akhir.
Puncak Pengalaman: Dialog Konsultatif	Asisten Chat RAG	Fitur paling disukai dan bermanfaat. Mengubah interaksi dari “mencari” menjadi “berkonsultasi”. Mampu menangani kueri ambigu/kompleks dengan rekomendasi yang mempertimbangkan batasan. Justifikasi (reasoning) membangun kepercayaan pengguna.	*"Yang chat ini paling bagus. Rasanya seperti ngobrol sama teman yang jago sepatu. Dia tidak cuma kasih daftar, tapi juga kasih tahu kenapa sepatu A lebih bagus buat lari maraton daripada sepatu B."*	Interaksi percakapan dengan justifikasi adalah kunci untuk pengalaman pengguna yang superior. Mengurangi beban kognitif dan meningkatkan kepercayaan pengguna secara drastis.
Nilai Tambah Signifikan RAG	Asisten Chat RAG	Memberikan nilai tambah terbesar saat niat pengguna tidak jelas, kebutuhan kompleks, atau tahap eksplorasi.• Superior untuk penemuan produk baru, sedangkan fitur lain hanya “cukup” untuk pencarian spesifik.	*"Jujur, saya sering buka aplikasi cuma buat 'window shopping' pakai chat ini. Kadang saya cuma bilang 'lagi cari kado buat teman yang suka hiking dan tentunya bisa dipakai lama', dan dia bisa kasih ide-ide yang tidak pernah saya pikirkan*	RAG bukan hanya fitur pencarian lebih baik, tetapi alat penemuan yang efektif, yang secara fundamental mengubah cara pengguna berinteraksi

			sebelumnya. Kalau pakai search bar biasa, saya pasti bingung mau ketik apa."	dengan platform e-commerce.

Analisis data kualitatif dari Tabel 4. 7 menunjukkan sebuah narasi progresif yang jelas dalam pengalaman pengguna, bergerak dari interaksi pencarian yang kaku menuju dialog yang konsultatif. Empat tema utama yang muncul secara kolektif menjawab rumusan masalah penelitian, menguraikan perbandingan pengalaman pengguna dan nilai tambah dari setiap metode pencarian.

A. Tema 1: Keterbatasan Pencarian Leksikal pada Niat Abstrak

Temuan yang konsisten dari seluruh partisipan adalah munculnya frustrasi ketika menggunakan fitur Pencarian Kata Kunci untuk tugas-tugas yang bersifat eksploratif atau konseptual. Fitur ini dinilai paling tidak efektif karena kegagalan fundamentalnya dalam memahami sinonim, konteks, atau niat abstrak di balik kueri. Seperti yang diungkapkan oleh seorang partisipan, pencarian untuk "sepatu buat lari jauh" tidak membuahkan hasil karena sistem tidak dapat menginterpretasikan konsep "lari jauh" sebagai kebutuhan akan fitur spesifik seperti "bantalan lebih". Hal ini menegaskan bahwa pencarian leksikal hanya memadai untuk skenario pencarian produk spesifik dan menciptakan friksi yang signifikan untuk kebutuhan belanja modern yang lebih eksploratif.

B. Tema 2: Lompatan Intuitif melalui Pencarian Semantik

Partisipan menunjukkan reaksi positif yang signifikan saat beralih ke fitur Pencarian Semantik, baik melalui teks maupun gambar. Ada perasaan kolektif bahwa sistem mulai "mengerti" niat mereka di luar pencocokan kata kunci literal. Fitur pencarian gambar, khususnya, sangat dipuji karena kemampuannya menemukan produk yang serupa secara visual dari sebuah inspirasi, seperti menemukan sepatu boots lain dengan model yang mirip. Meskipun demikian, teridentifikasi batasan bahwa meskipun hasilnya relevan, partisipan masih merasa perlu memfilter dan membandingkan sendiri dari daftar hasil yang panjang. Ini menunjukkan bahwa pencarian semantik berhasil mengurangi friksi, namun masih menyisakan beban kognitif bagi pengguna dalam pengambilan keputusan akhir.

C. Tema 3: Puncak Pengalaman: Dialog Konsultatif dengan RAG

Secara universal, seluruh partisipan (5 dari 5) secara eksplisit menyatakan bahwa fitur Asisten Chat RAG adalah yang paling mereka sukai dan paling bermanfaat. Keberhasilan fitur ini terletak pada kemampuannya mengubah paradigma interaksi dari sekadar "mencari" menjadi "berkonsultasi". Fitur RAG unggul dalam menangani skenario ambigu dan kompleks, seperti mencari hadiah dengan beberapa batasan. Namun, aspek yang paling krusial adalah adanya justifikasi. Kemampuan sistem untuk menjelaskan *mengapa* sebuah produk direkomendasikan seperti yang diungkapkan partisipan, "seperti ngobrol sama teman yang jago sepatu" terbukti menjadi kunci dalam membangun kepercayaan dan secara drastis mengurangi beban kognitif pengguna.

D. Tema 4: RAG sebagai Solusi Nilai Tambah Signifikan

Ketika ditanya mengenai skenario di mana RAG memberikan nilai tambah paling signifikan, partisipan secara konsisten menunjuk pada situasi di mana niat mereka tidak jelas, kebutuhannya kompleks, atau saat mereka berada dalam tahap eksplorasi. Untuk pencarian yang sangat spesifik, fitur *baseline* dianggap "cukup", namun untuk penemuan produk baru, RAG dianggap superior. Hal ini memposisikan RAG bukan hanya sebagai fitur pencarian yang lebih baik, tetapi sebagai sebuah alat penemuan yang efektif. Fitur ini secara fundamental mengubah cara pengguna berinteraksi dengan platform, dari sekadar alat transaksi menjadi mitra belanja yang cerdas dan konsultatif.

BAB V
KESIMPULAN DAN SARAN

Bab ini menyajikan rangkuman dari keseluruhan proses dan temuan penelitian, serta merumuskan implikasi dan saran untuk pengembangan di masa depan. Bagian kesimpulan akan secara langsung menjawab rumusan masalah yang telah diajukan, sementara bagian saran akan mengidentifikasi potensi-potensi penelitian lanjutan yang dapat dibangun di atas fondasi dari studi ini.

5.1 Kesimpulan

Penelitian ini menegaskan bahwa sistem RAG Multimodal berhasil dirancang dan diimplementasikan secara end-to-end pada domain e-commerce dengan menghadirkan empat kapabilitas utama, yaitu pencarian kata kunci, pencarian semantik berbasis teks, pencarian semantik berbasis gambar, dan asisten percakapan berbasis RAG. Sistem ini dibangun di atas arsitektur yang mengintegrasikan Streamlit sebagai antarmuka pengguna, SigLIP sebagai model embedding teks dan gambar berdimensi 1152, PostgreSQL+pgvector dengan indeks HNSW untuk mendukung pencarian vektor yang efisien, serta Gemini 2.5 Pro sebagai generator percakapan. Pada tahap retrieval, strategi *late fusion* digunakan untuk menggabungkan konteks yang relevan, sementara alur RAG diterapkan untuk menjamin grounding informasi yang kuat sebelum respons konsultatif dihasilkan.

Hasil evaluasi kuantitatif menggunakan 60 kueri *ground truth* menunjukkan bahwa RAG Multimodal memberikan efektivitas retrieval yang lebih tinggi dibandingkan seluruh mode pencarian lainnya, dengan capaian rata-rata *Precision@10* sebesar 0,86, *Recall@10* sebesar 0,70, *MAP@10* sebesar 0,83, dan *nDCG@10* sebesar 0,86. Pencapaian ini terbukti lebih unggul dibandingkan Multimodal Search, Semantic Search, maupun Keyword Search, meskipun keunggulan Keyword Search masih terlihat pada kueri leksikal yang sangat spesifik dan bersifat eksplisit. Namun demikian, performa metode tersebut menurun secara signifikan ketika diterapkan pada kueri semantik maupun multimodal.

Pada evaluasi kualitas jawaban menggunakan 20 kueri uji, asisten percakapan RAG Multimodal mencapai skor *Faithfulness* sebesar 0,90 dan *Answer Relevance* sebesar 0,80, yang menunjukkan bahwa jawaban yang dihasilkan tidak hanya setia terhadap konteks produk, tetapi juga relevan dengan maksud pertanyaan. Kondisi ini berimplikasi pada berkurangnya risiko halusinasi sekaligus meningkatnya utilitas praktis sistem. Lebih jauh, hasil evaluasi kualitatif melalui *task scenario* dan metode *Think-Aloud* mengindikasikan bahwa asisten percakapan

RAG menjadi fitur yang paling disukai partisipan. Hal ini disebabkan oleh transformasi pola interaksi yang tidak lagi terbatas pada aktivitas pencarian, tetapi berkembang menjadi pengalaman konsultasi yang menyertakan justifikasi dapat diaudit serta menurunkan beban kognitif dalam proses penyaringan dan perbandingan hasil, khususnya pada kebutuhan yang kompleks, ambigu, atau eksploratif.

Selain itu, pencarian semantik berbasis teks dan gambar dipandang memberikan lompatan intuitif dibandingkan pencocokan leksikal karena mampu menangkap maksud yang melampaui representasi literal kata. Pencarian berbasis gambar, khususnya, diapresiasi karena mendukung kebutuhan inspirasi visual, meskipun pengguna tetap dituntut untuk melakukan penyaringan manual terhadap daftar hasil yang panjang.

Berdasarkan keseluruhan temuan tersebut, dapat disimpulkan bahwa implementasi teknis berhasil diwujudkan sesuai rancangan arsitektur dan kebutuhan fungsional, efektivitas retrieval serta kualitas jawaban sistem RAG terbukti unggul melalui metrik objektif maupun evaluasi jawaban, dan preferensi pengguna secara konsisten menempatkan RAG sebagai fitur dengan nilai tambah tertinggi pada skenario nonspesifik yang menuntut penalaran multimodal, justifikasi, dan dialog adaptif. Dengan demikian, penelitian ini menegaskan bahwa RAG Multimodal merupakan pendekatan yang lebih relevan bagi e-commerce modern karena mampu mengatasi keterbatasan metode pencarian tradisional sekaligus menghadirkan pengalaman belanja yang bersifat konsultatif, terjustifikasi, dan berpusat pada pengguna.

5.2 Saran

Berdasarkan temuan dan batasan dalam penelitian ini, beberapa saran untuk pengembangan dan penelitian di masa depan dapat diidentifikasi:

A. **Ekspansi ke Domain Data dan Modalitas yang Lebih Beragam:**

1 Aplikasi pada Domain Produk yang Berbeda**:**

Penelitian ini berfokus pada produk *footwear*. Studi lanjutan dapat menguji dan mengadaptasi arsitektur sistem ini pada domain produk lain yang memiliki karakteristik data berbeda, seperti ranah otomotif (dengan detail teknis yang kompleks) atau ranah olahraga (dengan atribut performa yang spesifik), untuk mengevaluasi generalisasi dan tantangan unik di setiap domain.

2 Integrasi Bentuk Multimodal yang Lebih Beragam

Sistem saat ini menangani input teks dan gambar. Penelitian di masa depan dapat mengeksplorasi penambahan modalitas input lain seperti audio (misalnya,

pencarian produk menggunakan perintah suara) atau video (misalnya, menemukan produk dari klip video ulasan), yang akan menuntut arsitektur *retrieval* yang lebih canggih.

B. **Penggunaan Arsitektur dan Teknologi Skala Produksi:**

Aplikasi yang dibangun menggunakan Streamlit sangat ideal untuk penelitian, namun untuk skala produksi, arsitektur yang lebih *proper* perlu dipertimbangkan. Ini bisa melibatkan penggunaan *framework backend* yang lebih robust (seperti FastAPI atau Django), *database* vektor terdistribusi (seperti Pinecone atau Weaviate dalam mode klaster), dan implementasi mekanisme *caching* untuk mengurangi latensi.

DAFTAR PUSTAKA

Abootorabi, M. M., Zobeiri, A., Dehghani, M., Mohammadkhani, M., Mohammadi, B., Ghahroodi, O., Baghshah, M. S., & Asgari, E. (2025). *Ask in Any Modality: A Comprehensive Survey on Multimodal Retrieval-Augmented Generation*. https://arxiv.org/pdf/2502.08826

Bach, T. A., Khan, A., Hallock, H., Beltrão, G., & Sousa, S. (2024). A Systematic Literature Review of User Trust in AI-Enabled Systems: An HCI Perspective. *International Journal of Human-Computer Interaction*, *40*(5), 1251–1266. https://doi.org/10.1080/10447318.2022.2138826;WGROUP:STRING:PUBLICATION

Bai, J., Chen, M., Xiaosheng, J. L., Daniel, M., Xu, Y., Akolakis, C., Allen, T., Atkin, D., Bandiera, O., Banerjee, A., Bergquist, L., Chor, D., Donaldson, D., Duflo, E., Faber, B., Hanson, G., Hsieh, C.-T., Jia, P., Khan-Delwal, A., … Xu, D. Y. (2020). *Search and Information Frictions on Global E-Commerce Platforms: Evidence from AliExpress*. https://doi.org/10.3386/W28100

Daniel, J., & Martin, J. H. (2021). *Speech and Language Processing*.

Erlina F. Santika. (2024). *Tingkat Penetrasi Internet Indonesia Capai 79,5% per 2024*. https://databoks.katadata.co.id/teknologi-telekomunikasi/statistik/e6f9d69e252de32/tingkat-penetrasi-internet-indonesia-capai-795-per-2024

Erukude, S. T., Veluru, S. R., & Marella, V. C. (2025). Multimodal Deep Learning: A Survey of Models, Fusion Strategies, Applications, and Research Challenges. *International Journal of Computer Applications*, *187*(19), 1–7. https://doi.org/10.5120/IJCA2025925264

Gan, A., Yu, H., Zhang, K., Liu, Q., Yan, W., Huang, Z., Tong, S., Chen, E., & Hu, G. (2025). Retrieval Augmented Generation Evaluation in the Era of Large Language Models: A Comprehensive Survey. *Front. Comput. Sci*, *2025*(0), 1–18. https://doi.org/10.1007/sxxxxx-yyy-zzzz-1

Gemini Team. (2025). *Gemini 2.5: Pushing the Frontier with Advanced Reasoning, Multimodality, Long Context, and Next Generation Agentic Capabilities*.

Google, T. and B. & C. (2025). *e-Conomy SEA 2020*. https://www.temasek.com.sg/content/dam/temasek-corporate/news-and-views/resources/reports/e-conomy-sea-2020-report.pdf

Gupta, S., Ranjan, R., & Singh, S. N. (2024). *A Comprehensive Survey of Retrieval-Augmented Generation (RAG): Evolution, Current Landscape and Future Directions*. https://arxiv.org/pdf/2410.12837

Hendriksen, M. (2022). Multimodal Retrieval in E-Commerce: From Categories to Images, Text, and Back. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* , *13186 LNCS*, 505–512. https://doi.org/10.1007/978-3-030-99739-7_62

Hu, Y. (2024). Design and Research of Cross-Border E-Commerce Short Video Recommendation System Based on Multi-Modal Fusion Transformer Model. *International Journal of Advanced Computer Science and Applications*, *15*(8), 410–419. https://doi.org/10.14569/IJACSA.2024.0150840

Jannach, D. (2021). A Survey on Conversational Recommender Systems. *ACM Comput. Surv*, *54*.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W. T., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Advances in Neural Information Processing Systems*, *2020-December*. https://arxiv.org/pdf/2005.11401

Li, Z., Wang, B., Prince, O., Le Texier, T., & Addai-Dansoh, S. (2025). *EXPLORING PERCEPTION OF PLATFORM LANGUAGE ON CONSUMER DECISION MAKING: A COGNITIVE LOAD PERSPECTIVE* *.

Liang, M., & Niu, T. (2022). Research on Text Classification Techniques Based on Improved TF-IDF Algorithm and LSTM Inputs. *Procedia Computer Science*, *208*, 460–470. https://doi.org/10.1016/J.PROCS.2022.10.064

Liang, W., Zhang, Y., Kwon, Y., Yeung, S., & Zou, J. (2022). Mind the Gap: Understanding the Modality Gap in Multi-modal Contrastive Representation Learning. *Advances in Neural Information Processing Systems*, *35*. https://arxiv.org/pdf/2203.02053

Mortaheb, M., Amir, M. A. (, Khojastepour,), Chakradhar, S. T., & Ulukus, S. (2025). *RAG-Check: Evaluating Multimodal Retrieval Augmented Generation Performance*. https://arxiv.org/pdf/2501.03995

Mouhiha, M., Oualhaj, O. A., & Mabrouk, A. (2024). Combining Collaborative Filtering and Content Based Filtering for Recommendation Systems. *Proceedings - 11th International Conference on Wireless Networks and Mobile Communications, WINCOM 2024*. https://doi.org/10.1109/WINCOM62286.2024.10658169

Munyampirwa, B., Lakshman, V., & Coleman, B. (2025). *Down with the Hierarchy: The "H" in HNSW Stands for 'Hubs.'* https://github.com/BlaiseMuhirwa/flatnav

Nurdin, A. A., & Abidin, Z. (2023). The Influence of Recommendation System Quality on E-commerce Customer Loyalty with Cognition Affective Behavior Theory. *Journal of Advances in Information Systems and Technology*, *5*(1), 1–11. https://doi.org/10.15294/JAIST.V5I1.65910

Pratiwi, K. D. (2022). E-COMMERCE AND ECONOMIC GROWTH IN INDONESIA: ANALYSIS OF PANEL DATA REGRESSION. *GEMA PUBLICA*, *7*(1), 171–186. https://doi.org/10.14710/GP.7.1.2022.171-186

Rokon, O. F., Simion, A., Du, W., Wen, M., Yao, H., & Lee, K.-C. (2024). *Enhancement of E-commerce Sponsored Search Relevancy with LLM*.

Sanguigni, F., Morelli, D., Cornia, M., & Cucchiara, R. (2025). *Fashion-RAG: Multimodal Fashion Image Editing via Retrieval-Augmented Generation*. https://arxiv.org/pdf/2504.14011

Sohn, J., Park, Y., Yoon, C., Park, S., Hwang, H., Sung, M., Kim, H., & Kang, J. (2024). *Rationale-Guided Retrieval Augmented Generation for Medical Question Answering*. 12739–12753. https://doi.org/10.18653/v1/2025.naacl-long.635

Uta, M., Felfernig, A., Le, V. M., Tran, T. N. T., Garber, D., Lubos, S., & Burgstaller, T. (2024). Knowledge-based recommender systems: overview and research directions. *Frontiers in Big Data*, *7*, 1304439. https://doi.org/10.3389/FDATA.2024.1304439/BIBTEX

Yu, H., Gan, A., Zhang, K., Tong, S., Liu, Q., & Liu, Z. (2024). Evaluation of Retrieval-Augmented Generation: A Survey. *Communications in Computer and Information Science*, *2301*, 102–120. https://doi.org/10.1007/978-981-96-1024-2_8

Zhai, X., Mustafa, B., Kolesnikov, A., & Beyer, L. (2023). Sigmoid Loss for Language Image Pre-Training. *Proceedings of the IEEE International Conference on Computer Vision*, 11941–11952. https://doi.org/10.1109/ICCV51070.2023.01100

Es, S., James, J., Espinosa-Anke, L., & Schockaert, S. (2023). Ragas: Automated Evaluation of Retrieval Augmented Generation. *EACL 2024 -*

18th Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of System Demonstrations, 150–158. https://doi.org/10.18653/v1/2024.eacl-demo.16

Fan, M., & Zhu, L. (2022). Think-Aloud Verbalizations for Identifying User Experience Problems: Effects of Language Proficiency with Chinese Non-Native English Speakers. *ACM International Conference Proceeding Series*, 22–32. https://doi.org/10.1145/3490355.3490358

Jadon, A., & Patil, A. (2023). A Comprehensive Survey of Evaluation Techniques for Recommendation Systems. *Communications in Computer and Information Science*, *2185 CCIS*, 281–304. https://doi.org/10.1007/978-3-031-71484-9_25

LAMPIRAN