PERSEPSI RELEVANSI PADA PENCARIAN SEMANTIK *E-COMMERCE*: ANALISIS PERBANDINGAN KUALITATIF *MODEL EMBEDDING* OPENAI DAN VOYAGEAI MELALUI BLIND *A/B TESTING*

Disusun Oleh:

N a m a : Adyuta Indra Adyatma
NIM : 21523226

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

HALAMAN PENGESAHAN DOSEN PEMBIMBING

PERSEPSI RELEVANSI PADA PENCARIAN SEMANTIK *E-COMMERCE*: ANALISIS PERBANDINGAN KUALITATIF MODEL EMBEDDING OPENAI DAN VOYAGEAI MELALUI BLIND A/B TESTING

TUGAS AKHIR

Disusun Oleh:

N a m a : Adyuta Indra Adyatma

NIM : 21523226

Yogyakarta, 28 Agustus 2025

Pembimbing,

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

HALAMAN PENGESAHAN DOSEN PENGUJI

PERSEPSI RELEVANSI PADA PENCARIAN SEMANTIK *E-COMMERCE*: ANALISIS PERBANDINGAN KUALITATIF MODEL EMBEDDING OPENAI DAN VOYAGEAI MELALUI BLIND A/B TESTING

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 Agustus 2025

Tim Penguji

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

Anggota 1

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

Anggota 2

Rahadian Kurniawan, S.Kom., M.Kom.

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 : Adyuta Indra Adyatma

NIM. : 21523226

Tugas akhir dengan judul:

PERSEPSI RELEVANSI PADA PENCARIAN SEMANTIK *E- COMMERCE*: ANALISIS PERBANDINGAN KUALITATIF MODEL EMBEDDING OPENAI DAN VOYAGEAI MELALUI BLIND A/B TESTING

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, 21 Juli 2025

(Adyuta Indra Adyatma)

HALAMAN PERSEMBAHAN

Dengan segenap rasa syukur, karya sederhana ini saya persembahkan kepada:

Allah Subhanahu wa Ta'ala, atas segala Karunia, petunjuk, dan kekuatan yang tak pernah putus.

Kedua orang tua saya yang terhebat,

Bapak Daris Munarto dan Mama Inayatul Bundariyah,

atas setiap tetes keringat, doa yang tak terhingga, dan cinta tanpa syarat.

Ini adalah bakti kecilku untuk kalian.

Kedua kakak tersayang,

Priliese Indra Devi dan Robby Indra Lukman,

yang selalu menjadi inspirasi dan pendukung utama.

Sahabat-sahabat seperjuangan,

Rekan-rekan di KAMADINO (Rizal, Pandu, Ravel, Haikal, Michael, Javier),

Hilqudz Dzikri, Arya Andhika, Ihwan, Ridho, Rifky, Galuh, Alex, Aldi

terima kasih untuk setiap tawa, diskusi, dan semangat yang membuat perjalanan ini terasa lebih ringan.

Lebih dari sekadar pencapaian pribadi, karya ini adalah wujud kecil dari bakti dan cinta saya kepada kedua orang tua.

HALAMAN MOTO

"At some point, everything's gonna go south and you're going to say, this is it. This is how I end. Now, you can either accept that, or you can get to work. That's all it is. You just begin. You do the math. You solve one problem and you solve the next one and then the next. And if you solve enough problems, you get to come home."

— Matt Damon, The Martian

KATA PENGANTAR

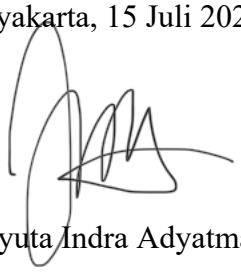
Segala puji bagi Allah Subhanahu wa Ta'ala, yang dengan rahmat dan kemudahan dari-Nya, penelitian dan penulisan tugas akhir yang berjudul "Persepsi Relevansi pada Pencarian Semantik *E-Commerce*: Perbandingan Kualitatif *Model Embedding* OpenAI dan VoyageAI dengan Metode *Blind A/B Testing*" ini dapat terselesaikan. Proses ini merupakan sebuah perjalanan intelektual yang penuh tantangan, menguji ketekunan, dan menuntut pemikiran yang kritis.

Penyelesaian tugas akhir ini tidak akan terwujud tanpa bimbingan, dukungan, dan doa dari berbagai pihak. Oleh karena itu, dengan tulus saya ingin menyampaikan rasa terima kasih yang mendalam kepada:

1. Allah Subhanahu wa Ta'ala atas segala Rahmat, Nikmat, dan Karunia-Nya sehingga laporan tugas akhir ini dapat diselesaikan dengan baik.
2. Dr. Mukhammad Andri Setiawan S.T., M.Sc. dosen pembimbing terbaik yang luar biasa dan inspiratif. Beliau membimbing sebagai seorang pribadi yang menunjukkan keteladanan sebagai seorang akademisi sejati.
3. Kedua orang tua saya yang hebat dan tersayang, Bapak Daris Munarto dan Mama Inayatul Bundariyah serta serta kedua kakak tercinta, Priliese Indra Devi dan Robby Indra Lukman. Dukungan moril, materiil, dan doa yang tak pernah putus yang menjadi bekal terkuat yang membuat saya bertahan hingga titik ini.
4. Rekan terbaik, sahabat, dan pihak lain yang tidak dapat saya sebutkan satu per satu. Kontribusi, diskusi, dan dukungan sekecil apa pun memiliki arti yang sangat besar dalam penyelesaian penelitian ini.

Penyelesaian tugas akhir ini bukanlah akhir dari sebuah proses belajar, melainkan sebuah titik awal. Harapan terbesar penulis adalah agar karya ini dapat menjadi pijakan yang memicu diskusi lebih lanjut dan memberikan kontribusi nyata dalam pengembangan sistem pencarian yang lebih memahami manusia. Semoga penelitian ini dapat bermanfaat bagi para pembaca dan menjadi dasar bagi pengembangan selanjutnya.

Yogyakarta, 15 Juli 2025



(Adyuta Indra Adyatma)

SARI

Penelitian ini bertujuan untuk mengevaluasi secara kualitatif efektivitas dua model embedding, yaitu OpenAI *text-embedding-3-large* dan VoyageAI *voyage-3-large*, dalam konteks pencarian semantik pada platform *e-commerce*. Latar belakang penelitian ini adalah adanya keterbatasan sistem pencarian berbasis kata kunci yang seringkali gagal memahami intensi pengguna, sehingga menghasilkan pencarian yang tidak relevan. Meskipun pencarian semantik menawarkan solusi, pemahaman mendalam mengenai persepsi pengguna terhadap hasil dari model yang berbeda masih menjadi area yang perlu dieksplorasi untuk meningkatkan pengalaman belanja *online*.

Untuk menunjang penelitian ini, dikembangkan sebuah aplikasi *e-commerce* berbasis *website*. Aplikasi ini memiliki fitur pencarian semantik yang terintegrasi dengan API dari OpenAI dan VoyageAI, serta didukung oleh Pinecone sebagai *vector database* untuk menyimpan dan mencari representasi vektor produk. Sistem ini dibangun menggunakan kerangka kerja pengembangan Next.js untuk antarmuka pengguna dan Supabase sebagai basis data relasional, yang memungkinkan dilakukannya perbandingan langsung antara kedua *model embedding* dalam lingkungan yang terkontrol.

Metodologi penelitian yang digunakan adalah pendekatan kualitatif dengan desain eksperimen *blind A/B testing*. Partisipan dengan pengetahuan domain produk yang relevan diminta untuk melakukan serangkaian tugas pencarian menggunakan kedua model yang dilabeli secara netral ('Model A' dan 'Model B'). Data kualitatif dikumpulkan melalui wawancara mendalam untuk menggali persepsi pengguna mengenai relevansi, akurasi, dan kepuasan terhadap hasil pencarian dari masing-masing model.

Temuan utama dari penelitian ini menunjukkan adanya ketidakselarasan antara metrik teknis dan persepsi pengguna. Meskipun model VoyageAI seringkali menghasilkan skor kemiripan (similarity score) yang lebih tinggi, mayoritas partisipan lebih menyukai hasil dari model OpenAI karena dinilai lebih unggul dalam memahami konteks dan menangani kueri yang ambigu. Hasil ini menyimpulkan bahwa preferensi pengguna tidak semata-mata ditentukan oleh akurasi literal, melainkan oleh kemampuan model dalam mendukung penemuan produk secara intuitif, yang menyoroti adanya *trade-off* strategis dalam pemilihan *model embedding*.

Kata kunci: *Pencarian Semantik, Model Embedding, Persepsi Pengguna, Evaluasi Kualitatif, A/B Testing, E-commerce, OpenAI, VoyageAI.*

GLOSARIUM

Blind A/B Testing	Metode evaluasi di mana dua versi (A dan B) dari sebuah fitur dibandingkan secara langsung oleh pengguna yang tidak mengetahui versi mana yang sedang mereka uji, untuk mendapatkan umpan balik yang objektif.
API	Sebuah antarmuka yang memungkinkan dua aplikasi perangkat lunak untuk berkomunikasi satu sama lain, seperti antara sistem penelitian ini dengan layanan OpenAI dan VoyageAI.
Cosine Similarity	Rumus matematika yang digunakan untuk mengukur kesamaan antara dua vektor dengan menghitung kosinus sudut di antara keduanya. Skor yang mendekati 1 menandakan kesamaan semantik yang tinggi.
Dimensi	Jumlah komponen dalam sebuah vektor *embedding*. Dimensi yang lebih tinggi (misalnya, 3072) dapat menangkap makna yang lebih kompleks tetapi membutuhkan lebih banyak sumber daya komputasi.
Embedding	Proses mengubah data tekstual (kata atau kalimat) menjadi representasi numerik dalam bentuk vektor di ruang multi-dimensi. Vektor yang berdekatan secara geometris merepresentasikan konsep yang mirip secara makna.
Extreme Programming	Sebuah metodologi pengembangan perangkat lunak yang berfokus pada fleksibilitas dan iterasi cepat, yang diadopsi dalam pengembangan sistem pada penelitian ini.
LLM	Model kecerdasan buatan yang dilatih pada data teks dalam jumlah masif sehingga mampu memahami dan menghasilkan bahasa manusia.
Similarity Search	Proses pencarian dalam vector database untuk menemukan vektor-vektor yang paling mirip atau paling dekat secara geometris dengan sebuah vektor kueri.
Vector Database	Basis data yang dirancang khusus untuk menyimpan, mengelola, dan melakukan pencarian secara efisien pada data dalam format vektor berdimensi tinggi, seperti embeddings.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Di era digital yang semakin kompetitif, platform *e-commerce* tidak hanya dituntut untuk menyediakan kemudahan bertransaksi, tetapi juga harus mampu memahami preferensi dan kebutuhan pengguna. Salah satu interaksi penting adalah melalui fitur pencarian, di mana pengguna seringkali memasukkan kueri yang beragam untuk menemukan produk yang diinginkan. Namun, sistem pencarian konvensional yang mengandalkan pencocokan kata kunci (*keyword-based search*) kerap menemui kendala dalam menangkap makna semantik di balik kueri pengguna. Akibatnya, hasil pencarian produk seringkali kurang akurat, tidak relevan, dan gagal memenuhi ekspektasi pengguna. Kondisi ini berpotensi menurunkan kepuasan pengguna, padahal pengalaman belanja yang positif merupakan faktor kunci dalam membangun loyalitas pelanggan di tengah persaingan industri *e-commerce* yang ketat. Ketidakpuasan pengalaman pengguna seperti hasil pencarian yang buruk dapat menjadi motivasi untuk pengguna beralih ke platform lain yang menawarkan pengalaman lebih intuitif (Rawis dkk., 2022).

Perkembangan dalam *Natural Language Processing* (NLP) menawarkan solusi untuk menjembatani kesenjangan pemahaman antara bahasa manusia dan mesin (Maithili dkk., 2023). NLP, melalui kombinasi teknik pemrosesan bahasa alami dan kecerdasan buatan, memungkinkan analisis teks yang lebih mendalam, melampaui pencocokan kata kunci hingga ke pemahaman konteks dan makna semantik. Salah satu teknologi NLP yang fundamental dalam hal ini adalah *text embeddings*, yang merepresentasikan kata atau frasa ke dalam representasi vektor numerik berdimensi tinggi dengan mempertimbangkan kedekatan makna semantiknya (Asudani dkk., 2023). Pendekatan ini memungkinkan sistem pencarian untuk mengidentifikasi hubungan konseptual antar istilah, tidak hanya berdasarkan kesamaan leksikal, dan meningkatkan relevansi pencarian serta rekomendasi produk di *e-commerce* (Zhang & Chai, 2021.).

Meskipun implementasi *embeddings* telah banyak digunakan, pemilihan model *embeddings* yang optimal untuk konteks pencarian produk *e-commerce* yang spesifik, serta infrastruktur pendukungnya, masih menjadi area eksplorasi yang penting. Model *embeddings* seperti *text-embedding-3-large* dari OpenAI dan *voyage-3-large* dari VoyageAI menawarkan

ketersediaan dan kemudahan melalui API yang memudahkan integrasi. Di sisi lain, *vector database* seperti Pinecone menyediakan arsitektur yang dirancang khusus untuk menyimpan, mengelola, dan melakukan pencarian data vektor berdimensi tinggi secara efisien (Ma dkk., 2025). Kombinasi *model embeddings* dengan *vector database* ini dapat meningkatkan performa pencarian secara signifikan.

Namun, studi yang secara mendalam membandingkan kinerja *model embeddings* spesifik seperti OpenAI *text-embedding-3-large* dan VoyageAI *voyage-3-large*, yang diimplementasikan dengan *vector database* Pinecone untuk pencarian semantik di *e-commerce*, masih terbatas. Kebanyakan penelitian cenderung fokus pada evaluasi metrik kuantitatif. Padahal, persepsi dan pengalaman pengguna nyata, terutama dari mereka yang memiliki pengetahuan domain produk, memegang peranan penting dalam menilai efektivitas sebenarnya dari solusi pencarian semantik. Evaluasi kualitatif dapat mengungkap nuansa preferensi pengguna yang mungkin tidak tertangkap oleh indikator kuantitatif semata.

Penelitian ini bertujuan untuk menjembatani celah tersebut dengan berfokus pada dua domain produk yang relevan: *fitness* dan *skincare.* Pemilihan kedua domain ini didasari pada relevansi kedua pasar dengan tren penjualan dengan volume yang masif. Pasar *e-commerce fitness* dan *skincare* menjadi komponen ekonomi kesejahteraan global dengan nilai $6.3 triliun pada tahun 2023 untuk pasar kesehatan dan kebugaran (Gupta, 2025). Sementara itu, pasar *e-commerce* khusus produk perawatan diri sebesar $23.29 miliar pada tahun yang sama (Singh, 2025). Namun yang lebih krusial, keduanya mencerminkan pola perilaku pencarian pengguna yang berbeda secara fundamental.

Domain fitness cenderung melibatkan pencarian berbasis tujuan dan spesifikasi teknis. Konsumen seringkali mencari solusi untuk mencapai target kebugaran, sehingga evaluasi produk melibatkan pertimbangan pada fitur dan spesifikasi teknis (Fife dkk., 2023, 2024; Lee & Lee, 2018). Sebaliknya, domain skincare didominasi oleh pencarian berbasis kebutuhan personal dan solusi masalah (needs-oriented & problem-solving search). Pengguna mencari solusi untuk kondisi kulit yang personal dan sangat bergantung pada bahan yang digunakan oleh produk (Choi & Lee, 2019). Karakteristik ini menjadikan kedua domain sebagai tolok ukur untuk mengevaluasi kemampuan model dalam menangani variasi intensi dan ambiguitas kueri.

Melalui pengembangan sebuah platform e-commerce fungsional, penelitian ini akan melakukan analisis perbandingan kualitatif secara langsung antara model embedding OpenAI dan VoyageAI pada kedua domain tersebut. Dengan menggunakan metode blind A/B testing

yang melibatkan pengguna dengan pengetahuan domain, penelitian ini akan menggali umpan balik mendalam mengenai persepsi mereka terhadap relevansi dan akurasi hasil pencarian dari kedua model. Diharapkan, temuan dari studi ini dapat menjadi referensi strategis bagi pengembang dalam membangun sistem pencarian e-commerce yang lebih cerdas dan responsif terhadap kebutuhan manusia.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, terlihat bahwa implementasi pencarian semantik menggunakan model *embeddings* dan *vector database* memiliki potensi besar untuk meningkatkan kualitas pencarian di platform *e-commerce*. Meskipun demikian, pemilihan *model embeddings* yang tepat serta pemahaman mendalam mengenai persepsi pengguna terhadap hasil pencarian dari model yang berbeda masih memerlukan investigasi lebih lanjut. Oleh karena itu, penelitian ini merumuskan masalah sebagai berikut:

a. Bagaimana perbandingan efektivitas antara *model embeddings* OpenAI *text-embedding-3-large* dan VoyageAI *voyage-3-large* dalam menghasilkan pencarian produk yang relevan dan akurat pada platform *e-commerce*, berdasarkan evaluasi A/B testing dari pengguna dengan pengetahuan domain produk?
b. Faktor-faktor apa saja dari hasil pencarian (misalnya, relevansi, keberagaman, penanganan ambiguitas) yang paling memengaruhi preferensi pengguna terhadap salah satu *model embeddings* dalam skenario pencarian semantik *e-commerce*?
c. Bagaimana perbedaan arsitektur, khususnya ukuran dimensi vektor pada model OpenAI *text-embedding-3-large* dan VoyageAI *voyage-3-large*, memengaruhi persepsi pengguna terhadap relevansi dan kualitas hasil pencarian, dan apa implikasinya bagi pemilihan model dalam konteks *e-commerce*?

1.3 Batasan Masalah dan Lingkup Penelitian

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

a. Dataset dan Deskripsi Produk

   Penelitian ini menggunakan dataset "Amazon Products Dataset 2023" dari Kaggle (Parab, 2023). Mengingat dataset tersebut tidak menyertakan deskripsi produk yang detail, deskripsi produk akan di-*generate* menggunakan model GPT-4o sebagai bagian dari *pre-processing* data untuk memberikan kedalaman konteks. Validitas dan representativitas deskripsi yang di-*generate* ini akan diakui sebagai salah satu batasan.

b. *Model Embeddings* yang Digunakan

Perbandingan model dalam penelitian ini hanya melibatkan model *embeddings* dari OpenAI (*text-embedding-3-large*) dan VoyageAI (*voyage-3-large*). Kedua model ini dipilih karena keduanya tersedia melalui layanan API, yang memudahkan integrasi dengan proyek. Model lain di luar kedua model tersebut tidak menjadi fokus penelitian.

c. Pendekatan Persepsi Relevansi

Penelitian ini secara tegas membatasi pada pendekatan persepsi relevansi satu arah, yaitu relevansi dipahami sebagai penilaian subjektif dari sisi pengguna terhadap hasil pencarian berdasarkan kueri eksplisit yang dimasukkan. Penelitian ini tidak mencakup pendekatan dua arah yang melibatkan sistem dalam memprediksi atau merekomendasikan kebutuhan pengguna. Dengan batasan ini, penelitian berfokus pada evaluasi kualitatif terhadap bagaimana sistem pencarian semantik menginterpretasi input eksplisit pengguna, bukan pada mekanisme proaktif berbasis psikologi prediktif.

d. Metode Evaluasi Kualitatif

Evaluasi kualitatif dalam penelitian ini dilakukan melalui proses *blind A/B Testing* yang melibatkan wawancara mendalam dengan pengguna terpilih (dengan pengetahuan domain produk). Pengujian difokuskan pada perbandingan langsung hasil pencarian antara *model embeddings* OpenAI (*text-embedding-3-large*) dan VoyageAI (*voyage-3-large*) secara kualitatif dalam konteks pencarian semantik di platform *e-commerce*. Umpan balik yang dikumpulkan mencakup aspek kepuasan pengguna, persepsi relevansi hasil pencarian, serta pemahaman terhadap konteks kueri pencarian. Penelitian ini hanya mengandalkan metode evaluasi kualitatif melalui *blind A/B Testing* yang telah dirancang, dan tidak mencakup metode lain di luar skenario tersebut.

e. Fokus Implementasi

Pengembangan sistem pencarian semantik dilakukan dalam bentuk prototipe menggunakan NextJS 14, dengan Pinecone sebagai *vector database* dan Supabase PostgreSQL sebagai basis data relasional. Penelitian tidak mencakup implementasi skala produksi penuh atau optimasi kinerja sistem di luar kebutuhan evaluasi.

f. Lingkup Partisipan

Partisipan evaluasi adalah individu yang memiliki pengetahuan atau pengalaman dalam domain produk yang diuji (misalnya, peralatan *fitness* dan produk perawatan kulit) dan memiliki pengalaman berbelanja *online.*

1.4 Tujuan Penelitian

Secara umum, penelitian ini bertujuan untuk mengevaluasi dan membandingkan secara kualitatif efektivitas *model embeddings* OpenAI *text-embedding-3-large* dan VoyageAI *voyage-3-large* dalam meningkatkan pengalaman pencarian semantik pada platform *e-commerce*. Secara spesifik, tujuan penelitian ini adalah:

a. Menganalisis dan membandingkan kualitas hasil pencarian produk yang dihasilkan oleh *model embeddings* OpenAI *text-embedding-3-large* dan VoyageAI *voyage-3-large* berdasarkan relevansi, akurasi, dan kemampuan menangani berbagai jenis kueri, melalui evaluasi kualitatif pengguna.
b. Mengidentifikasi preferensi pengguna serta faktor-faktor kualitatif yang mendasari preferensi tersebut terhadap salah satu *model embeddings* dalam konteks pencarian semantik *e-commerce*.
c. Memberikan rekomendasi berbasis bukti mengenai pertimbangan dalam pemilihan dan implementasi *model embeddings* untuk optimalisasi fitur pencarian semantik di platform *e-commerce.*

1.5 Manfaat Penelitian

A. Manfaat Teoretis/Akademis

a. Memperkaya literatur ilmiah mengenai evaluasi perbandingan *model embeddings* dalam domain pencarian semantik *e-commerce*, khususnya dengan penekanan pada evaluasi kualitatif dan persepsi pengguna.
b. Menyumbangkan pemahaman mengenai faktor-faktor yang memengaruhi kualitas pencarian semantik dari perspektif pengguna dengan pengetahuan domain.

B. Manfaat Praktis

a. Memberikan panduan dan wawasan bagi pengembang *e-commerce* dalam memilih *model embeddings* yang lebih sesuai dengan kebutuhan platform dan ekspektasi pengguna mereka.
b. Menyediakan bukti empiris mengenai bagaimana *model embeddings* yang berbeda secara arsitektural menangani berbagai jenis kueri pengguna, yang dapat digunakan untuk mengoptimalkan strategi pencarian.
c. Menganalisis implikasi dari perbedaan arsitektur model terhadap preferensi pengguna untuk memberikan rekomendasi berbasis bukti mengenai pertimbangan dalam pemilihan dan implementasi model embeddings untuk optimalisasi fitur pencarian semantik.

Dengan demikian, penelitian ini diharapkan dapat menghadirkan solusi praktis serta landasan teoretis yang mendalam dalam pemanfaatan *model embeddings* untuk pencarian semantik dalam konteks *e-commerce*. Penelitian ini tidak hanya menyediakan kerangka evaluasi menyeluruh berbasis *blind A/B testing* dan evaluasi pengguna, tetapi juga melibatkan perspektif kualitatif untuk menggali secara mendalam preferensi dan pengalaman pengguna.

Hasil penelitian diharapkan menjadi referensi strategis bagi pengembang dan pengelola platform *e-commerce* dalam meningkatkan efektivitas sistem pencarian, sehingga dapat mendukung pengambilan keputusan yang lebih tepat dalam pengembangan teknologi.

1.6 Sistematika Penulisan

Sistematika penulisan dari penelitian ini adalah sebagai berikut:

a. BAB I PENDAHULUAN

Menjelaskan latar belakang penelitian, rumusan masalah, tujuan penelitian, manfaat penelitian, serta batasan masalah. Pembahasan pada bab ini menjadi landasan utama untuk memahami konteks dan kepentingan penelitian.

b. BAB II KAJIAN PUSTAKA

Memaparkan landasan teoretis yang relevan, mulai dari kebutuhan pencarian cerdas dalam *e-commerce*, psikologi pencarian pengguna, hingga teknologi inti seperti text embeddings, arsitektur model, *vector database*, dan *prompt engineering*. Selain itu, dibahas pula tinjauan pustaka dari penelitian terdahulu untuk memposisikan kontribusi penelitian ini.

c. BAB III METODOLOGI PENELITIAN

Menguraikan dua metodologi utama yang digunakan. Pertama, metodologi pengembangan perangkat lunak menggunakan Extreme Programming (XP) yang mencakup fase perencanaan, desain, implementasi, dan pengujian sistem. Kedua, metodologi evaluasi penelitian yang menggunakan pendekatan kualitatif dengan desain blind A/B testing untuk membandingkan kedua model embeddings, lengkap dengan prosedur dan kriteria partisipan.

d. BAB IV HASIL DAN PEMBAHASAN

Menguraikan fase *Coding* dan *Testing* yang mencakup hasil penerapan fitur pencarian semantik yang telah dikembangkan, serta temuan kualitatif dari *blind A/B Testing* terkait kepuasan dan preferensi pengguna terhadap model embeddings yang dibandingkan.

e. BAB V KESIMPULAN

Berisi kesimpulan yang diambil dari seluruh proses penelitian dan memberikan saran-saran untuk pengembangan lebih lanjut sistem pencarian semantik berbasis model embeddings di platform *e-commerce.*

BAB II
DASAR TEORI DAN TINJAUAN PUSTAKA

2.1 Kebutuhan akan Pencarian Cerdas dalam *E-Commerce*

Evolusi pesat *e-commerce* telah mengubah cara konsumen menemukan dan membeli produk secara daring. Memahami bagaimana sistem ini meningkatkan penemuan produk, keterlibatan pengguna, dan tingkat konversi semakin penting (Sajid dkk., 2022). Dalam ekosistem yang terus berkembang ini, kegunaan mesin pencari berfungsi sebagai salah satu antarmuka utama antara konsumen dan katalog produk. Kualitas interaksi ini secara langsung memengaruhi penemuan produk, keterlibatan pengguna, dan pada akhirnya, tingkat konversi atau penjualan. Oleh karena itu, efektivitas sistem pencarian bukan lagi sekadar fitur tambahan, melainkan sebuah langkah strategis yang menentukan keunggulan kompetitif sebuah *platform* (Almunawar dkk., 2021).

Meskipun telah lama menjadi standar industri, sistem pencarian tradisional yang berbasis kata kunci (*keyword-based*) memiliki keterbatasan mendasar yang semakin terasa di tengah kompleksitas perilaku belanja modern. Prinsip kerja sistem ini adalah pencocokan leksikal atau kata kata yang digunakan, di mana sistem dimasukkan pengguna dalam deskripsi produk. Keterbatasan utama dari pendekatan ini adalah mencari kemunculan persis dari istilah yang ketidakmampuannya untuk memahami maksud atau intensi pengguna *(user intent)* yang sesungguhnya. Fenomena ini merupakan perwujudan dari "masalah ketidakcocokan kosakata" (*vocabulary mismatch problem*), di mana pengguna dan sistem menggunakan istilah yang berbeda untuk merujuk pada konsep yang sama (Zhan dkk., 2021). Sebagai contoh, seorang pengguna mungkin mencari "sepatu lari yang empuk", sementara produk yang relevan mungkin dideskripsikan sebagai "alas kaki dengan bantalan maksimal". Sistem berbasis kata kunci akan kesulitan menjembatani kesenjangan semantik ini.

Kegagalan dalam memahami makna yang lebih mendalam dan konteks ini secara langsung berdampak pada pengalaman pengguna. Penelitian menunjukkan hasil pencarian yang tidak relevan dapat menciptakan beban kognitif, menghambat kemampuan pengguna untuk menemukan informasi yang relevan secara efisien. Calleja dan Willoughby menjelaskan bagaimana informasi yang tidak relevan dapat memperlambat proses pencarian, dengan menekankan mengenai ketidaksesuaian semantik mengalihkan fokus pengguna dan dapat

mengakibatkan pengguna meninggalkan tugas pencarian sepenuhnya (Calleja & Willoughby, 2023).

Namun, untuk benar-benar mengatasi keterbatasan ini, kita tidak hanya memerlukan teknologi yang lebih cerdas, tetapi juga pemahaman mendalam tentang bagaimana pengguna sebenarnya mencari informasi dalam konteks *e-commerce* yang kompleks. Perilaku pengguna inilah yang pada akhirnya menentukan persepsi terhadap suatu sistem pencarian pada *e-commerce*.

2.2 Psikologi Pencarian Pengguna di *E-Commerce*

Memahami pengguna adalah kunci. Perilaku pencarian informasi sangat beragam dan tidak terjadi dalam satu pola yang sama antar pengguna. Pengguna menggunakan sistem pencarian dengan tujuan dan tingkat kepastian yang berbeda-beda, yang secara signifikan memengaruhi apa yang mereka anggap sebagai hasil yang "baik". Dalam konteks *e-commerce*, membedakan antara dua mode utama perilaku pencarian—*lookup* dan eksplorasi—sangatlah penting.

Secara umum, aktivitas pencarian dapat diklasifikasikan ke dalam dua kategori utama. Kategori pertama adalah pencarian *lookup (lookup search)*, yang terjadi ketika pengguna memiliki tujuan yang sangat spesifik dan terdefinisi dengan baik (misalnya, mencari "*iPhone 15 Pro Max 256GB*"). Jenis pencarian ini mencerminkan tujuan pengguna yang terdefinisi dengan baik dan biasanya menghasilkan perilaku pencarian yang dapat lebih diprediksi. Jumlah aktivitas klik acak terfokuskan pada hasil pencarian teratas dengan menarik perhatian sekitar 90% dari total klik, menyoroti peran penting peringkat pada mesin pencarian dalam membentuk hasil dan kepuasan pengguna (Urman & Makhortykh, 2021).

Kategori kedua adalah *pencarian eksplorasi (exploratory search)*, yang mencakup variasi aktivitas yang jauh lebih luas, di mana pengguna mungkin tidak familiar dengan domain yang dicari atau tidak yakin tentang tujuan mereka (misalnya, "*Sesuatu yang dapat membuat saya menjadi lebih kuat*"). Pengguna sering kali memulai dengan kebutuhan yang samar-samar dan menggunakan proses pencarian untuk memperjelas pilihan mereka. Untuk jenis pencarian ini, definisi "relevan" menjadi lebih cair dan bernuansa. Variasi tujuan pengguna selama pencarian eksploratif menunjukkan bahwa memahami intensi pencarian adalah tantangan multi-aspek, yang sering dipengaruhi oleh faktor kontekstual dan tingkat pemahaman pengguna dengan topik yang dicari (Urman dkk., 2021).

Hal ini mengarah pada dilema implementasi dalam mesin pencari *e-commerce*, yaitu *trade-off* antara eksploitasi dan eksplorasi. Eksploitasi berfokus pada presisi atau ketepatan

untuk pengguna *lookup*, sementara eksplorasi berfokus pada keragaman atau variasi untuk pengguna eksplorasi. Hal ini menjadikan tantangan yang cukup rumit, mengutamakan presisi secara berlebihan dapat berdampak negatif, karena "menghilangkan kesenangan dari eksplorasi" dan mengurangi keterlibatan pengguna dalam jangka panjang. Sebaliknya, kurangnya presisi dan tingginya keragaman hasil dapat membuat pengguna "*lookup search*" kewalahan, memperpanjang waktu yang dibutuhkan untuk menemukan informasi spesifik, dan pada akhirnya menurunkan efisiensi pencarian mereka .

Studi mengatakan perilaku pengguna dapat berbeda tergantung konteks, seperti bagaimana sikap politik dapat membentuk kueri pencarian terkait topik kontroversial (van Hoof dkk., 2024). Selain itu, penelitian juga menegaskan perlunya mesin pencari yang secara efektif mampu menginterpretasikan dan merespon kebutuhan pengguna yang bervariasi, serta memastikan hasilnya selaras dengan harapan dan kebutuhan pengguna (Gupta & Tiwari, 2024). Dikarenakan strategi pencarian pengguna berbeda secara signifikan antara *lookup* dan *exploratory*, sistem pencarian harus beradaptasi untuk mengakomodasi kepuasan pengguna dalam berbagai kebutuhan atau keinginan.

Dengan memahami tantangan ganda dari perilaku *lookup search* dan *exploratory* ini, kita sekarang dapat mengkaji teknologi pencarian semantik yang dirancang untuk menjawab kebutuhan ini. Berbeda dengan pencocokan kata kunci, pencarian semantik bertujuan untuk memahami makna dan hubungan antar kata, meniru pemahaman manusia terhadap bahasa. Alih-alih hanya mencocokkan string teks, sistem ini menafsirkan maksud di balik kueri pengguna untuk memberikan hasil yang secara konseptual itu relevan, bahkan jika tidak ada tumpang tindih kata kunci sama sekali.

Salah satu pendekatan dalam pencarian semantik adalah *Natural Language Processing* (NLP) yang memanfaatkan *Large Language Models (*LLMs). LLM meningkatkan kemampuan sistem pencarian untuk memahami maksud pengguna dalam konteks percakapan, sehingga meningkatkan relevansi dan keakuratan hasil pencarian (Yuan dkk., 2025). Lebih lanjut, Sultana dkk. membahas bagaimana pemahaman semantik melalui LLM memfasilitasi interpretasi kueri yang lebih canggih, yang secara langsung mengarah pada keterlibatan dan kepuasan pengguna yang lebih baik (Sultana dkk., 2024).

2.3 Konsep Persepsi Relevansi Dalam Konteks Pencarian Semantik

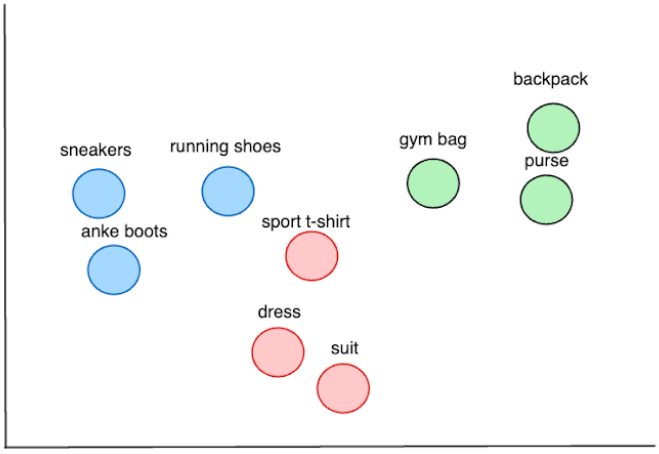
Keberhasilan sebuah sistem pencarian semantik tidak hanya ditentukan oleh metrik teknis, melainkan juga oleh sejauh mana sistem mampu memenuhi kebutuhan informasi pengguna secara kontekstual. Dalam kerangka ini, relevansi dipahami sebagai persepsi subjektif

pengguna terhadap kesesuaian hasil pencarian. (Ruthven, 2021) menegaskan bahwa relevansi lebih tepat dipandang sebagai pengalaman pengguna *(user experience)* yang mencakup kesesuaian semantik, kegunaan, hingga kepuasan subjektif. Pandangan ini dilengkapi oleh (Peikos & Pasi, 2024), yang memandang relevansi sebagai konsep dinamis dengan dimensi *topicality, usefulness, context-fit,* dan *satisfaction*. Namun, perlu ditegaskan bahwa penelitian ini hanya mengadopsi pendekatan satu arah. Persepsi relevansi dinilai berdasarkan bagaimana pengguna menanggapi hasil pencarian yang dihasilkan dari kueri eksplisit yang mereka masukkan. Pendekatan ini konsisten dengan penelitian mutakhir oleh (Ahluwalia dkk., 2024), yang menekankan pentingnya pemrosesan kueri eksplisit pengguna dalam model semantic search. Walaupun penelitian tersebut menggabungkan *keyword search, embeddings*, dan *query structuring*, dasarnya tetap bertumpu pada input eksplisit pengguna. Dengan demikian, penelitian ini membatasi penggunaan konsep persepsi relevansi hanya pada interaksi satu arah, yakni dari pengguna ke sistem. Pendekatan ini sejalan dengan tujuan penelitian untuk menilai efektivitas *model embedding* dalam menjawab kebutuhan pencarian berdasarkan kueri eksplisit, serta memberikan dasar yang jelas untuk membedakan ruang lingkup penelitian ini dari pendekatan dua arah yang melibatkan prediksi kebutuhan.

2.4 Teknologi Pencarian Semantik

Untuk memahami cara kerja pencarian semantik, penting untuk mengkaji teknologi inti yang memungkinkannya: representasi *vector embeddings* dan arsitektur model yang mendasarinya. Perbedaan dalam implementasi teknologi ini, terutama dalam hal ukuran dimensi, secara langsung memengaruhi kemampuan dan karakteristik kinerja sistem pencarian, seperti yang diamati dalam penelitian ini.

2.4.1 Representasi Semantik (*Embeddings*)



Gambar 2.1 Representasi vektor dalam ruang vektor

Sumber: Ziliz

Dasar utama dari pencarian semantik modern adalah konsep *text embeddings*. *Embedding* adalah proses mengubah data tekstual—seperti kata, kalimat, atau deskripsi produk—menjadi representasi numerik padat dalam bentuk vektor di dalam ruang multi-dimensi. Gambar 2.1 menunjukkan ruang vektor di mana kedekatan geometris antar vektor merepresentasikan kedekatan semantik antar konsep yang diwakilinya (R. Liu, 2024). Dengan kata lain, produk dengan deskripsi "*gym bag*" dan "*backpack*" akan dipetakan ke titik-titik yang berdekatan di dalam ruang vektor, meskipun keduanya menggunakan kata-kata yang berbeda.

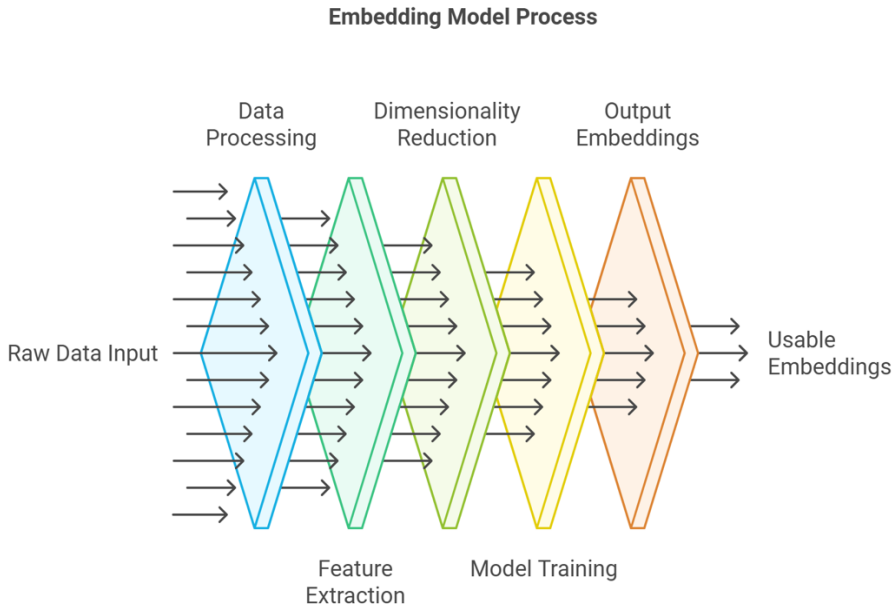
Pendekatan ini merupakan peningkatan dari metode representasi teks tradisional seperti *bag-of-words* (BoW) atau TF-IDF. Metode-metode lama menghasilkan representasi yang *sparse* (jarang), di mana sebagian besar elemen vektor adalah nol, dan setiap dimensi secara kaku merepresentasikan satu kata spesifik dalam kosakata. Representasi *sparse* ini tidak kurang efisien secara komputasi karena dimensionalitasnya yang sangat tinggi (sama dengan jumlah kata unik), tetapi juga gagal total dalam menangkap hubungan semantik seperti sinonim atau parafrasa (Seinen dkk., 2022).

Sebaliknya, *embedding* menghasilkan vektor yang *dense* (padat) dan berdimensi lebih rendah (meskipun masih tinggi, misalnya ratusan hingga ribuan dimensi), di mana setiap dimensi bersama-sama membentuk makna. Ketika kueri pencarian masuk, kueri tersebut juga diubah menjadi vektor menggunakan *model embedding* yang sama. Sistem kemudian menghitung kesamaan antara vektor kueri dan semua vektor produk di dalam *index vector database*. Metode perhitungan yang paling umum digunakan adalah *cosine similarity*, yang mengukur sudut antara dua vektor. Sudut yang lebih kecil (menghasilkan skor kosinus yang

lebih tinggi, mendekati 1) menandakan kesamaan semantik yang lebih besar. Proses ini memungkinkan sistem untuk menemukan produk yang relevan secara konteks, bukan hanya yang cocok secara leksikal atau kata kata yang digunakan (Neelakantan dkk., t.t.). Setelah memahami bagaimana *embeddings* mengubah teks menjadi vektor berdimensi rendah, langkah selanjutnya adalah mempelajari cara kerja model-model tersebut, mulai dari komponen penyusunnya hingga proses pelatihannya, untuk menghasilkan embedding seperti yang telah dijelaskan.

2.4.2 Arsitektur Model *Embedding*

Model *embedding* adalah teknik dalam *Natural Language Processing* (NLP) yang memetakan teks—baik tingkat kata, frasa, maupun dokumen—ke dalam vektor numerik berdimensi. Representasi vektor ini memungkinkan sistem untuk menghitung kemiripan, analogi, atau hubungan semantik antar-teks, sehingga memudahkan tugas seperti pencarian, klasifikasi, dan rekomendasi (Sun dkk., 2021).



Gambar 2. 2 Proses Pembuatan *Embeddings* menggunakan *Model Embeddings*

Sumber: (Mitchell, 2024)

Secara umum, proses kerja sebuah *model embedding* diilustrasikan pada Gambar 2. 2 dapat diuraikan dalam beberapa tahapan. Tahap pertama adalah pemrosesan data mentah (*input data processing*), di mana *text input* dipecah menjadi unit-unit yang lebih kecil melalui proses tokenisasi (*tokenization*). *Token* ini kemudian dimasukkan ke dalam *model embedding* yang sebelumnya telah dilatih pada dataset yang sangat besar. Selanjutnya, model melakukan

ekstraksi fitur (*feature extraction*) untuk menganalisis konteks dan makna dari teks tersebut. Model yang sudah terlatih ini kemudian memadatkan informasi tersebut ke dalam representasi vektor berdimensi lebih rendah. Hasil akhir dari proses ini adalah keluaran berupa *embedding*—sebuah vektor numerik padat *(dense vector embeddings)* yang siap digunakan untuk tugas pencarian semantik.

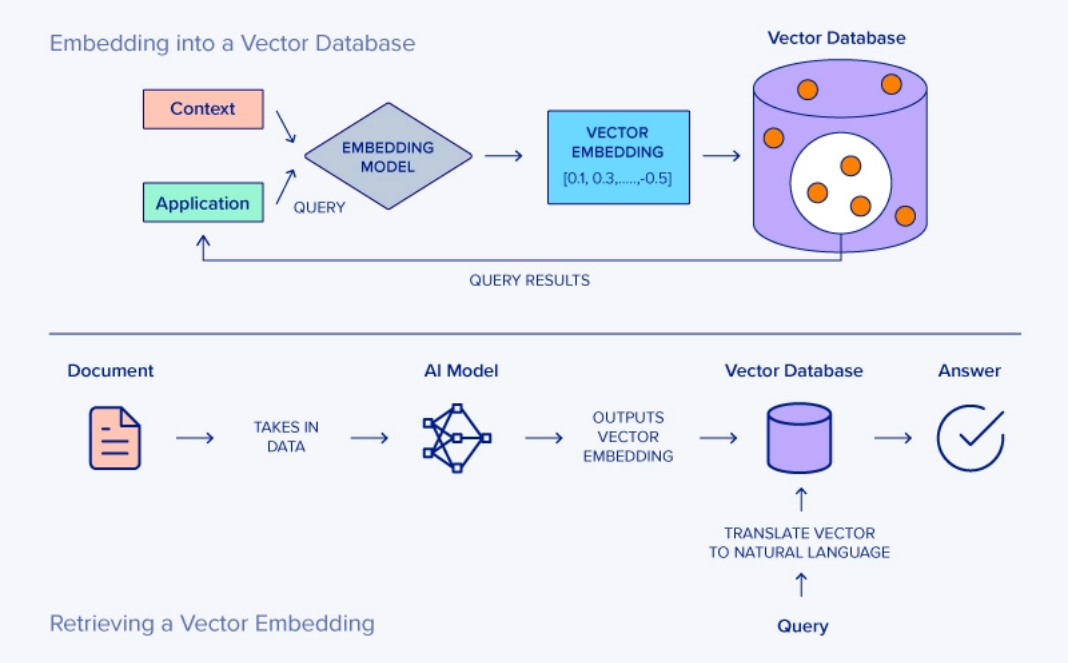
Penelitian ini secara spesifik membandingkan dua model *embedding*: OpenAI *text-embedding-3-large* dan VoyageAI *voyage-3-large*. Salah satu perbedaan arsitektural dari kedua buah model ini adalah *vector embedding* yang dihasilkan: 3072 dimensi untuk OpenAI *text-embedding-3-large* dan *voyage-3-large* 1024 dimensi untuk VoyageAI. Pemilihan ukuran dimensi ini mencerminkan desain arsitektur yang berbeda dan memiliki pengaruh terhadap kemampuan model. Ukuran dimensi yang lebih tinggi, seperti 3072 pada model OpenAI *text-embedding-3-large*, memberikan kemampuan "representasi" yang lebih besar bagi model untuk menangkap hubungan yang kompleks dan bernuansa. Kemampuan representasi yang lebih besar ini secara teoretis memungkinkan model untuk lebih baik dalam mengurai kueri yang ambigu (Albujasim dkk., 2023).

Namun, kemampuan representasi yang lebih besar ini memiliki *trade off* atau konsekuensi. Ukuran dimensi yang lebih tinggi secara langsung berimplikasi pada penggunaan memori yang lebih besar, karena penyimpanan vektor produk dengan 3072 dimensi memerlukan sumber daya yang jauh lebih besar dibandingkan 1024 dimensi (Johnson dkk., 2021). Di samping itu, proses kalkulasi pada vektor berdimensi tinggi—seperti menghitung *similarity score*—memakan waktu komputasi yang lebih lama.

Di sisi lain, ukuran dimensi yang lebih rendah, seperti 1024 pada model VoyageAI, menawarkan efisiensi yang lebih tinggi. Namun, ini juga membawa risiko "kompresi informasi" atau "kehilangan informasi". Dengan memadatkan representasi semantik ke dalam ruang yang lebih kecil, beberapa detail atau nuansa mungkin hilang. Ruang vektor yang lebih "padat" ini mungkin menyebabkan pasangan kueri-produk yang sangat cocok secara langsung menghasilkan skor kesamaan yang lebih tinggi, tetapi mungkin kurang mampu merepresentasikan hubungan konseptual yang lebih jauh.

2.4.3 Urgensi Penggunaan *Vector Database*

Tantangan selanjutnya adalah bagaimana menyimpan dan mencari data *vector embeddings* yang dimiliki. Basis data relasional, meskipun andal untuk data terstruktur, tidak dirancang untuk melakukan penyimpanan dan pencarian kemiripan vektor *(similarity search)* pada data vektor. Di sinilah urgensi penggunaan *vector database* muncul.



Gambar 2.3 Alur kerja *Vector Database*

Sumber: Exxact Corporation

Cara *vector database* bekerja umumnya terdiri dari dua tahap utama. Gambar 2.3 memberikan gambaran alur kerja *vector database*. Proses dimulai dengan menyimpan teks, seperti deskripsi produk, ke dalam basis data relasional. Dataset produk yang diterima kemudian diubah menjadi bentuk vektor menggunakan *model embeddings* yang dipilih, dan vektor yang baru dibuat ini kemudian disimpan di dalam *vector database*. Ketika sebuah kueri pencarian dimasukkan, kueri tersebut juga diubah menjadi vektor untuk perbandingan. Sistem kemudian mengidentifikasi vektor-vektor yang tersimpan dengan nilai kemiripan tertinggi dan mengembalikannya. Akhirnya, hasil yang relevan ini ditampilkan kepada pengguna .

Pinecone *vector database* yang digunakan dalam penelitian ini, adalah *vector database* yang secara khusus dibuat untuk mengindeks dan melakukan pencarian pada data vektor produk. Vector database ini memanfaatkan algoritma *Approximate Nearest Neighbor* (ANN) yang memungkinkan untukuntuk menemukan vektor yang paling mirip dengan vektor kueri dengan cepat (Carrara dkk., 2022). Dalam konteks penelitian ini, *vector database* berfungsi sebagai teknologi penting yang memungkinkan eksperimen perbandingan. Dengan menyimpan embedding dari OpenAI *text-embedding-3-large* dan VoyageAI *voyage-3-large* dalam indeks terpisah, sistem dapat mengambil hasil pencarian dari kedua model untuk kueri yang sama, sehingga mempermudah evaluasi perbandingan.

2.4.4 *Similarity Search*

Setelah data tekstual, baik kueri pengguna maupun deskripsi produk, berhasil ditransformasikan menjadi representasi vektor, langkah selanjutnya adalah tahap inti dari pencarian semantik, yaitu pencarian kemiripan atau *similarity search*. Proses ini adalah tentang bagaimana sistem mengukur dan membandingkan kedekatan antara vektor kueri dengan seluruh vektor produk yang tersimpan di dalam basis data untuk menemukan kecocokan yang paling relevan secara konteks, bukan sekadar leksikal atau kata kata yang digunakan. Metode perhitungan yang paling umum digunakan dalam pencarian semantik dan juga diimplementasikan dalam penelitian ini adalah *cosine similarity*.

Cosine similarity adalah metrik yang mengukur kosinus sudut antara dua vektor di dalam ruang multi-dimensi. Skor yang dihasilkan berkisar antara -1 hingga 1, di mana skor yang lebih kecil (mendekati 1) menandakan sudut yang lebih kecil dan mengindikasikan kesamaan semantik yang lebih besar. Keunggulan utama metrik ini adalah ia tidak bergantung pada panjang vektor, melainkan hanya pada orientasi (arah) vektor tersebut (Widianto dkk., 2024). Hal ini membuatnya efektif untuk membandingkan kemiripan makna teks, terlepas dari panjang kalimat atau jumlah kata yang digunakan. Secara matematis, kesamaan ini dihitung dengan rumus:

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

Dalam persamaan (2.1, pembilang $\mathrm{A} \cdot \mathrm{B}$ adalah *dot product* dari vektor A dan B, yang mengukur seberapa searah kedua vektor tersebut. Sementara itu, penyebutnya $\|\mathrm{A}\|\|\mathrm{B}\|$, adalah hasil perkalian magnitudo (panjang) dari masing-masing vektor. Dengan membagi *dot product* dengan perkalian magnitudo, kita melakukan normalisasi yang menghilangkan pengaruh panjang dan hanya menyisakan orientasi arah. Dengan demikian, kesamaan yang diukur tidak lagi bergantung pada jumlah kata atau panjang dokumen, melainkan murni berfokus pada orientasi atau arah vektor yang merepresentasikan kesamaan makna. Hasil dari perhitungan inilah yang memberikan nilai antara -1 (berlawanan arah), 0 (tidak berhubungan/ortogonal), hingga 1 (arah identik), yang secara langsung merepresentasikan kesamaan kosinus antara kedua vektor.

Melakukan *similarity search* pada data vektor merupakan tantangan komputasi yang tidak dapat ditangani dengan baik oleh basis data relasional tradisional. Oleh karena itu, penelitian ini memanfaatkan Pinecone *vector database*, yang secara khusus dirancang untuk menyimpan, mengindeks, dan melakukan *similarity search* pada data vektor. Pinecone menggunakan salah

satu algoritma seperti *Approximate Nearest Neighbor* (ANN) untuk menemukan vektor-vektor yang paling mirip dengan sangat cepat, tanpa harus membandingkan vektor kueri dengan setiap vektor di dalam database satu per satu.

Dalam alur kerja sistem yang dikembangkan, ketika pengguna memasukkan kueri, teks tersebut diubah menjadi *query embedding*. Vektor kueri ini kemudian digunakan untuk melakukan *similarity search* di dalam indeks Pinecone. Proses ini menghasilkan daftar ID produk yang telah diurutkan berdasarkan *similarity score* dari tertinggi ke terendah. Selanjutnya, ID produk yang terurut ini digunakan untuk mengambil data produk secara lengkap—seperti nama, harga, dan gambar—dari basis data Supabase PostgreSQL. Daftar produk yang relevan inilah yang kemudian disajikan kepada pengguna sebagai hasil akhir pencarian.

2.5 Supabase PostgreSQL

Pemanfaatan *vector database* seperti Pinecone, sebagaimana yang telah dijelaskan, tidak serta-merta menggantikan peran basis data relasional konvensional dalam sebuah arsitektur sistem. Sebaliknya, kedua teknologi ini bersifat komplementer; masing-masing memiliki fungsi spesifik yang saling melengkapi untuk membangun sistem pencarian yang efisien dan fungsional. Sementara *vector database* digunakan untuk menangani pencarian kemiripan pada data vektor, basis data relasional tetap menjadi komponen penting untuk mengelola data terstruktur seperti metadata produk dan informasi pengguna. Dalam penelitian ini, peran basis data relasional ini dijalankan oleh Supabase PostgreSQL.

Supabase adalah sebuah *platform open source* yang menyediakan *tools* untuk pengembangan aplikasi. Platform yang didirikan pada tahun 2020 ini menawarkan berbagai layanan *backend*, termasuk autentikasi, basis data, penyimpanan file, API yang dibuat secara otomatis, serta *edge functions*. Supabase secara luas dipasarkan sebagai alternatif *open-source* untuk Firebase dan menyediakan pustaka klien (client library) yang kompatibel dengan JavaScript, Python, dan Dart (Supabase, 2025).

Dikarenakan fondasinya adalah PostgreSQL, basis data ini bersifat relasional, yang memungkinkan data untuk diorganisasikan ke dalam tabel-tabel yang saling terhubung melalui relasi yang telah didefinisikan, dengan item di dalamnya disusun dalam format baris dan kolom. Saat mendefinisikan sebuah tabel, setiap kolom wajib memiliki tipe data yang telah ditentukan, di mana Supabase mendukung tipe data bawaan maupun tipe data kustom yang dapat dibuat sesuai kebutuhan.

2.6 Prompt Engineering

Pada penelitian ini, terdapat tantangan yang muncul dikarenakan dataset Amazon Products Sales Dataset 2023 (Parab, 2023) yang digunakan tidak menyertakan kolom deskripsi produk. Ketiadaan deskripsi ini merupakan sebuah keterbatasan yang signifikan, sebab *model embedding* bergantung pada dataset produk yang mendalam untuk dapat lebih menangkap makna dari setiap produk secara akurat.

Untuk mengatasi keterbatasan ini, penelitian mengadopsi pendekatan pra-pemrosesan *(pre-processing)* dengan men-*generate* deskripsi produk menggunakan *Large Language Models* (LLM) dari OpenAI. Di sinilah *prompt engineering* menjadi solusi untuk memastikan dataset memiliki informasi yang lebih mendalam terhadap produk. *Prompt engineering* adalah cara menyusun pertanyaan atau perintah yang sangat spesifik kepada AI untuk mendapatkan jawaban yang paling tepat. Alih-alih sekadar meminta model untuk "buatkan deskripsi", penelitian ini mengembangkan sebuah strategi prompt yang terstruktur. Teknik ini penting untuk memastikan luaran yang dihasilkan oleh model *Generative AI* relevan sesuai dengan keinginan. Seperti yang akan diuraikan pada Bab 4, pendekatan ini melibatkan beberapa strategi utama. Pertama, model diberikan peran atau persona spesifik, yaitu sebagai "Pakar Pemasaran Produk dan SEO Semantik". Kedua, diberikan instruksi format yang ketat, antara lain menghasilkan output dalam bentuk paragraf naratif tunggal dan secara eksplisit melarang penggunaan *bullet points*. Ketiga, *prompt* tersebut secara eksplisit mendorong elaborasi kontekstual yang mendalam, seperti instruksi untuk mengubah fitur menjadi manfaat dan menghubungkannya dengan konteks penggunaan serta gaya hidup target pengguna.

Liu dkk. (2023) menggarisbawahi bahwa efektivitas *prompting* sangat ditentukan oleh kejelasan prompt, karena hal tersebut berdampak langsung pada kualitas respons yang diberikan oleh model AI (X. Liu dkk., 2023). Dengan mengontrol input secara presisi melalui rekayasa prompt, LLM diarahkan untuk menghasilkan deskripsi produk yang deskriptif dan mendalam, sehingga menyediakan data sumber yang optimal untuk diproses oleh *model embeddings*.

2.7 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**
Enhancing knowledge retrieval with in-context learning and semantic search through generative AI (Ghali dkk., 2025)	*Knowledge retrieval* pengetahuan umum	*Generative Text Retrieval* (GTR) & GTR-T	Analisis kuantitatif kinerja sistem *knowledge retrieval*
Enhancing semantic search using ontologies: A hybrid information retrieval approach for industrial text (Naqvi dkk., 2025)	Doku men teks industri	*Hybrid Semantic* (BERT) + Ontology	Analisis kuantitatif kinerja *information retrieval*
FinMTEB: Finance Massive Text Embedding Benchmark (Tang & Yang, 2025)	Sektor keuangan/ *finance* (FinMTEB)	Benchmark *embedding* (FinMTEB) & model Fin-E5	analisis kuantitatif (nDCG, MRR, dll.)
Enhancing Automated Medical Coding: Evaluating Embedding Models for ICD-10-CM Code Mapping (Klotzman, 2024)	Medis	*Embedding* untuk pemetaan kode ICDM-10 (medis)	Kuantitatif: akurasi, *Chi-Square Test*
Human Evaluation Experiment of Legal Information Retrieval Methods (Novotná, 2021)		*Latent Dirichlet Allocation* (LDA), *Non-negative Matrix Factorization* (NMF), doc2vec	Melakukan evaluasi kuantitatif model topik (LDA, NMF, doc2vec) sekaligus penilaian kualitatif oleh ahli hukum terhadap relevansi retrieval tiap metode.

Query2Prod2Vec Grounded Word Embeddings for eCommerce (Bianchi dkk., 2021)	*E-commerce*	*Embedding e-commerce (lexical analogies)*	Kuantitatif: *Hit Rate* (HR) pada tugas analogi; studi kualitatif kecil sebagai pelengkap

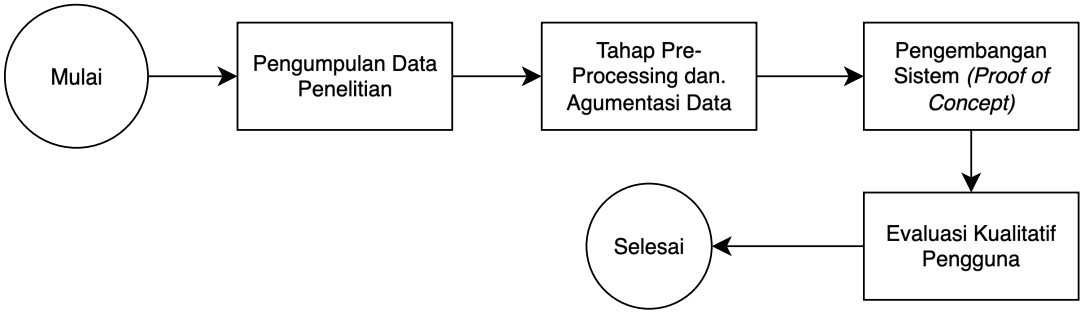
Analisis terhadap penelitian-penelitian terdahulu pada Tabel 2.1 menunjukkan sebuah pola yang kompleks namun jelas. Di satu sisi, terdapat fokus yang kuat pada evaluasi kuantitatif, di mana pemilihan metodologi selaras dengan objektif penelitian. Studi oleh Klotzman (2024), misalnya, yang bertujuan mengukur akurasi model OpenAI dan VoyageAI dalam domain medis, secara tepat menggunakan evaluasi kuantitatif seperti uji statistik. Di sisi lain, penelitian seperti yang dilakukan oleh Novotna (2024) menunjukkan bahwa evaluasi kualitatif oleh ahli memang digunakan dalam literatur *information retrieval*. Namun, pendekatannya diterapkan dalam domain hukum yang terstruktur.

Pola ini juga ditemui pada penelitian yang dilakukan dalam domain *e-commerce*. Studi oleh Bianchi et al. (2021), meskipun berkonteks *e-commerce*, memiliki objektif utama untuk mengevaluasi performa pada tugas analogi leksikal yang terprogram, dengan indikator kuantitatif sebagai tolok ukur utamanya dan studi kualitatif hanya sebagai pelengkap. Kombinasi temuan ini secara jelas menyoroti adanya kesenjangan yang spesifik: kurangnya penelitian yang menjadikan persepsi pengguna kualitatif sebagai objektif dan metodologi evaluasi inti untuk melakukan esplorasi perbandingan model-model *embedding* dalam konteks e-*commerce.*

Penelitian ini berperan untuk mengisi kesenjangan tersebut. Dengan menetapkan pemahaman persepsi pengguna sebagai objektif inti, maka pemilihan metodologi evaluasi kualitatif berbasis persepsi pengguna menjadi sebuah indikator penting, bukan sekadar alternatif. Pendekatan ini memungkinkan studi ini untuk memahami mengapa pengguna merasakan relevansi, sebuah aspek yang penting namun secara sifat berada di luar cakupan indikator kuantitatif.

BAB III
METODOLOGI PENELITIAN

Penelitian ini menggunakan tahap yang sistematis untuk mengevaluasi efektivitas *model embeddding* dalam konteks pencarian semantik *e-commerce*. Metodologi ini mencakup serangkaian proses yang saling terkait, mulai dari tahap persiapan data, pengembangan sistem sebagai instrumen pengujian, hingga evaluasi kualitatif dengan pengguna akhir. Untuk memberikan gambaran yang jelas mengenai keseluruhan alur kerja, penelitian ini dirancang mengikuti diagram alir pada Gambar 3.1.



Gambar 3.1 Alur Penelitian

Penelitian ini menggunakan metodologi *Extreme Programming* (XP) sebagai kerangka pengembangan perangkat lunak untuk implementasi sistem *e-commerce* dengan pencarian semantik. Pendekatan ini melibatkan serangkaian tahapan sistematis, termasuk *Planning, Design, Coding,* dan *Testing.* untuk memahami kebutuhan sistem secara menyeluruh dan memastikan sistem dapat berjalan dengan baik. XP memberikan fleksibilitas dalam implementasi, termasuk dukungan terhadap *solo programming* yang memungkinkan pengembangan efisien dalam tim kecil atau bahkan *individual developer.* Penelitian ini bertujuan untuk mengembangkan sistem *e-commerce* yang mengintegrasikan teknologi pencarian semantik dan melakukan evaluasi perbandingan terhadap preferensi pengguna dalam penggunaan dua *model embeddings* yang berbeda.

Setiap tahapan dalam diagram alir tersebut akan diuraikan secara rinci pada sub-bab berikut untuk menjelaskan proses penelitian secara komprehensif.

3.1 Pengumpulan dan Persiapan Data Penelitian

3.1.1 Sumber Dataset

Penelitian ini menggunakan Amazon Product Sales Dataset 2023 (Parab, 2023). Dataset ini adalah dataset publik yang tersedia di Kaggle. Dataset ini dipilih karena data yang disajikan adalah data produk asli dari Amazon E-Commerce. Namun, dataset ini memiliki keterbatasan

yaitu tidak menyertakan deskripsi dari produk. Oleh karena itu, dilakukan augmentasi data pada dataset ini yang akan dijelaskan pada subbab selanjutnya.

3.2 Metode Augmentasi Data Deskripsi Produk (*Pre-Processing*)

Untuk mengatasi keterbatasan dataset, penelitian ini mengadopsi pendekatan pra-pemrosesan dengan men-generate deskripsi produk menggunakan Large Language Model (LLM) dari OpenAI, yaitu model GPT-4o. Agar deskripsi yang dihasilkan memiliki kedalaman makna yang optimal untuk pencarian semantik, dirancang sebuah strategi prompt engineering yang spesifik.

Pendekatan utama yang digunakan adalah persona-based prompting, di mana LLM diberikan peran sebagai "a highly experienced Product Marketing and Semantic SEO expert". Persona ini bertujuan mengarahkan LLM untuk menghasilkan teks yang persuasif, berorientasi pada manfaat, dan kaya akan hubungan konseptual antar kata.

```
You are a highly experienced Product Marketing and Semantic SEO expert for a
premium international e-commerce platform.
    Your task is to write a very detailed, rich, and specific product
description in fluent **English**. This description must provide a deep,
contextual understanding of the product, intended for semantic search research.

    **Product Data:**
    - Product Name: {row['name']}
    - Category Path: {row['categories']}
    - Price: {price_formatted}

    **Writing Instructions:**
    1.  **Language:** The entire output must be in **English**.
    2.  **Narrative Format:** Describe the product in a single, flowing,
information-rich narrative paragraph. **DO NOT USE bullet points or list
formats.**
    3.  **Deep Elaboration:** Do not just list features from the product name.
Elaborate on every detail, turning it into a tangible benefit. For example, if
the name includes 'slim fit', describe how this cut creates a modern, tailored
silhouette. If it mentions a material like 'katun' (cotton), explain its
comfort and feel (e.g., 'breathable, soft on the skin, and perfect for tropical
climates').
    4.  **Usage Context:** Imagine and describe the target user (e.g., a young
professional, a university student, a fashion enthusiast) and the ideal
situations or events for using this product (e.g., for a formal event, office
wear, a wedding reception, or a casual weekend outing).
```

```
    5.  **Lifestyle Connection:** Create a story or connect the product to a
specific lifestyle, aspiration, or value. Make the product feel like more than
just an item.
    6.  **Specificity and Detail:** Ensure the description is highly specific,
contextually mentioning details from the product name and category to enrich
its semantic meaning.
    7.  **Ideal Length:** Aim for a description between 80 and 150 words to
ensure informational depth.
    8.  **Direct Output:** Provide only the description text itself, without
any introductory or concluding remarks from you as the AI.
```

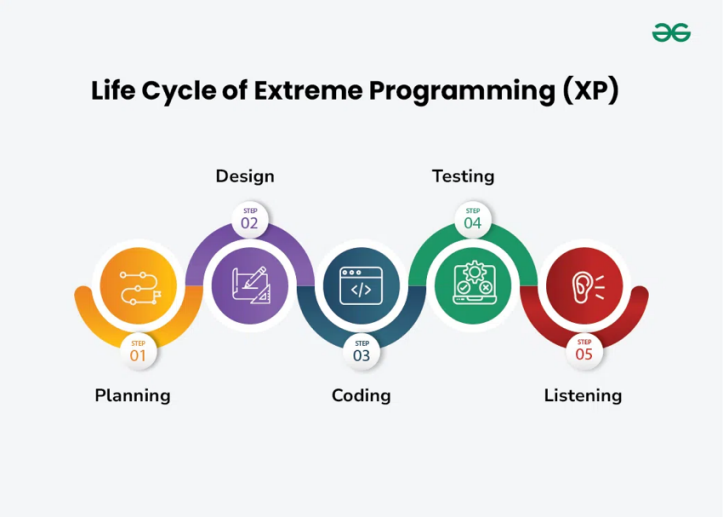
Gambar 3. 2 *Prompt* untuk Augmentasi Data

Prompt juga diperkaya dengan instruksi spesifik untuk melakukan elaborasi kontekstual, seperti *"Deep Elaboration", "Usage Context",* dan *"Lifestyle Connection"* yang secara eksplisit mendorong model untuk mengubah fitur menjadi manfaat dan membangun narasi di sekitar produk. Detail prompt yang digunakan dapat dilihat pada Gambar 3. 2

Proses generasi deskripsi ini dieksekusi secara terprogram melalui sebuah skrip Python. Untuk setiap baris produk dalam dataset, skrip tersebut secara dinamis menyusun prompt dengan memasukkan data produk (nama, kategori, harga), lalu mengirimkannya ke API OpenAI untuk menghasilkan deskripsi yang sesuai. Hasil dari proses ini adalah sebuah file CSV baru yang telah diperkaya dengan kolom deskripsi produk yang siap digunakan untuk tahap selanjutnya.

3.3 Pengembangan dan Evaluasi Sistem Pengujian

Penelitian ini menggunakan metodologi Extreme Programming (XP) sebagai kerangkapengembangan perangkat lunak untuk implementasi sistem e-commerce dengan pencariansemantik. Pendekatan ini melibatkan serangkaian tahapan sistematis, termasuk *Planning, Design, Coding, Testing,* dan *Listening*, untuk memahami kebutuhan sistem secara menyeluruh dan memastikan sistem dapat berjalan dengan baik. XP memberikan fleksibilitas dalam implementasi, termasuk dukungan terhadap solo programming yang memungkinkan pengembangan efisien dalam tim kecil atau bahkan individual developer. Penelitian ini bertujuan untuk mengembangkan sistem e-commerce yang mengintegrasikan teknologi pencarian semantik dan melakukan evaluasi perbandingan terhadap preferensi pengguna dalam penggunaan dua model embeddings yang berbeda.



Gambar 3.3 *Extreme Programming Life Cycle*

Sumber: *GeeksforGeeks, 2025*

Gambar 3.3 dalam penelitian ini mengacu pada alur metodologi XP, yang menjadi indikator utama dalam pengembangan sistem pada penelitian ini. Selain itu, penelitian ini melibatkan implementasi dua *model embeddings* yang berbeda dalam fitur pencarian semantik serta melakukan evaluasi perbandingan melalui pengujian pengguna *(blind A/B testing)* untuk menganalisis preferensi dan efektivitas masing-masing model. Dengan demikian, penelitian ini memberikan kontribusi dalam pemahaman praktis tentang penerapan XP dalam pengembangan sistem *e-commerce* dengan teknologi pencarian semantik, sekaligus menghasilkan wawasan empiris mengenai preferensi pengguna terhadap performa *model embeddings* dalam konteks pencarian produk *e-commerce.*

3.3.1 Fase Planning

Planning Phase (fase perencanaan) dalam siklus *Extreme Programming* (XP) penelitian ini berfokus pada penetapan lingkup dan definisi kebutuhan yang diperlukan sistem *e-commerce* dengan fitur pencarian semantik yang akan dikembangkan. Kegiatan utama dalam fase ini adalah analisis kebutuhan sistem, di mana peneliti melakukan identifikasi dan perumusan kebutuhan fungsional, non-fungsional, serta kebutuhan tampilan antarmuka pengguna. Proses analisis ini mendasarkan diri pada temuan-temuan pada Bab II, observasi tantangan pencarian *e-commerce*, dan penyesuaiannya dengan tujuan evaluasi *model embeddings*.

A. Analisis Kebutuhan Sistem

Sistem yang dikembangkan adalah aplikasi *e-commerce* berbasis website yang mengintegrasikan fitur pencarian semantik. Aplikasi ini dibangun menggunakan *framework* Next.js, dengan mengintegrasikan model OpenAI *text-embedding-3-large* dan VoyageAI *voyage-3-large* sebagai model *embeddings* untuk mengubah data produk *e-commerce* menjadi representasi nilai vektor. Pinecone *vector database* digunakan untuk penyimpanan dan pencarian data produk dalam bentuk vektor, sementara Supabase PostgreSQL berfungsi sebagai basis data relasional untuk mendukung penyimpanan metadata produk dan pengambilan detail produk untuk antarmuka pengguna. Detail dari kebutuhan sistem yang dikembangkan adalah sebagai berikut.

a. Kebutuhan Fungsional

1. Manajemen Katalog Produk
    a. Menampilkan Katalog Produk: Pengguna harus dapat menjelajahi dan melihat daftar produk yang tersedia dalam katalog secara terstruktur. Ini mencakup halaman utama dan katalog yang menyajikan data produk
    b. Menampilkan Detail Produk: Pengguna harus dapat mengakses informasi lengkap mengenai satu produk spesifik, seperti deskripsi produk, harga, gambar, dan kategori secara rinci.
2. Pencarian Semantik

    Pencarian semantik ini adalah fitur inti yang dievaluasi. Sistem wajib menyediakan fitur pencarian yang mampu melampaui pencocokan kata kunci literal. atau leksikal. Fitur ini harus dapat:

    a. Memahami konteks dan menginterpretasikan intensi pengguna bahkan dari kueri yang ambigu atau menggunakan sinonim.
    b. Mengintegrasikan *model embeddings* (OpenAI *text-embedding-3-large* dan VoyageAI *voyage-3-large*) untuk mengubah kueri pengguna dan deskripsi produk menjadi nilai vektor.
    c. Menampilkan daftar produk yang relevan secara semantik berdasarkan perhitungan kemiripan vektor antara kueri dan produk di dalam *vector database*.
3. Kebutuhan Non-fungsional

    Kebutuhan non-fungsional yang dibutuhkan adalah sistem harus mengimplementasikan mekanisme pencatatan *(logging)* secara mendetail. *Log* ini harus bisa mencatat informasi penting seperti proses transformasi data menjadi

embedding saat proses *data seeding* dan input kueri, *similarity score* yang dihasilkan untuk tiap kueri-produk, dan durasi API calls.

4. Kebutuhan Tampilan Antarmuka

Desain tampilan antarmuka pengguna (UI) harus mendukung tujuan utama penelitian yaitu evaluasi fitur pencarian semantik, dengan tetap memperhatikan kemudahan penggunaan dan kejelasan informasi. Berikut adalah kebutuhan dari tampilan antarmuka yang dikembangkan:

a. Halaman Utama

Halaman utama menjadi gerbang utama di mana pertama kali pengguna mengakses *e-commerce.* Pada halaman ini pengguna dapat melihat beberapa produk yang ditampilkan, komponen *navigation bar* untuk melakukan navigasi menuju fitur lain seperti halaman katalog untuk melakukan pencarian semantik.

b. Halaman Katalog

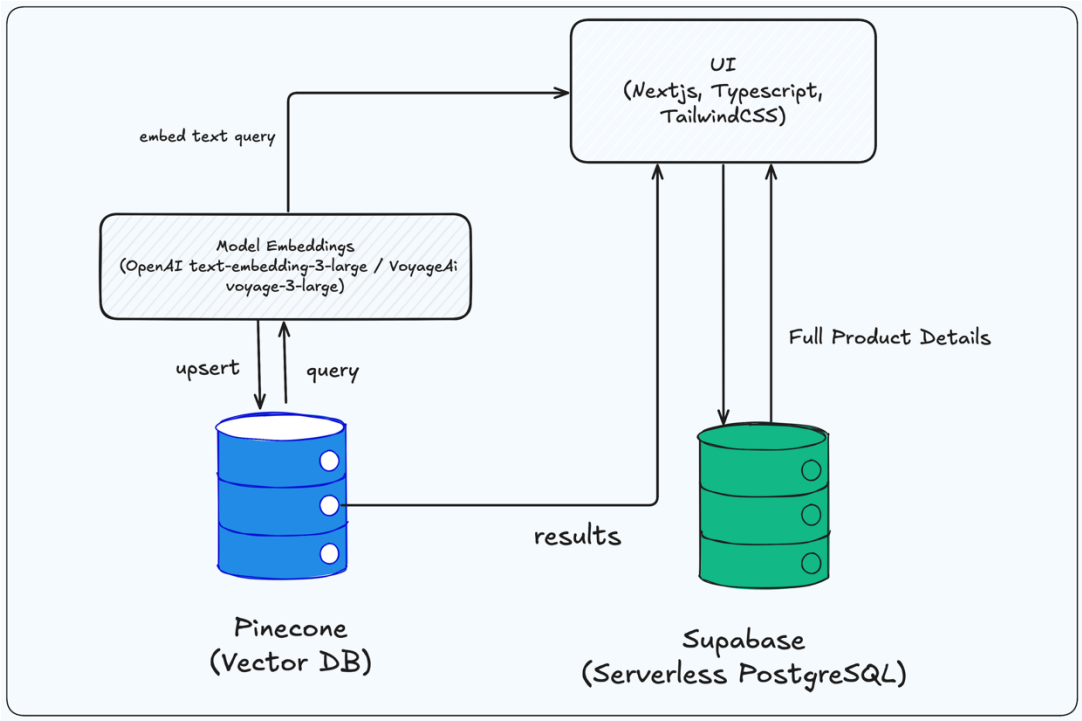
Halaman ini berfungsi sebagai area utama untuk implementasi dan evaluasi fitur pencarian semantik. Antarmuka pengguna (UI) pada halaman ini dirancang untuk memiliki beberapa komponen. Pertama, disediakan sebuah input pencarian yang jelas dan mudah diakses untuk pengguna. Selanjutnya, hasil pencarian produk akan ditampilkan dalam format *grid* yang terstruktur, di mana setiap produk menyajikan informasi utama seperti gambar, nama, dan harga. Pengguna juga diberikan kemampuan untuk mengklik produk individual untuk melihat detail yang lebih spesifik. Sebagai bagian penting dari proses evaluasi, diimplementasikan sebuah menu *dropdown* untuk mengganti *model embeddings*. Mekanisme ini menggunakan pendekatan blind assessment, di mana Model A (OpenAI *text-embedding-3-large*) dan Model B (VoyageAI *voyage-3-large*) dapat dialihkan tanpa sepengetahuan pengguna, untuk memastikan objektif dalam penilaian performa masing-masing model tanpa adanya bias.

3.3.2 *Design Phase* (Fase Desain)

Pada fase ini, dijelaskan secara mendetail mengenai desain arsitektur sisem yang dikembangkan. *Design phase* ini merupakan tahap di mana dilakukan perancangan konsep-konsep dasar yang akan merepresentasikan entitas data dan interaksi antar pengguna dengan sistem. Beberapa konsep dasar yang dibuat pada fase ini adalah Desain arsitektur sistem, *entity relationship diagram* (ERD) dan *use case diagram*.

A. Desain Arsitektur Sistem

Pada penelitian ini, sistem diimplementasikan untuk melakukan pencarian semantik, yang mana prosenya adalah kueri pencarian yang ditulis pengguna akan di-*embed* atau ditransformasikan menjadi nilai vektor menggunakan *model embeddings* (OpenAI *text-embedding-3-large* & VoyageAI *voyage-3-large*) kemudian dilakukan *cosine similariy search* dengan data produk di Pinecone *vector database* yang telah ditransformasikan menjadi nilai vektor yang kemudian dikembalikan hasil data produk yang relevan berdasarkan nilai *similarity score* tertinggi ke terendah. Berikut adalah detail gambaran umum dari sistem yang diimplementasikan.



Gambar 3. 4 *Flow* Arsitektur Sistem.

Gambar 3. 4 mengilustrasikan arsitektur sistem yang dikembangkan. Arsitektur ini terdiri dari beberapa komponen utama yang saling berinteraksi:

a. Antarmuka Pengguna (UI): Dikembangkan menggunakan Next.js, TypeScript, dan TailwindCSS, yang berfungsi sebagai interaksi utama bagi pengguna. Pengguna memasukkan kueri teks pencarian melalui UI, yang kemudian dikirim ke komponen *model embeddings*. UI juga menerima dan menampilkan hasil pencarian dari Pinecone serta detail produk lengkap dari Supabase.
b. *Model Embeddings* (OpenAI *text-embedding-3-large* & VoyageAI *voyage-3-large*): Komponen ini bertugas mentransformasikan kueri teks dari UI menjadi *nilai vektor*. Selain itu, pada tahap *data seeding,* komponen ini juga digunakan untuk membuat nilai

vektor dari deskripsi produk yang kemudian disimpan (*upsert*) ke Pinecone. Saat pencarian, vektor kueri digunakan untuk melakukan kueri ke Pinecone.

c. Pinecone *Vector Database*: Pinecone berperan sebagai tempat penyimpanan semua vektor embedding produk. Pinecone menerima vektor kueri dari komponen *model embeddings* dan melakukan pencarian kemiripan untuk menghasilkan daftar ID produk dan skor relevansinya, yang kemudian dikirim ke UI antarmuka.
d. Supabase (Serverless PostgreSQL): Supabase menyimpan semua metadata produk non-vektor, seperti nama, deskripsi teks asli, harga, kategori, dan URL gambar. Ketika UI memerlukan detail lengkap dari produk yang relevan (berdasarkan ID dari Pinecone), informasi ini diambil dari Supabase.

Alur kerja pencarian semantik dalam sistem yang diimplementasikan ini melibatkan serangkaian langkah, mulai dari input pengguna hingga penampilan hasil. Proses diawali ketika pengguna menginputkan kueri pencarian melalui kolom pencarian yang tersedia di antarmuka pengguna di halaman katalog produk. Komponen *frontend* kemudian menangkap input tersebut dan mengirimkannya sebagai permintaan ke backend sistem melalui *API calls*. Endpoint API pada server Next.js selanjutnya menerima kueri pencarian ini untuk pemrosesan lebih lanjut.

Setelah menerima kueri, dimulai tahap transformasi vektor. Kueri teks diteruskan ke layanan API dari *model embedding* yang aktif (OpenAI *text-embedding-3-large* atau VoyageAI *voyage-3-large*) untuk diubah menjadi nilai vektor. Vektor kueri yang dihasilkan ini kemudian digunakan untuk melakukan *similarity search* di Pinecone Vector Database. Pinecone, yang telah menyimpan vektor representasi dari seluruh katalog produk, menghitung kemiripan menggunakan algoritma *cosine similarity* dan mengembalikan daftar ID produk beserta skor kemiripan (*similarity*), yang sudah diurutkan dari relevansi tertinggi ke terendah.

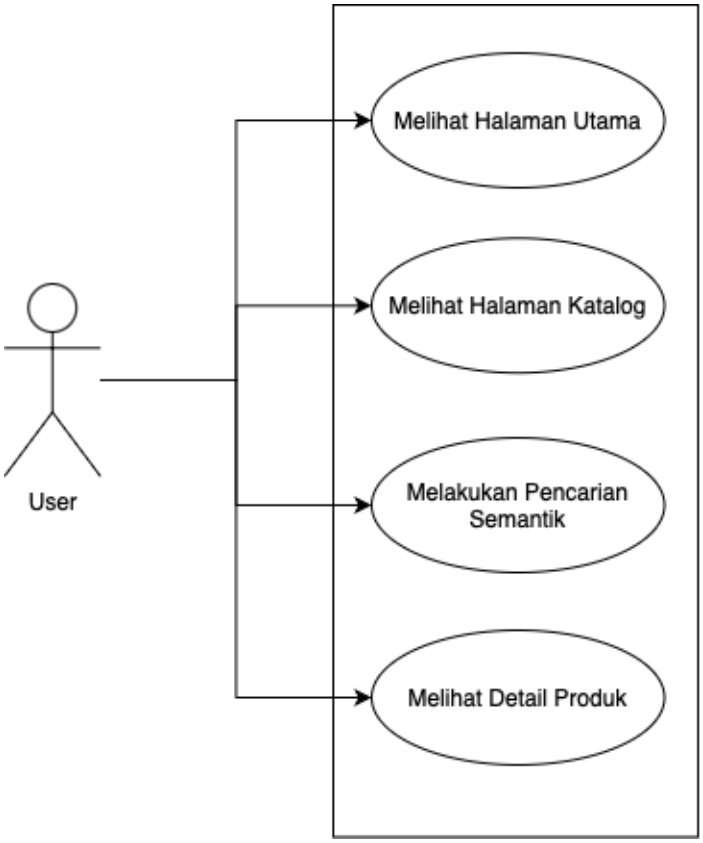
Tahap berikutnya adalah pengambilan data produk. Berdasarkan daftar ID produk yang diterima dari Pinecone, diambil metadata produk secara lengkap—seperti nama, deskripsi, harga, URL gambar, dan kategori—dari basis data Supabase PostgreSQL. Data vektor (ID dan skor) dari Pinecone dan data metadata dari Supabase kemudian diintegrasikan dan disusun oleh menjadi satu respons API terstruktur dalam bentuk JSON.

Akhirnya, pada fase output, respons API terstruktur yang berisi informasi produk lengkap beserta skor relevansinya ini dikirim kembali ke *frontend*. Komponen *frontend* menerima data tersebut dan merendernya sebagai daftar produk yang relevan dengan kueri awal pengguna. Pengguna kemudian dapat melihat daftar produk yang telah diurutkan berdasarkan relevansi

semantik, yang memungkinkan mereka untuk menemukan produk yang paling sesuai dengan kebutuhan atau intensi yang pengguna maksud.

B. Use Case Diagram

Untuk memberi gambaran sistem, digunakan pemodelan *Use Case Diagram*. Diagram use case dari sistem ini memiliki satu aktor dan empat aktivitas. Aktivitas yang telah didefinisikan dan dibutuhkan oleh sistem dapat dilihat pada Gambar 3.1 berikut.



Gambar 3. 5 *Use case diagram*

Pada Gambar 3. 5, setiap *use case* diuraikan mulai dari aktor yang terlibat dan deskripsi dari *use case* terkait pada Tabel 3. 1 Penjelasan *Use Case Diagram* berikut:

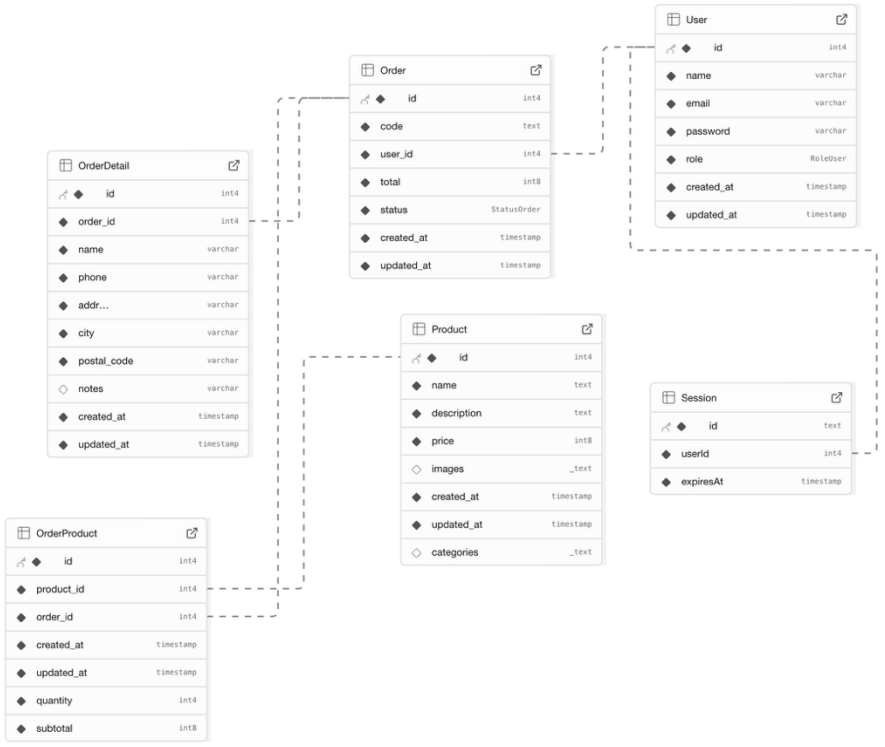
Tabel 3. 1 Penjelasan *Use Case Diagram*

No.	Aktor	Use Case	Deskripsi
1	*User*	Melihat Halaman Utama	Pengguna mengakses dan melihat tampilan awal sistem *e-commerce*. Halaman utama menyajikan informasi produk, dan navigasi ke fitur katalog.
2	*User*	Melihat Halaman Katalog	Pengguna mengakses dan menjelajahi keseluruhan daftar produk yang tersedia.

3	*User*	Melakukan Pencarian Semantik	Pengguna dalam memasukkan kueri pencarian untuk menemukan produk. Sistem kemudian memproses kueri tersebut menggunakan mekanisme pencarian semantik untuk menampilkan produk yang relevan.
4	*User*	Melihat Detail Produk	Interaksi di mana pengguna memilih sebuah produk (dari katalog atau hasil pencarian) untuk mengakses dan melihat informasi yang lebih mendalam seperti deskripsi lengkap, harga, dan spesifikasi.

C. Desain Entity Relationship Diagram (ERD)

Entity Relationship Diagram memberikan gambaran hubungan antara entitas dalam basis data. ERD sistem ini meliputi beberapa tabel seperti *users, orders, orders_detail, orders_product, products,* dan *categories*.



Gambar 3.6 *Entity Relationship Diagram*

Berikut penjelasan detail mengenai ERD pada gambar:

a. Tabel users

Menyimpan informasi tentang pengguna (user).

1. **id**: ID unik pengguna (*Primary Key*).
2. **name**: Nama pengguna.
3. **email**: Alamat email pengguna (unik).
4. **password**: Kata sandi pengguna (*hashed*).
5. **role**: Peran pengguna dalam sistem (*customer* & *superadmin*). digunakan ENUM atau tipe data kustom.
6. **created_at**: Timestamp saat pengguna dibuat.
7. **updated_at**: Timestamp saat pengguna terakhir diperbarui.

b. Tabel Category:

1. **Id**: ID unik untuk setiap kategori(Primary Key).
2. **category_id**: ID kategori produk (Foreign Key ke tabel Category)
3. **name**: Nama produk.
4. **created_at**: Timestamp saat kategori dibuat.
5. **updated_at**: Timestamp saat kategori terakhir diperbaharui.

c. Tabel products:

Menyimpan informasi detail mengenai setiap produk yang tersedia di e-commerce.

1. **Id**: ID unik untuk setiap kategori(Primary Key).
2. **name** (text): Nama produk.
3. **description** (text): Deskripsi lengkap mengenai produk.
4. **price** (int8): Harga produk. Tipe data int8 (bigint) digunakan untuk nilai numerik besar.
5. **images** (text[]): Array atau daftar URL atau path supabase ke gambar-gambar produk.
6. **created_at** (timestamp): Timestamp saat data produk pertama kali ditambahkan.
7. **updated_at** (timestamp): Timestamp saat data produk terakhir kali diperbarui.
8. **categories** (_text): Array atau daftar kategori yang terkait dengan produk.

d. Tabel orders:

1. **id** (integer): ID unik untuk setiap pesanan (*Primary Key*).
2. **code** (varchar): kode unik pesanan
3. **user_id** (integer): ID pengguna yang membuat pesanan (*foreign key* yang merujuk ke kolom id pada tabel users).
4. **total:** Total harga pesanan.
5. **status:** Status pesanan.
6. **created_at:** Timestamp saat pesanan dibuat.
7. **updated_at:** Timestamp saat pesanan terakhir diperbarui.

e. Tabel orders_details:

1. **id** (Integer): ID unik untuk setiap detail pesanan (Primary Key).
2. **order_id** (integer): ID pesanan yang terkait dengan detail ini (*Foreign key* yang merujuk ke kolom ide pada tabel orders).

3. **name** (varchar): Nama penerima pesanan.
4. **phone** (varchar): Nomor telepon pesanan.
5. **address** (varchar): Alamat pengiriman pesanan.
6. **city** (Varchar): Kota tujuan pengiriman.
7. **postal_code** (Varchar): Kode pos pengiriman.
8. **notes** (Varchar): Catatan tambahan untuk pesanan.
9. **created_at:** Timestamp saat pesanan dibuat.
10. **updated_at:** Timestamp saat pesanan terakhir diperbarui.

f. Tabel orders_product:

1. **id** (Integer): ID unik untuk setiap item produk dalam pesanan (Primary Key).
2. **product_id** (Integer): ID produk yang terkait (Foreign Key yang merujuk ke kolom id dalam tabel products).
3. **order_id** (Integer): ID pesanan yang berisi produk ini (Foreign Key yang merujuk ke kolom id dalam tabel orders).
4. **subtotal** (Integer): Subtotal harga untuk jumlah item produk tertentu dalam pesanan.
5. **quantity** (Integer): Jumlah produk dalam pesanan.
6. **created_at:** Timestamp saat pesanan dibuat.
7. **updated_at:** Timestamp saat pesanan terakhir diperbarui.

g. Tabel Session

1. **ID:** id unik untuk setiap sesi (Primary Key).
2. **userId:** ID pengguna yang terkait dengan sesi (Foreign Key ke tabel User)
3. **expiresAt:** Waktu kedaluwarsa sesi.

Penjelasan Relasi:

a. User dan Order

Satu *user* dapat memiliki banyak Order (relasi satu-ke-banyak). Ini diimplementasikan dengan user_id di tabel Order sebagai foreign key yang merujuk ke id di tabel User.

b. Order dan OrderDetail

Setiap Order memiliki satu OrderDetail (relasi satu-ke-satu, diimplementasikan seolah satu-ke-banyak dari sisi Order ke OrderDetail jika order_id di OrderDetail unik atau ada constraint). order_id di OrderDetail merujuk ke id di Order.

c. Order dan OrderProduct

Sebuah Order dapat terdiri dari banyak OrderProduct (relasi satu-ke-banyak). order_id di OrderProduct merujuk ke id di Order.

d. Product dan OrderProduct

Sebuah Product dapat muncul di banyak OrderProduct (relasi satu-ke-banyak). product_id di OrderProduct merujuk ke id di Product.

e. User dan Session

Seorang User dapat memiliki banyak Session (relasi satu-ke-banyak). userId di tabel Session merujuk ke id di tabel User.

D. Perancangan Desain Antarmuka Pengguna

Desain antarmuka pengguna (UI) dalam penelitian ini dirancang untuk memenuhi dua tujuan utama: pertama, menyediakan *platform* yang fungsional bagi pengguna untuk melakukan tugas pencarian produk; dan kedua, mengakomodasi proses evaluasi *blind A/B testing*. Mengingat fokus utama penelitian ini adalah pada evaluasi fitur *backend* (yaitu, evaluasi perbandingan *model embeddings*) dan bukan pada desain antarmuka, maka pengembangan UI mengadopsi pendekatan yang pragmatis dan efisien.

Untuk mempercepat proses pengembangan dan memastikan standar desain yang modern dan responsif, penelitian ini memanfaatkan *boilerplate* antarmuka e-commerce yang tersedia di internet. Penggunaan *boilerplate* ini memungkinkan peneliti untuk fokus pada implementasi logika pencarian semantik dan mekanisme *switching* antar model, tanpa harus membangun seluruh komponen visual dari awal. Komponen-komponen standar seperti *layout grid* untuk produk, halaman detail produk, dan navigasi utama diadopsi dari *boilerplate* tersebut.

Meskipun demikian, dilakukan kustomisasi pada komponen-komponen yang relevan dengan tujuan penelitian. Modifikasi utama meliputi:

a. Implementasi Komponen Pencarian

Sebuah kolom pencarian diimplementasikan untuk menangkap kueri pengguna dan memicu pemanggilan API ke backend.

b. Mekanisme Blind A/B Testing

Komponen penting yang dikembangkan secara spesifik untuk penelitian ini adalah sebuah *menu dropdown* yang memungkinkan pengguna untuk beralih antara "Model A" dan "Model B". Komponen ini dirancang untuk memastikan partisipan tidak mengetahui *model embedding* mana yang sedang aktif, sehingga menjaga objektif evaluasi.

c. Tampilan Hasil Dinamis

Antarmuka dimodifikasi untuk dapat me-*render* hasil pencarian yang diterima dari *backend* secara dinamis, menampilkan gambar, nama, dan harga produk sesuai dengan urutan relevansi yang dikembalikan oleh setiap *model embedding*.

Dengan pendekatan ini, dihasilkan sebuah prototipe fungsional yang memiliki tampilan antarmuka yang profesional dan intuitif, sekaligus secara presisi mendukung kebutuhan pengumpulan data untuk evaluasi kualitatif.

E. Metode Database Seeding dan Indexing Vektor

Setelah dilakukan augmentasi data pada dataset, tahap selanjutnya adalah memasukkan data ke dalam basis data sistem. Proses ini dilakukan menggunakan sebuah *seeding script* yang dibuat menggunakan TypeScript. Skrip ini dirancang untuk menjalankan dua tugas utama:

a. Seeding Supabase PostgreSQL: *script* melakukan iterasi pada baris CSV dataset, lalu menyimpan metadata produk seperti ke dalam tabel `Product` di Supabase PostgreSQL via Prisma ORM

b. Indexing Pinecone Vector Database: Untuk setiap produk yang disimpan di Supabase, *script* menggabungkan beberapa atribut teks (nama, kategori, dan deskripsi) menjadi satu input. Teks ini kemudian ditransformasikan menggunakan model embeddings untuk menghasilkan representasi vektor. Untuk mengakomodasi perbandingan A/B, proses ini dilakukan secara paralel untuk kedua model embedding. Vektor dari model OpenAI text-embedding-3-large (3072 dimensi) disimpan di indeks Pinecone ecommerce-3-large, sementara vektor dari model VoyageAI voyage-3-large (1024 dimensi) disimpan di indeks Pinecone ecommerce-voyage-3-large.

F. Desain Eksperimen dan Prosedur Evaluasi

Untuk menjawab rumusan masalah mengenai perbandingan efektivitas dan preferensi pengguna, penelitian ini menggunakan metode evaluasi kualitatif dengan desain *blind A/B testing*.

a. Partisipan

Partisipan dalam penelitian ini dipilih berdasarkan kriteria tertentu agar dapat mewakili pengguna e-commerce yang relevan dalam konteks penelitian. Kriteria tersebut meliputi rentang usia 18-34 tahun serta memiliki pengalaman berbelanja secara daring melalui platform e-commerce. Selain itu, responden harus memiliki pengetahuan atau pengalaman pada salah satu doimain produk yang digunakan dalam penelitian ini, yaitu produk fitness maupun produk skincare, sehingga dapat memberikan penilaian yang bermakna terhadap relevansi hasil pencarian.

Pemilihan hanya dua domain produk ini didasarkan pada tiga pertimbangan utama. Pertama, kategori *fitness* dan *skincare* memiliki jumlah data yang representatif dalam Amazon Product Sales Dataset 2023. Kedua, keduanya merepresentasikan konteks belanja daring yang umum dan populer di Indonesia. Ketiga, kedua domain ini mampu mencerminkan perbedaan pola pencarian: produk fitness biasanya dicari dengan spesifikasi teknis yang jelas, sementara skincare lebih banyak dipilih berdasarkan preferensi personal. Dengan demikian, ruang lingkup penelitian menjadi lebih terfokus tanpa mengurangi keutuhan analisis.

b. Prosedur Evaluasi

Evaluasi penelitian ini dirancang sebagai sebuah asesmen perbandingan buta *(blind comparative assessment)* untuk memastikan tercapainya objektif dan meminimalkan potensi bias dari partisipan. Prosedur evaluasi dilakukan melalui beberapa tahapan sistematis sebagai berikut:

a. Pengenalan Sistem

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

b. Penugasan

Partisipan diminta untuk melakukan serangkaian tugas pencarian produk. Untuk mengevaluasi performa model secara komprehensif di berbagai skenario,

partisipan diinstruksikan untuk merumuskan dan menguji empat kategori kueri. Kategori ini dirancang untuk mencerminkan variasi kompleksitas bahasa pengguna pola pencarian utama dalam e-commerce, yaitu *lookup search* dan *exploratory search*.

Tabel 3. 2 Kriteria Penugasan Kueri

Kategori Kueri	Karakteristik Utama	Contoh	Pola Pencarian
Pendek	Input minimal, sangat spesifik	"*dumbbells*"	*Lookup search*
Sedang (2–3 kata)	Kombinasi istilah sederhana, arah pencarian jelas	"*home gym rowing machine*"	*Lookup search*
Panjang dan Deskriptif	Kalimat ≥ 5 kata, kebutuhan detail & konteks kompleks	"*equipment for building leg muscle* without *taking up too much space*"	*Exploratory search*
Ambigu	Intensi samar, terbuka pada interpretasi beragam	"*something for recovery after workout*"	*Exploratory search*

Detail dari penugasan tersebut adalah sebagai berikut:

1. Kueri Pendek (1-2 kata)

   Digunakan untuk menilai kemampuan model memahami konteks dari input yang sangat minimal. Contoh: *"dumbbells". (lookup search).*

2. Kueri Sedang (2-3 kata)

   Digunakan untuk menguji bagaimana model menangani kombinasi beberapa istilah yang lebih terarah. Contoh: *"home gym rowing machine". (lookup search).*

3. Kueri Panjang dan Deskriptif

   Digunakan untuk mengevaluasi pemahaman konteks yang lebih mendalam terhadap kebutuhan pengguna yang spesifik atau multi-aspek. Contoh:

"equipment for building leg muscle without taking up too much space". (exploratory search).

4. Kueri Ambigu

   Digunakan untuk menilai kemampuan model menghadapi ketidakpastian maksud pengguna. Contoh: *"something for recovery after workout". (exploratory search).*

c. Interaksi A/B

Untuk setiap kueri yang diuji, partisipan melihat hasil pencarian dari kedua *model embedding* tanpa mengetahui model mana yang sedang aktif. Antarmuka pengguna menyediakan sebuah menu dropdown yang dilabeli dengan netral ("Model A" untuk OpenAI dan "Model B" untuk VoyageAI). Mekanisme ini memastikan bahwa penilaian partisipan murni didasarkan pada kualitas hasil pencarian yang ditampilkan.

d. Pengumpulan Data

Setelah berinteraksi dan membandingkan hasil dari kedua model untuk setiap kueri, data kualitatif digali melalui wawancara semi-terstruktur. Fokus utama wawancara adalah untuk menggali umpan balik mendalam mengenai persepsi pengguna, terutama pada kesesuaian hasil dengan makna atau intensi yang dimaksud, kehadiran produk-produk yang tidak terduga namun tetap dianggap relevan oleh partisipan, serta tingkat kepuasan mereka secara keseluruhan terhadap kualitas dan keberagaman hasil pencarian yang diberikan masing-masing *model embedding.* Pendekatan ini memungkinkan dilakukannya analisis tematik untuk memahami secara mendalam bagaimana perbedaan antar model embedding memengaruhi pengalmaan dan preferensi pengguna.

Keempat kategori kueri yang digunakan dalam eksperimen ini sekaligus merepresentasikan dua pola pencarian utama dalam e-commerce, yaitu *lookup search* dan exploratory search. Kueri pendek dan sedang menggambarkan pola lookup search, di mana partisipan memiliki tujuan pencarian yang jelas dan spesifik. Sebaliknya, kueri panjang dan ambigu merepresentasikan pola exploratory search, yang digunakan ketika partisipan belum memiliki gambaran produk tertentu dan cenderung mengeksplorasi berbagai kemungkinan hasil

pencarian. Dengan demikian, eksperimen ini mampu mencakup variasi perilaku pencarian yang umum terjadi pada pengguna e-commerce.

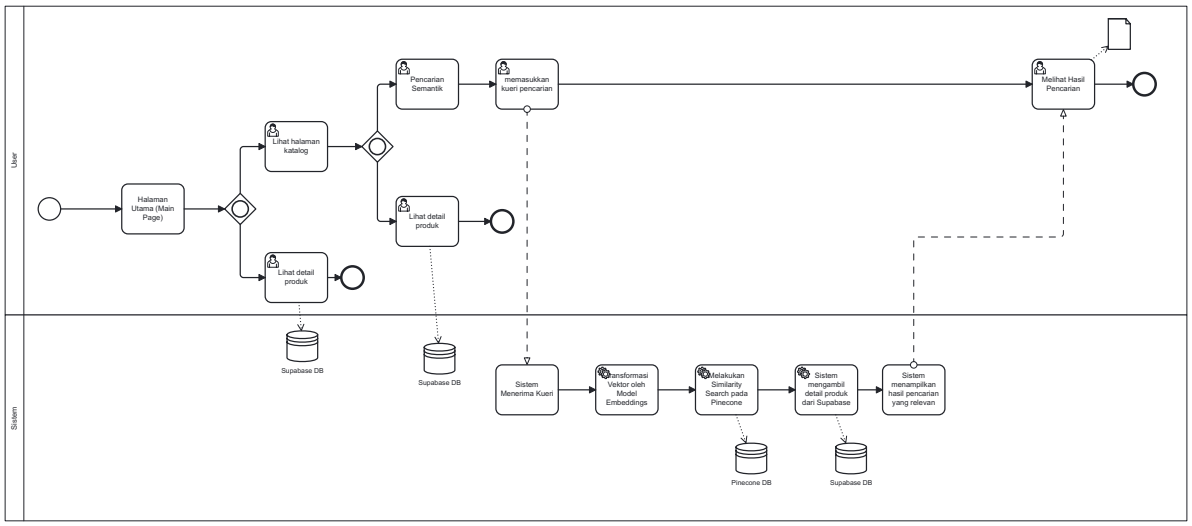
BAB IV
HASIL DAN PEMBAHASAN

Bab ini menjelaskan hasil implementasi dari tahap *design* 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 sistem pencarian *e-commerce* ini mengadopsi metodologi *Extreme Programming* (XP) yang bersifat iteratif untuk memastikan pembangunan yang terstruktur dan bertahap. Oleh karena itu, pemaparan akan diawali dengan rincian hasil implementasi yang terbagi ke dalam tiga iterasi utama, mencakup konfigurasi lingkungan, persiapan data, hingga implementasi fitur pencarian semantik.

Setelah menguraikan hasil pengembangan sistem, fokus pembahasan akan beralih pada pembahasan temuan penelitian. Bagian ini akan menyajikan hasil evaluasi kualitatif melalui metode *blind A/B Testing* yang dirancang untuk membandingkan secara langsung efektivitas *model embeddings* OpenAI *text-embedding-3-large* dan VoyageAI *voyage-3-large* dari perspektif pengguna. Puncak dari bab ini adalah analisis dan pembahasan mendalam terhadap analisis kualitatif, yang bertujuan untuk menjawab rumusan masalah mengenai preferensi pengguna dan faktor-faktor yang memengaruhinya. Untuk memberikan gambaran menyeluruh mengenai alur kerja sistem yang menjadi objek evaluasi, pembahasan akan dimulai dengan penyajian diagram proses bisnis.

4.1 Diagram Proses Bisnis

Untuk memberikan gambaran menyeluruh mengenai alur kerja sistem dari perspektif pengguna akhir, sebuah diagram proses bisnis dirancang menggunakan notasi BPMN (*Business Process Model and Notation*). Sebagaimana diilustrasikan pada Gambar 4.1, diagram ini memetakan interaksi antara pengguna dan sistem *e-commerce*, sekaligus menjabarkan proses yang terjadi selama fungsi pencarian semantik dijalankan. Diagram ini terbagi menjadi dua lane utama, yaitu "*User*" dan "*Sistem*", yang secara visual memisahkan tindakan yang diinisiasi oleh pengguna dengan proses yang dieksekusi oleh sistem.



Gambar 4.1 Diagram Proses Bisnis

Alur kerja diawali ketika pengguna mengakses aplikasi dan tiba di halaman utama. Pada titik ini, pengguna dapat memilih untuk menjelajahi produk yang disajikan atau menavigasi ke halaman katalog. Kedua tindakan awal ini memicu proses pada *lane* "Sistem", di mana sistem akan mengambil data detail produk dari SupabaseDB untuk ditampilkan kepada pengguna.

Fungsionalitas inti, yakni pencarian semantik, dimulai saat pengguna memasukkan kueri pada kolom pencarian yang tersedia. Setelah kueri diterima, alur proses beralih sepenuhnya ke *lane* "Sistem". Langkah pertama adalah transformasi kueri teks menjadi representasi vektor menggunakan *model embeddings* yang aktif, baik OpenAI maupun VoyageAI. Vektor kueri yang telah dihasilkan ini kemudian dimanfaatkan untuk menjalankan *similarity search* di dalam Pinecone *vector database*. Setelah Pinecone mengembalikan daftar ID produk yang paling relevan secara semantik, sistem melanjutkan proses dengan mengambil detail produk secara lengkap dari SupabaseDB. Pada tahap akhir, sistem menyajikan hasil pencarian yang telah terurut berdasarkan relevansi pada antarmuka, yang selanjutnya dapat ditinjau oleh pengguna.

4.2 Fase *Coding* (Implementasi Sistem)

Pada sub-bab ini, peneliti menguraikan proses implementasi sistem pencarian *e-commerce* berbasis pencarian semantik. Pengembangan sistem ini dilakukan dengan pendekatan iteratif sesuai metodologi *Extreme Programming* (XP), yang secara garis besar terbagi menjadi tiga iterasi utama untuk memastikan pembangunan yang terstruktur dan bertahap.

4.2.1 Iterasi Pertama: Konfigurasi Proyek

Iterasi pertama dalam siklus pengembangan difokuskan pada pembangunan fondasi proyek. Tujuan utama tahap ini adalah untuk memastikan seluruh *environment*, meliputi basis data relasional, *vector database*, dan klien API untuk *model embedding*, dapat saling terhubung dan berfungsi dengan baik. Aktivitas utama pada iterasi ini mencakup konfigurasi basis data, inisialisasi koneksi ke Pinecone, dan penyiapan API *client*.

A. Konfigurasi Prisma ORM & Supabase Postgresql Database

Interaksi dengan basis data PostgreSQL di Supabase dikelola menggunakan Prisma ORM untuk memfasilitasi perancangan skema, migrasi, dan eksekusi kueri secara *type-safe*. Entitas data utama yang menjadi fokus dalam implementasi pencarian semantik adalah model Product, yang skemanya didefinisikan dalam file prisma.schema.

```
generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider  = "postgresql"
  url       = env("DATABASE_URL")
  directUrl = env("DIRECT_URL")
}

model Product {
  id          Int      @id @default(autoincrement())
  categories  String[]
  name        String
  description String   @db.Text
  price       BigInt   @db.BigInt
  images      String[]

  orders OrderProduct[]

  created_at DateTime @default(now())
  updated_at DateTime @updatedAt
}
```

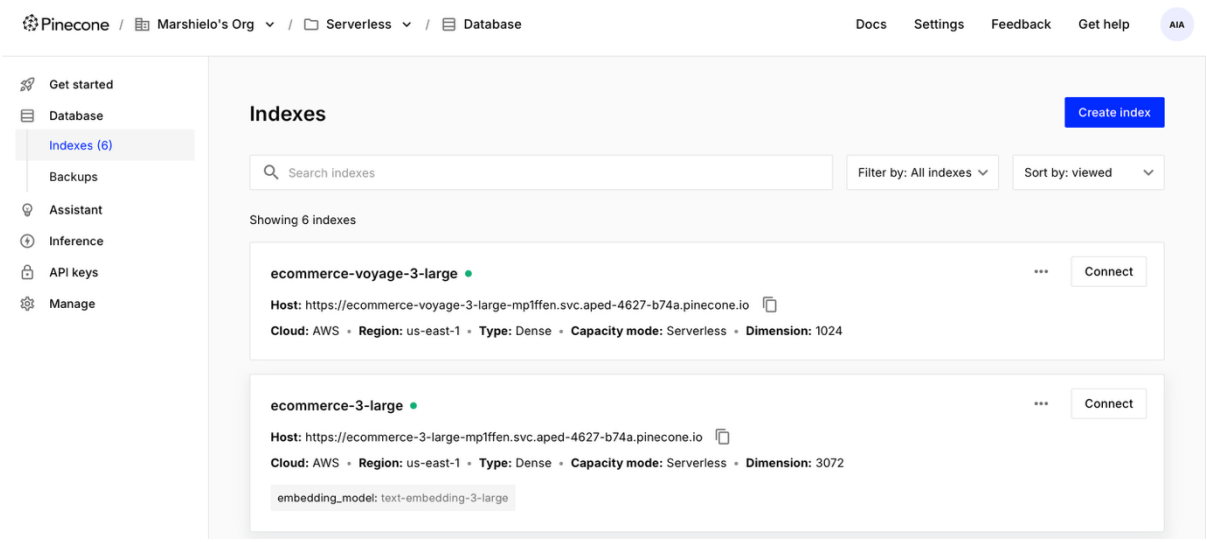
Gambar 4. 2 Konfigurasi prisma.schema

Model *Product* adalah entitas data utama yang digunakan untuk fitur pencarian semantik. *id* berfungsi sebagai primary key yang menghubungkan metadata produk di PostgreSQL dengan representasi vektornya di Pinecone. Sementara itu, atribut *name, description*, dan

categories merupakan sumber data tekstual utama yang dalam implementasinya digabungkan menjadi satu input komprehensif untuk diubah menjadi *vector embeddings* oleh model OpenAI dan VoyageAI. Atribut lainnya seperti *price* dan *images* digunakan untuk menyajikan data produk secara lengkap kepada pengguna setelah hasil pencarian diperoleh.

B. Konfigurasi Pinecone Vector Database

Untuk mendukung blind testing antara dua model *embedding*, dua index terpisah dibuat di Pinecone: *ecommerce-3-large* untuk model OpenAI dengan 3072 dimensi, dan *ecommerce-voyage-3-large* untuk model VoyageAI dengan 1024 dimensi. Kedua index tersebut menggunakan metrik *cosine similarity* untuk perhitungan kemiripan vektor. Gambar 4. 3 menunjukkan kedua indeks yang telah berhasil dibuat pada Pinecone.



Gambar 4. 3 Pinecone Indeks pada Pinecone Console.

Untuk berinteraksi dengan Pinecone, dikembangkan beberapa fungsi inti dalam TypeScript. Fungsi `initializePinecone` dibuat untuk membuat index secara dinamis jika belum ada, dengan dimensi yang disesuaikan berdasarkan nama index (3072 untuk OpenAI atau 1024 untuk VoyageAI).

```
import { Pinecone, PineconeRecord, RecordMetadata } from "@pinecone-database/pinecone";
const pc = new Pinecone({ apiKey: process.env.PINECONE_API_KEY as string });

export async function initializePinecone(indexName: string) {
  try {
    const existingIndexes = await pc.listIndexes();
```

```
    if (!existingIndexes.indexes?.some((index) => index.name === indexName))
{
      console.log(`Creating index "${indexName}"...`);
      await pc.createIndex({
        name: indexName,
        dimension: indexName === "ecommerce-3-large" ? 3072 : 1024,
        metric: "cosine" as const,
        spec: {
          serverless: {
            cloud: "aws" as const,
            region: "us-east-1",
          },
        },
      });
      console.log(`Index "${indexName}" created!`);
    } else {
      console.log(`Index "${indexName}" exists.`);
    }

    const index = await pc.describeIndex(indexName);
    console.log(`Retrieved index "${index.name}".`);

    return index;
  } catch (error) {
    console.error("Error initializing Pinecone:", error);
    throw error;
  }
}
```

Gambar 4. 4 Konfigurasi Pinecone

Prosesnya dimulai dengan memanggil `pc.listIndexes()` untuk memeriksa semua index yang ada. Jika index dengan nama yang diberikan (indexName) belum ada, fungsi `pc.createIndex()` akan dieksekusi. Di dalamnya, nilai dimension diatur secara dinamis: jika indexName adalah "*ecommerce-3-large*" (untuk OpenAI), dimensi diatur ke 3072. Jika tidak, dimensi akan diatur ke 1024 (untuk VoyageAI). Hal ini dilakukan untuk memastikan setiap index dibuat dengan konfigurasi yang benar sesuai dengan *model embedding* pasangannya. Setelah index dipastikan siap, fungsi ini mengembalikan objek yang berisi detail index tersebut agar dapat digunakan oleh fungsi-fungsi lain.

Selanjutnya, fungsi `upsertProductVector` digunakan untuk memasukkan atau memperbarui data vektor produk ke dalam index yang ditentukan. Fungsi ini menerima ID produk, data vektor (embedding), dan metadata terkait.

```
Export async function upsertProductVector(productId: number, embedding: number[], metadata: RecordMetadata, indexName: string): Promise<void> {
  const vector: PineconeRecord<RecordMetadata> = {
    id: productId.toString(),
    values: embedding,
    metadata,
  };

  const indeks = pc.Index(indexName); // Dynamically use the indexName here
  await indeks.namespace("products-1").upsert([vector]);
}
```

Gambar 4. 5 Fungsi Upsert ke Pinecone Indeks

Fungsi ini menerima empat parameter utama: `productId` sebagai ID unik, embedding yang merupakan data vektornya, metadata untuk menyimpan informasi tambahan seperti nama atau kategori produk, dan indexName untuk menentukan index tujuan. Di dalam fungsi, sebuah objek vector diformat sesuai dengan struktur yang disyaratkan oleh Pinecone. Perintah `pc.Index(indexName)` kemudian digunakan untuk memilih index target secara dinamis. Akhirnya, metode .upsert() dieksekusi untuk menyimpan vektor tersebut. Metode ini akan menambahkan vektor baru jika ID-nya belum ada, atau memperbaruinya jika sudah ada.

```
export async function searchProductVectors(
queryEmbedding: number[],
indexName: string,
topK: number = 15
) {
const index = pc.Index(indexName); // Memilih index secara dinamis
return await index.namespace("products-1").query({
vector: queryEmbedding,
topK,
includeMetadata: true,
});
}
```

Gambar 4. 6 Fungsi Pencarian Semantik

Fungsi terakhir untuk fitur pencarian adalah `searchProductVectors()`. Fungsi ini menerima sebuah vektor kueri, nama index target, dan jumlah hasil teratas yang diinginkan

(topK). Dengan menggunakan metode `.query()`, fungsi ini menghitung *cosine similarity* antara vektor kueri dengan semua vektor di dalam namespace "products-1" dan mengembalikan daftar produk yang paling relevan beserta metadatanya.

C. Konfigurasi OpenAI dan VoyageAI *Client API*

Langkah terakhir pada iterasi ini adalah melakukan konfigurasi *API client* untuk kedua *model embedding* yang dievaluasi. Untuk setiap model, dibuat fungsi terpisah yang bertugas melakukan *API calls* dan mengembalikan representasi vektor. Fungsi `generateOpenAIEmbeddings()` dibuat untuk berinteraksi dengan API OpenAI. Fungsi ini secara spesifik memanggil model OpenAI *text-embedding-3-large* untuk menghasilkan nilai vektor dengan 3072 dimensi.

```
import OpenAI from "openai";

const openai = new OpenAI({
  apiKey: process.env.OPENAI_API_KEY,
  dangerouslyAllowBrowser: true,
});

export async function generateProductEmbeddings(text: string): Promise<number[]
| null> {
  try {
    const response = await openai.embeddings.create({
      model: "text-embedding-3-large",
      input: text,
    });
    const productEmbeddings = response.data[0].embedding;
    if (response.data && response.data && response.data.length > 0 &&
response.data[0].embedding) {
      return productEmbeddings;
    } else {
      console.error("Invalid embedding response structure:", response.data);
      return null;
    }
  } catch (error: any) {
    console.error("Failed to generate product embeddings:", error.response?.data
|| error.message || error);
    return null;
  }
}
```

Gambar 4. 7 Fungsi Transformasi Embeddings OpenAI

Fungsi di atas melakukan inisialisasi terhadap OpenAI *client* menggunakan API *key* yang disimpan dalam *environment variables*. Fungsi `generateProductEmbeddings()` kemudian menerima *text* sebagai *input*. Di dalamnya, metode `openai.embeddings.create()` dipanggil dengan dua parameter utama: model yang diatur ke "*text-embedding-3-large*", dan input yang berisi teks yang akan diubah. Fungsi ini kemudian mengembalikan *vector embeddings* yang dihasilkan oleh API.

Serupa dengan OpenAI, fungsi `generateVoyageEmbeddings()` dibuat untuk menghubungkan ke API VoyageAI. Fungsi ini memanggil model *voyage-3-large* yang menghasilkan vektor dengan 1024 dimensi.

```
Import { VoyageAIClient } from "voyageai";

const client = new VoyageAIClient({ apiKey: process.env.VOYAGEAI_API_KEY });

export async function generateVoyageProductEmbeddings(text: string): 
Promise<number[] | null> {
  try {
    const response = await client.embed({
      input: text,
      model: "voyage-3-large",
    });
    const productEmbeddings = response.data[0].embedding;
    if (response.data && response.data && response.data.length > 0 && 
response.data[0].embedding) {
      return productEmbeddings;
    } else {
      console.error("Invalid embedding response structure:", response.data);
      return null;
    }
  } catch (error: any) {
    console.error("Failed to generate product embeddings:", error.response?.data 
|| error.message || error);
    return null;
  }
}
```

Gambar 4. 8 Fungsi Transformasi Embeddings VoyageAI

VoyageAI *client* diinisialisasi menggunakan API *key* yang spesifik untuk service tersebut. Fungsi `generateVoyageEmbeddings()` memanggil metode `.embed()` yang disediakan oleh klien VoyageAI. Metode ini juga menerima parameter input berupa teks dan

model yang diatur ke "*voyage-3-large*". Fungsi ini bertugas mengembalikan vektor yang dihasilkan dari pemrosesan oleh model VoyageAI. Dengan adanya dua fungsi terpisah ini, sistem dapat secara fleksibel memanggil model *embedding* yang berbeda sesuai dengan skenario *blind A/B testing* yang sedang berjalan.

4.2.2 Iterasi Kedua: Implementasi Pre-Processing dan Indexing Data

Sesuai dengan metode persiapan data yang telah diuraikan pada Bab 3.3, proses pra-pemrosesan dan seeding data berhasil dijalankan. Sebuah dataset baru yang telah diperkaya dengan deskripsi produk berhasil dibuat menggunakan model GPT-4o. Selanjutnya, seluruh metadata produk berhasil disimpan ke dalam basis data Supabase PostgreSQL, dan representasi vektor dari kedua model (OpenAI dan VoyageAI) berhasil diindeks ke dalam dua indeks terpisah di Pinecone. Dengan ini, sistem telah siap untuk implementasi fitur pencarian.

A. Implementasi Augmentasi Deskripsi Dataset

Strategi prompt engineering yang telah dirancang pada Bab III diimplementasikan ke dalam sebuah fungsi Python, create_product_prompt(). Fungsi ini secara dinamis mengambil atribut dari setiap produk dan menyisipkannya ke dalam prompt, seperti yang ditunjukkan pada Gambar 4.10.

```
def create_product_prompt(row):
    try:
        price_formatted = f"IDR {int(row['price']):,}"
    except (ValueError, TypeError):
        price_formatted = "Price information not available"

    prompt = f"""
    You are a highly experienced Product Marketing and Semantic SEO expert for
a premium international e-commerce platform.
    Your task is to write a very detailed, rich, and specific product
description in fluent **English**. This description must provide a deep,
contextual understanding of the product, intended for semantic search research.

    **Product Data:**
    - Product Name: {row['name']}
    - Category Path: {row['categories']}
    - Price: {price_formatted}

    **Writing Instructions:**
    1.  **Language:** The entire output must be in **English**.
    2.  **Narrative Format:** Describe the product in a single, flowing,
information-rich narrative paragraph. **DO NOT USE bullet points or list
formats.**
```

```
    3.  **Deep Elaboration:** Do not just list features from the product name.
Elaborate on every detail, turning it into a tangible benefit. For example, if
the name includes 'slim fit', describe how this cut creates a modern, tailored
silhouette. If it mentions a material like 'katun' (cotton), explain its
comfort and feel (e.g., 'breathable, soft on the skin, and perfect for tropical
climates').
    4.  **Usage Context:** Imagine and describe the target user (e.g., a young
professional, a university student, a fashion enthusiast) and the ideal
situations or events for using this product (e.g., for a formal event, office
wear, a wedding reception, or a casual weekend outing).
    5.  **Lifestyle Connection:** Create a story or connect the product to a
specific lifestyle, aspiration, or value. Make the product feel like more than
just an item.
    6.  **Specificity and Detail:** Ensure the description is highly specific,
contextually mentioning details from the product name and category to enrich
its semantic meaning.
    7.  **Ideal Length:** Aim for a description between 80 and 150 words to
ensure informational depth.
    8.  **Direct Output:** Provide only the description text itself, without
any introductory or concluding remarks from you as the AI.
    """
    return prompt.strip()
```

Gambar 4. 9 Fungsi instruksi pembuatan deskripsi produk sesuai dengan prompt

Fungsi tersebut kemudian dieksekusi dalam sebuah loop yang mengirimkan permintaan API ke model gpt-4o untuk setiap produk. Parameter temperature diatur pada nilai 0.75 untuk menjaga keseimbangan antara kreativitas dan relevansi deskripsi. Proses ini juga dilengkapi dengan error handling sederhana dan jeda satu detik di antara setiap panggilan API. Hasil implementasi skrip eksekusi dapat dilihat pada Gambar 4.10

```
generated_descriptions = []

for index, row in target_df.iterrows():
    prompt = create_product_prompt(row)

    try:
        response = client.chat.completions.create(
            model="gpt-4o",
            messages=[{"role": "user", "content": prompt}],
            temperature=0.75,
            max_tokens=300
        )
        description = response.choices[0].message.content.strip()
        generated_descriptions.append(description)
```

```
    except Exception as e:
        generated_descriptions.append("GENERATION_FAILED")

    time.sleep(1)
```

Gambar 4.10 Kode untuk *generate* deskripsi menggunakan model gpt-4o

Sebagai tahap akhir dari implementasi ini, semua deskripsi yang berhasil dibuat kemudian diintegrasikan kembali ke dalam DataFrame dan disimpan sebagai sebuah berkas CSV baru, yang ditunjukkan pada Gambar 4.12 yang menjadi sumber data final untuk tahapan selanjutnya.

```
target_df['generated_description_en'] = generated_descriptions

OUTPUT_PATH = "/kaggle/working/ECOMMERCE-DATASET-GENERATED.csv"

target_df.to_csv(OUTPUT_PATH, index=False, encoding='utf-8')

print(f"dataset saved successfully at: {OUTPUT_PATH}")
```

Gambar 4.11 Kode untuk menyimpan deskripsi ke dalam CSV

B. Implementasi *Seeding* dan *Indexing* Data

Sebagai bagian dari implementasi, dibuat sebuah objek konfigurasi `modelConfigs` untuk mengelola pemanggilan fungsi dan penentuan indeks yang benar untuk OpenAI dan VoyageAI secara dinamis. Konfigurasi ini memastikan setiap model menggunakan indeks dengan dimensi yang sesuai, seperti yang ditunjukkan pada Gambar 4.12.

```
const modelConfigs = {
  openai: {
    indexName: "ecommerce-3-large",
    namespace: "products-1",
    generateEmbeddings: generateProductEmbeddings,
  },
  voyage: {
    indexName: "ecommerce-voyage-3-large",
    namespace: "products-1",
    generateEmbeddings: generateVoyageProductEmbeddings,
  },
};
```

Gambar 4.12 Objek konfigurasi modelConfigs untuk model OpenAI dan VoyageAI.

Script kemudian diimplementasikan untuk melakukan iterasi pada data, mengunggah gambar ke Supabase Storage, dan menyimpan metadata produk ke dalam tabel Product menggunakan metode prisma.product.create(). Implementasi ini berhasil menciptakan record baru untuk setiap produk dan menghasilkan ID unik yang menjadi kunci penghubung antara data di Supabase dengan vektor di Pinecone.

```
const product = await prisma.product.create({
            data: {
              name,
              categories: categoryArray,
              description,
              price: parseFloat(price),
              images: uploadedImages,
            },
          });
```

Gambar 4.13 Kode untuk Membuat Record Baru menggunakan Prisma

Tahap akhir dari implementasi *script* adalah proses batch indexing yang dieksekusi secara paralel menggunakan Promise.all. Untuk setiap produk, gabungan teks dari nama, deskripsi, dan kategori dikirimkan ke API OpenAI dan VoyageAI untuk menghasilkan vektor. Vektor-vektor ini kemudian berhasil disimpan (upsert) ke indeks Pinecone yang sesuai, seperti yang ditunjukkan pada implementasi kode berikut:

```
await Promise.all(
  batch.map(async (product) => {
    const { id, name, description, categories } = product;
    const embeddingText = `${name} ${description || ""} ${categories.join("
")}`;

    const metadata = { name, description: description || "", /* ... */ };

    // Proses untuk model OpenAI
    const openaiEmbedding = await
modelConfigs.openai.generateEmbeddings(embeddingText);
    if (openaiEmbedding) {
      await openaiPineconeService.upsertProductVector(id, openaiEmbedding,
metadata);
    }

    // Proses untuk model VoyageAI
```

```
    const voyageEmbedding = await
modelConfigs.voyage.generateEmbeddings(embeddingText);
    if (voyageEmbedding) {
      await voyagePineconeService.upsertProductVector(id, voyageEmbedding,
metadata);
    }
  })
);
```

Gambar 4. 14 Kode untuk upsert (indeks) ke Pinecone

Dengan selesainya implementasi ini, seluruh data produk telah siap di dalam sistem, baik dalam bentuk metadata relasional maupun representasi vektor, untuk digunakan oleh fitur pencarian semantik pada iterasi selanjutnya.

4.2.3 Iterasi Ketiga: Implementasi API dan Hasil Implementasi Website

Iterasi ketiga difokuskan pada pengembangan antarmuka fungsional dan logika backend yang menghubungkannya. Tujuan utama dari iterasi ini adalah untuk mengintegrasikan semua komponen menjadi sebuah sistem *e-commerce*, yang memungkinkan pengguna untuk berinteraksi dengan sistem pencarian semantik dan mengakomodasi proses evaluasi *blind A/B Testing*.

A. Implementasi API

Untuk menunjang fitur aplikasi, dua *endpoint API* dikembangkan. Endpoint pertama bertanggung jawab untuk pengambilan data katalog secara umum, sedangkan endpoint kedua secara khusus menangani logika pencarian semantik yang menjadi inti dari penelitian ini.

a. Endpoint Pengambilan Daftar Produk

```
export async function GET(request: NextRequest) {
  try {
    const url = new URL(request.url);
    const limit = url.searchParams.get("limit");

    const products = await prisma.product.findMany({
      take: limit ? parseInt(limit, 10) : undefined, // Gunakan limit jika ada,
jika tidak biarkan undefined
      select: {
        id: true,
        images: true,
        name: true,
        categories: true,
        price: true,
      },
    });
```

```
    const mappedProducts = products.map((item) => {
      let imageUrl = (item as any).image_url;
      if (!imageUrl && item.images && item.images.length > 0) {
        imageUrl = item.images[0].startsWith("http") ? item.images[0] :
getImageUrl(item.images[0], "products");
      }

      return {
        ...item,
        image_url: imageUrl || null,
      };
    });

    return new Response(safeJsonStringify(mappedProducts), {
      status: 200,
      headers: { "Content-Type": "application/json" },
    });
  } catch (error) {
    return new Response(JSON.stringify({ error: "Internal server error" }), {
      status: 500,
      headers: { "Content-Type": "application/json" },
    });
  }
}
```

Gambar 4.15 Endpoint untuk mengambil data produk

Endpoint pada path `/api/products` ini bertugas untuk mengambil daftar produk dari database. Ketika diakses dengan metode GET, endpoint ini akan menjalankan kueri ke Supabase PostgreSQL menggunakan `prisma.product.findMany()` untuk memperoleh data produk. Selain itu, terdapat parameter limit opsional dari *body* permintaan, untuk membatasi jumlah produk yang ditampilkan.

b. Endpoint Pencarian Semantik

```
export async function POST(request: NextRequest) {
  try {
    const { search, model, indexName, namespace } = await request.json();

    if (!search || search.trim() === "") {
      return new Response(JSON.stringify({ error: "Search query is required"
}), {
        status: 400,
        headers: { "Content-Type": "application/json" },
      });
```

```
    }

    let products = [];

    try {
      products = await searchProducts(search.trim(), model, indexName,
namespace);
    } catch (searchError) {
      console.error("Vector search failed, falling back to text search:",
searchError);
      products = await prisma.product.findMany({
        where: {
          OR: [{ name: { contains: search.trim(), mode: "insensitive" } }, {
description: { contains: search.trim(), mode: "insensitive" } }],
        },
        select: {
          id: true,
          images: true,
          name: true,
          categories: true,
          price: true,
        },
      });
    }

    return new Response(safeJsonStringify(products), {
      status: 200,
      headers: { "Content-Type": "application/json" },
    });
  } catch (error) {
    console.error("Error in search API:", error);
    return new Response(JSON.stringify({ error: "Internal server error" }), {
      status: 500,
      headers: { "Content-Type": "application/json" },
    });
  }
}
```

Gambar 4.16 Endpoint untuk Melakukan Pencarian Semantik

Endpoint ini menangani semua permintaan pencarian semantik. Langkah pertama adalah memvalidasi untuk memastikan permintaan berisi kueri pencarian. Jika valid, langkah selanjutnya adalah memanggil fungsi layanan `searchProducts()`, yang menjalankan proses pencarian semantik. Hasil dari pencarian semantik kemudian dikirim kembali ke antarmuka pengguna sebagai respon.

```
export async function searchProducts(
  searchQuery: string,
  model: "openai" | "voyage" = "voyage",
  indexName: string = "ecommerce-voyage-3-large",
  namespace: string = "products-1",
  topK: number = 50,
  similarityThreshold: number = 0.3
): Promise<any[]> {
 //...kode sebelumnya....
   const queryResponse = await searchProductVectors(queryEmbedding, indexName);
   const vectorMatches = queryResponse.matches || [];
   const filteredMatches = vectorMatches.filter((match) => match.score >=
similarityThreshold);

  if (filteredMatches.length === 0) {
    logger.logNoResults(similarityThreshold);
    logger.logSearchComplete();
    return [];
  }

  // kode lanjutan...
}
```

Gambar 4. 17 Fungsi searchProducts untuk pencarian vektor.

Fungsi `searchProducts()` mengelola seluruh proses pencarian semantik secara sistematis. Proses dimulai dengan mengubah kueri teks dari pengguna menjadi nilai vektor, yang disimpan dalam variabel `queryEmbedding`. Setelah *embedding* berhasil dibuat, `queryEmbedding` tersebut digunakan untuk mencari produk yang paling mirip dalam database vektor Pinecone melalui Fungsi `searchProductVectors()`.

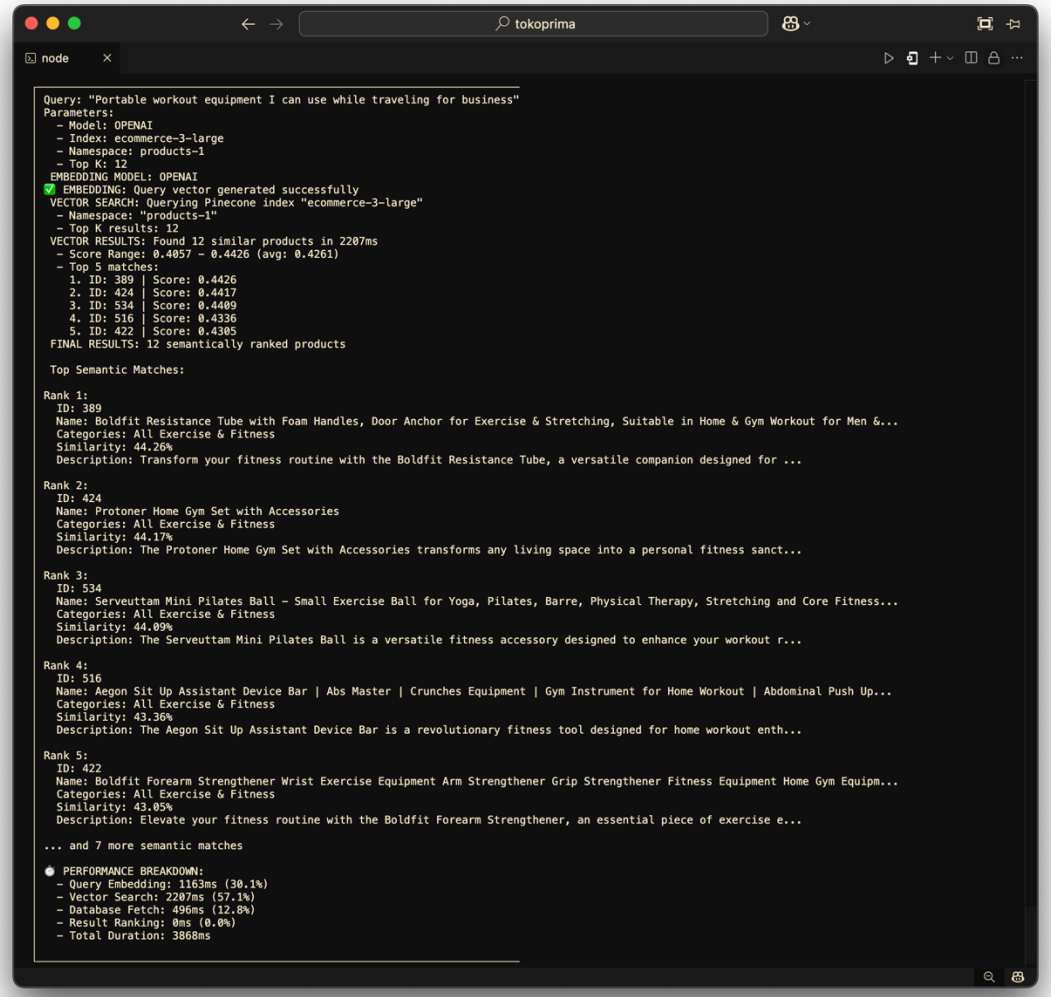
```
export async function searchProductVectors(queryEmbedding: number[], indexName:
string, topK: number = 12) {
  const index = pc.Index(indexName);
  return await index.namespace("products-1").query({
    vector: queryEmbedding,
    topK,
    includeMetadata: true,
  });
}
```

Gambar 4.18 Kode untuk menjalankan kueri ke database vektor.

Fungsi `searchProductVectors()` digunakan untuk mencari produk berdasarkan vektor kueri dalam database vektor Pinecone. Fungsi ini menerima parameter

`queryEmbedding`, yang merupakan representasi vektor kueri, serta `indexName` untuk menentukan indeks yang digunakan dalam pencarian. Fungsi ini melakukan pencarian di namespace `"products-1"` dengan parameter topK untuk menentukan jumlah hasil teratas yang diambil. Hasil pencarian berupa produk-produk yang paling mirip dengan vektor kueri dan menyertakan metadata terkait produk. Fungsi ini dijalankan secara *asynchronous.*

c. Gambaran Hasil Pencarian Semantik



Gambar 4.19 Logging Pencarian Semantik

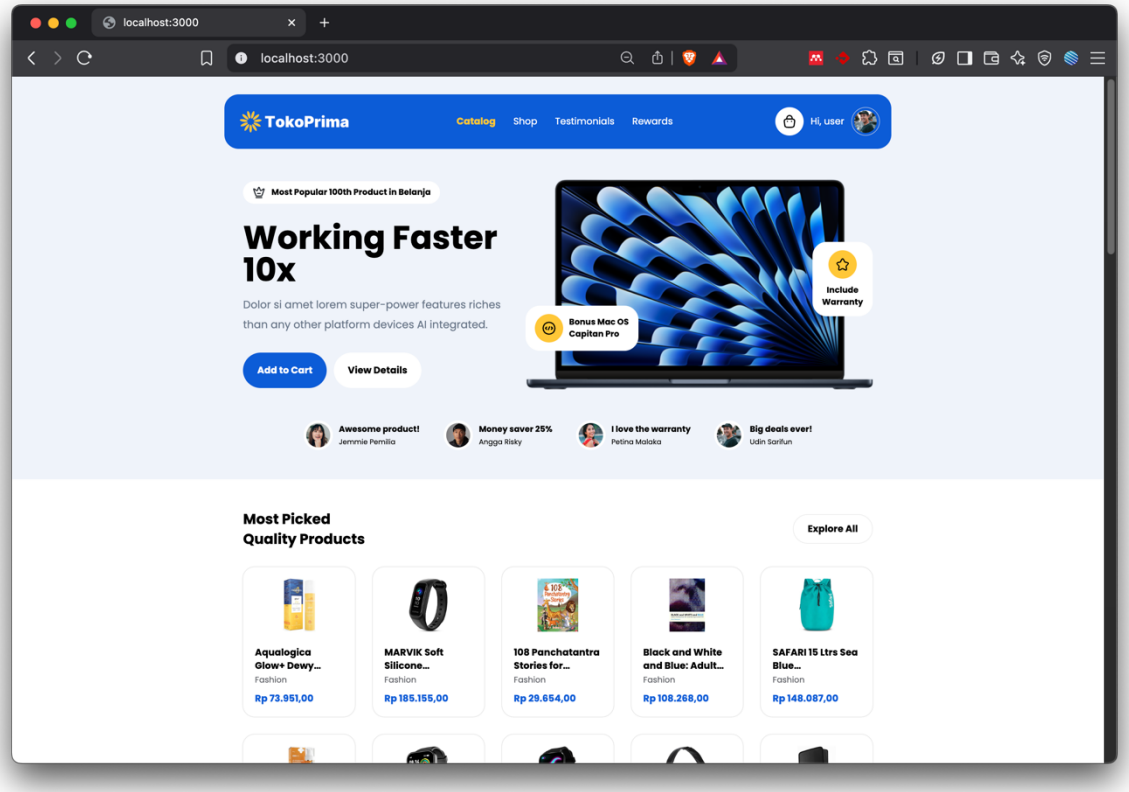
Gambar 4.19menampilkan *log* proses yang dihasilkan oleh fungsi `searchProducts()` ketika memproses sebuah kueri pencarian. Pada gambar, sistem memproses kueri pencarian "*portable workout equipment I can use while traveling for businesses"* menggunakan Model A (OpenAI *text-embedding-3-large*) yang mencari pada indeks ecommerce-3-large di Pinecone.

Log tersebut menjelaskan beberapa tahapan. Pertama, sistem menunjukkan pencarian vektor berhasil menemukan 12 produk yang relevan dengan *similarity score* lima produk teratas berkisar di angka 0.41 – 0.44. Selanjutnya total durasi yang diperlukan untuk melakukan setiap tahapan pencarian, mulai dari pencarian vektor hingga total waktu eksekusi. Pada kueri pencarian ini proses memakan waktu hampir 4 detik (3868*ms*).

B. Hasil Implementasi Antarmuka Pengguna

1. Halaman Utama

Berikut adalah hasil implementasi untuk halaman utama pada Gambar 4. 20.

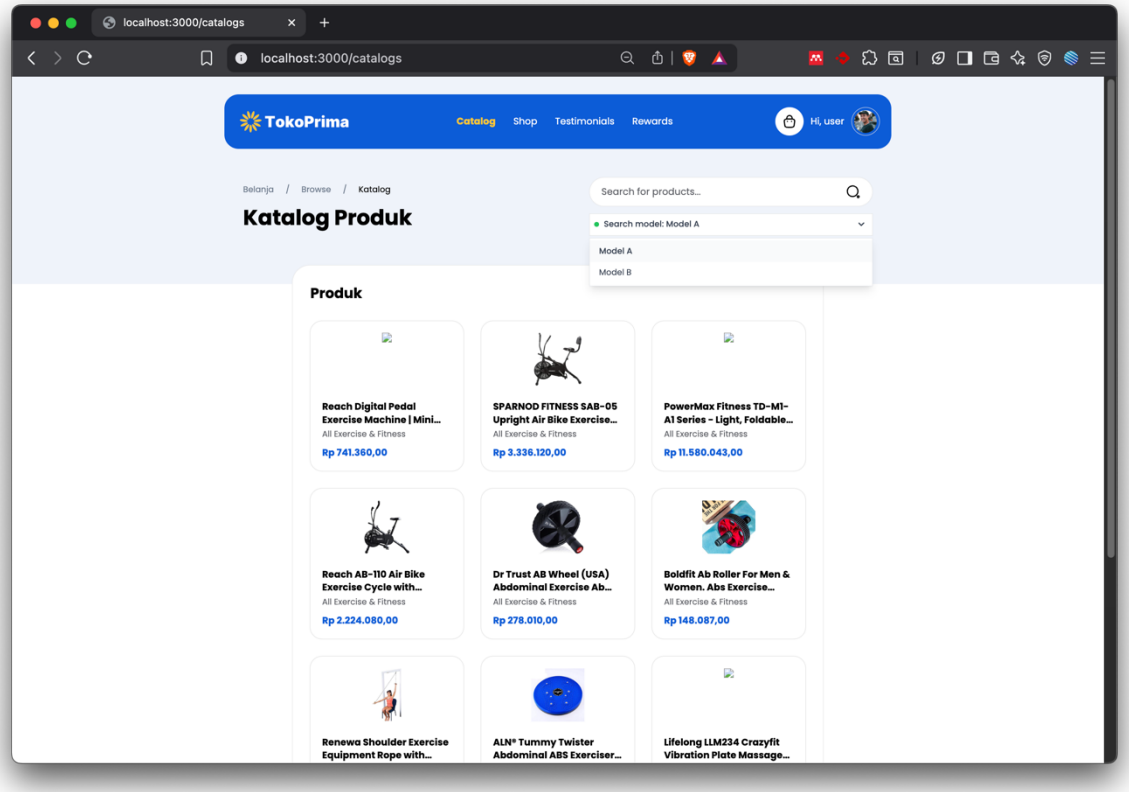


Gambar 4. 20 Halaman Utama

Tampilan halaman utama adalah halaman aplikasi yang terbuka saat pertama kali diakses pengguna. Halaman ini menyediakan navigasi menuju halaman katalog untuk melihat dan mencari produk. Untuk menampilkan daftar produk tersebut, halaman ini mengonsumsi endpoint `POST /api/products`, dengan mengirimkan parameter limit untuk membatasi jumlah produk yang ditampilkan.

2. Halaman Katalog

Berikut adalah hasil implementasi untuk halaman katalog pada Gambar 4. 21



Gambar 4. 21 Halaman Katalog

Tampilan halaman Katalog adalah tempat pengguna menjelajahi produk dan menggunakan fitur pencarian semantik. Halaman ini mengonsumsi dua endpoint API berbeda tergantung pada aksi pengguna. Saat pertama kali diakses, halaman ini memanggil `POST /api/products` untuk menampilkan seluruh katalog. Namun, ketika pengguna melakukan pencarian, halaman akan beralih memanggil `POST /api/search` dengan mengirimkan teks kueri dan parameter model yang aktif untuk mendapatkan hasil pencarian yang relevan secara semantik.

4.3 *Testing Phase* (Fase Testing)

Setelah tahap implementasi selesai, sistem memasuki tahap Testing untuk pengecekan fungsional. Tujuan dari fase ini adalah untuk memastikan bahwa sistem berjalan dengan baik dan setiap fiturnya berfungsi sesuai dengan spesifikasi yang telah dirancang sesuai dengan analisis kebutuhan sistem. Pengujian ini dilakukan menggunakan metode *black box testing* yang berfokus pada alur kerja utama dari perspektif pengguna. Skenario dan hasil dari pengujian fungsional ini dirangkum pada Tabel 4. 1.

4.3.1 Pengujian dan Validasi Fungsional Sistem

Pengujian fungsional dilakukan untuk memastikan setiap fitur berjalan sesuai dengan yang diharapkan. Pengujian ini menggunakan metode *black box testing*, yang berfokus pada fungsionalitas sistem dari perspektif pengguna tanpa memperhatikan struktur kode internal. Skenario pengujian mencakup semuasp alur kerja utama yang relevan untuk penelitian ini. Tabel 4. 1 Tabel Pengujian di bawah ini merangkum skenario pengujian fungsional yang dilakukan beserta hasilnya.

Tabel 4. 1 Tabel Pengujian

Skenario Pengujian	**Langkah-langkah pengujian**	**Hasil yang diharapkan**	**Hasil Aktual**	**Status**
Akses Halaman Utama	1. Buka browser. 2. Masukkan URL root aplikasi. 3. Tekan Enter.	Aplikasi memuat dan menampilkan halaman utama tanpa error.	Halaman utama berhasil dimuat dan ditampilkan sesuai harapan.	Lulus
Fungsionalitas Pencarian Semantik	1. Navigasi ke halaman katalog. 2. Ketik kueri pada kolom pencarian. 3. Tekan Enter.	Sistem menerima kueri, memprosesnya melalui API, dan mengembalikan produk yang relevan.	Sistem berhasil memproses kueri dan menampilkan hasil yang relevan tanpa kendala.	Lulus

Hasil pengujian fungsional pada Tabel 4. 1 menunjukkan bahwa semua skenario pengujian berhasil dan aplikasi berfungsi sesuai dengan spesifikasi yang telah ditentukan. Dengan demikian, prototipe sistem dinyatakan valid dan andal untuk digunakan pada tahap penelitian selanjutnya, yaitu evaluasi kualitatif melalui *blind A/B testing* dengan pengguna.

4.4 Fase *Listening*

4.4.1 Hasil Evaluasi *Blind A/B Testing* Kualitatif Pengguna

Setelah sistem divalidasi secara fungsional, penelitian memasuki tahap evaluasi dengan pengguna. Bagian ini menyajikan temuan dari blind A/B Testing yang dirancang untuk membandingkan efektivitas kedua model embedding berdasarkan persepsi dan pengalaman pengguna.

A. Demografi Partisipan

Demografi partisipan penelitian ditunjukkan pada Tabel 4. 2 Total partisipan berjumlah sembilan orang dengan rentang usia antara 21 hingga 33 tahun. Komposisi jenis kelamin terdiri atas enam laki-laki dan tiga perempuan. Dari sisi latar belakang gaya hidup, enam partisipan berasal dari kategori fitness yang mencakup gym enthusiasts, pro cyclist, dan personal trainer. Sementara itu, tiga partisipan lainnya berasal dari kategori skincare, termasuk dua orang skincare enthusiasts serta satu partisipan dengan latar belakang profesi dokter yang juga memiliki pengetahuan terkait produk perawatan kulit.

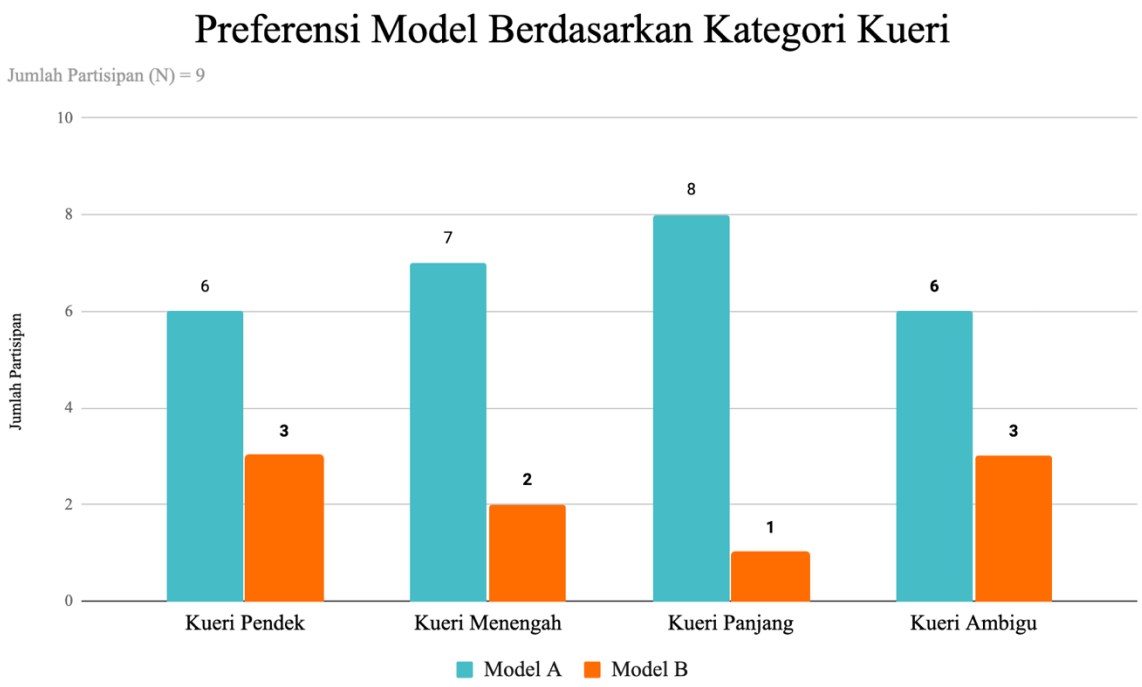
Tabel 4. 2 Data Partisipan

Responden	Usia	Jenis Kelamin	Domain Lifestyle/Pengetahuan
R1	23	L	Gym Enthusiasts
R2	30	L	Pro Cyclist
R3	22	L	Gym Enthusiasts
R4	25	L	Personal Trainer (OT)
R5	22	L	Gym Enthusiasts
R6	22	L	Gym Enthusiasts
R7	33	P	Dokter
R8	22	P	Skincare Enthusiasts
R9	21	P	Skincare Enthusiasts

Karakteristik ini menunjukkan adanya keberagaman dalam latar belakang dan pengalaman partisipan, sehingga memberikan sudut pandang yang bervariasi dalam menilai relevansi hasil pencarian semantik. Rentang usia yang relatif muda serta pengalaman berbelanja daring yang dimiliki juga mendukung kesesuaian profil partisipan dengan konteks penelitian.

B. Analisis Preferensi Pengguna Berdasarkan Kategori Kueri

Sesuai dengan prosedur evaluasi, data preferensi partisipan dianalisis berdasarkan empat kategori kueri yang telah diuji. Gambar 4.22 memperlihatkan keseluruhan preferensi pengguna untuk setiap kategori kueri.



Gambar 4.22 Preferensi Model Berdasarkan Kategori Kueri

a. Kueri Pendek

Pada kueri pendek, preferensi terbagi dengan enam dari sembilan partisipan memilih Model A. Alasan utama yang mendasari pilihan ini adalah hasil pencarian yang dinilai lebih relevan dan jelas. Seorang partisipan dari domain fitness menyatakan bahwa Model A dianggap bagus karena "tidak hanya terpatok pada satu pencarian spesifik, melainkan memberikan rekomendasi lain yang relevan" seperti menampilkan piringan beban saat mencari dumbbell. Di sisi lain, tiga partisipan yang memilih Model B menyebutkan bahwa hasilnya lebih spesifik dalam menangkap makna. Salah satu dari mereka berpendapat

bahwa Model B "lebih spesifik" karena menampilkan lebih banyak variasi dumbbell berukuran kecil yang sesuai dengan intensi pencariannya.

b. Kueri Sedang

Untuk kueri menengah, preferensi semakin menguat ke arah Model A, dengan tujuh dari sembilan partisipan memilihnya. Dukungan ini datang dari persepsi bahwa Model A lebih baik dalam memahami konteks kueri yang lebih spesifik. Seorang partisipan mengungkapkan bahwa hasil dari Model A "lebih presisi dan spesifik," sementara partisipan lain menyatakan bahwa produk yang ditampilkan "relevan dan tidak terlalu keluar dari konteks." Sebaliknya, Model B pada kategori ini dinilai cenderung memberikan hasil yang acak. Sebagai contoh, seorang partisipan yang mencari produk skincare menemukan bahwa Model B memang menampilkan beberapa produk yang relevan, namun secara keseluruhan hasilnya dinilai "'lebih random lagi' dan ada barang asing seperti sepatu.

c. Kueri Panjang

Dominasi Model A menjadi sangat jelas pada kueri panjang, di mana delapan dari sembilan partisipan secara konsisten memilihnya. Kemampuan Model A untuk memahami deskripsi yang kompleks dan bernuansa terbukti lebih unggul. Seorang partisipan dari domain skincare menyatakan bahwa Model A mampu "mencakup semua kebutuhan saya" dari kueri detail yang ia masukkan. Sebaliknya, Model B menunjukkan kelemahan dalam memproses kueri panjang, kerap kali hanya berfokus pada satu atau two kata kunci dan mengabaikan batasan lainnya. Seorang partisipan menggambarkan bahwa saat mencari "makeup remover with brightening effect", Model B salah memahami kata kunci "makeup" dan menampilkan produk-produk acak "seperti ring light dan nail polish," sehingga memberikan hasil yang kurang relevan.

d. Kueri Ambigu

Untuk kueri ambigu yang dirancang untuk menguji skenario pencarian eksplorasi, preferensi mayoritas tetap kuat untuk Model A, dengan enam dari sembilan suara. Para partisipan menilai Model A lebih cerdas dalam menerjemahkan intensi pengguna yang samar menjadi kategori produk yang relevan secara kontekstual. Sebagai contoh, saat dihadapkan pada kueri "skincare that works while I sleep", seorang partisipan mencatat bahwa Model A berhasil "menangkap maksudnya dengan baik" dengan menampilkan produk perawatan malam hari seperti night repair dan serum. Di sisi lain, Model B

menunjukkan kecenderungan untuk memberikan hasil yang sempit dan kurang relevan, di mana untuk kueri yang sama, model ini justru menampilkan "sunscreen dan ikat rambut" yang tidak sesuai dengan konteks.

C. Preferensi Model Keseluruhan

Secara keseluruhan, preferensi mayoritas (enam dari sembilan partisipan) cenderung berpihak pada Model A. Kualitas pengalaman pencarian secara keseluruhan menjadi pertimbangan yang lebih penting bagi pengguna dibandingkan sekadar kecocokan produk secara harfiah. Keunggulan utama Model A terletak pada kemampuannya menangani kueri yang kompleks dan ambigu. Pengguna, terutama yang memiliki pengetahuan domain, menghargai kemampuan Model A untuk memahami nuansa, memberikan hasil yang beragam namun tetap relevan secara kontekstual, dan menginterpretasikan bahasa natural. Hal ini menjadikan Model A sangat efektif untuk skenario pencarian eksplorasi (exploratory search), di mana pengguna belum memiliki produk spesifik dalam pikiran.

Meskipun demikian, bukan berarti Model B tidak memiliki keunggulan. Tiga dari sembilan partisipan secara konsisten memilih Model B, terutama untuk kueri pendek dan menengah. Kelemahan Model A yang kadang terlalu interpretatif menjadi kekuatan bagi Model B. Model B unggul dalam memberikan hasil yang lebih spesifik dan presisi ketika intensi pengguna sudah jelas. Seorang partisipan menyebutkan bahwa hasil dari Model B "lebih relevan dan lebih spesifik" untuk kueri yang tidak ambigu. Dengan demikian, Model B menunjukkan performa yang sangat baik untuk skenario pencarian lookup, di mana pengguna mencari produk yang sudah terdefinisi dengan baik dan menginginkan hasil yang langsung ke tujuan.

D. Ketidakselarasan *Similarity Score* vs Preferensi Pengguna

Salah satu temuan paling signifikan dalam penelitian ini adalah adanya ketidakselarasan antara similarity score yang dihasilkan sistem dengan persepsi relevansi oleh pengguna. Fenomena ini menjelaskan mengapa Model B, yang sering kali memberikan similarity score lebih tinggi untuk pencocokan langsung, tidak selalu menjadi pilihan utama. Perbedaan ini berakar pada arsitektur fundamental kedua model.

Model A (OpenAI text-embedding-3-large), dengan embedding 3072 dimensi, memiliki kapasitas representasi yang lebih besar. Keunggulannya adalah kemampuannya membangun "peta semantik" yang kaya dan bernuansa, memungkinkan koneksi antara konsep-konsep yang tidak terhubung secara leksikal. Inilah yang membuatnya superior dalam menangani ambiguitas dan kueri deskriptif. Namun, kelemahannya adalah representasi yang luas ini

terkadang dapat menghasilkan koneksi yang terlalu jauh, sehingga pada beberapa kasus menampilkan produk dari kategori yang sama sekali berbeda.

Model B (VoyageAI voyage-3-large), dengan embedding 1024 dimensi, menghasilkan representasi vektor yang lebih padat. Keunggulannya terletak pada efisiensi dan presisi untuk pencocokan yang jelas. Model ini menciptakan kelompok-kelompok makna yang lebih rapat, sehingga sangat andal dalam menemukan produk yang paling identik dengan kueri (*lookup search*) dan sering kali menghasilkan skor kemiripan yang lebih tinggi. Akan tetapi, kelemahannya adalah *information compression* yang berisiko menyebabkan "kehilangan informasi", di mana detail dan nuansa semantik dari kueri yang kompleks tidak tertangkap, membuatnya kurang efektif untuk pencarian eksplorasi.

Pada akhirnya, temuan ini menyimpulkan bahwa tidak ada satu model yang "terbaik" secara absolut. Pemilihan model merupakan sebuah keputusan strategis yang bergantung pada tujuan bisnis platform *e-commerce*: apakah ingin mengoptimalkan efisiensi dan kecepatan untuk pengguna yang tahu apa yang mereka cari (keunggulan Model B), atau ingin mengoptimalkan penemuan produk dan pengalaman eksplorasi bagi pengguna yang membutuhkan inspirasi (keunggulan Model A).

E. Implikasi Praktis untuk Platform *E-Commerce*

Penelitian ini memberikan rekomendasi solusi untuk sistem pencarian *e-commerce* yang menggunakan pendekatan ini untuk menjembatani antara *lookup search* dan *exploratory search* melalui pemilihan *model embedding* yang tepat sesuai kebutuhan. Untuk platform *e-commerce* yang fokus pada kecepatan *(lookup search)*, Model B (VoyageAI) lebih cocok karena baik dalam memberikan produk yang spesifik, membantu pengguna menemukan produk spesifik dengan cepat. Sebaliknya, untuk meningkatkan penemuan produk dan keterlibatan pengguna *(exploratory search),* Model A (OpenAI) lebih baik karena mampu memahami konteks, menangani ambiguitas, dan menyajikan hasil beragam. Penelitian ini memberikan implikasi praktis bagi pengembang untuk memilih *model embedding* yang sesuai dengan tujuan bisnis, apakah memaksimalkan kecepatan atau memperkaya pengalaman eksplorasi.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Penelitian ini menyimpulkan bahwa model OpenAI text-embedding-3-large (Model A) secara kualitatif lebih efektif dibandingkan VoyageAI voyage-3-large (Model B) menurut persepsi partisipan. Keunggulan ini bukan karena indikator kuantitatif, melainkan karena kemampuan model dalam memahami konteks, menangani kueri yang ambigu, dan menyajikan hasil yang relevan secara makna untuk mendukung hasil pencarian. Fenomena ini menunjukkan adanya ketidakselarasan antara indikator kuantitatif, di mana Model B sering menghasilkan *similarity score* lebih tinggi, dengan preferensi kualitatif pengguna. Perbedaan ini dapat dilihat pada arsitektur model dimensi vektor OpenAI yang lebih besar (3072) mampu menangkap representasi semantik yang lebih kaya dan bernuansa, sementara dimensi VoyageAI yang lebih kecil (1024) lebih efisien untuk pencocokan langsung namun berisiko kehilangan detail makna. Pada akhirnya, pemilihan *model embedding* adalah sebuah keputusan strategis yang melibatkan pertukaran *(trade-off)* antara presisi untuk pencarian spesifik dan kekayaan makna untuk pencarian eksplorasi.

5.2 Saran

Berdasarkan hasil analisis perbandingan *antara model embedding* OpenAI *text-embedding-3-large* dan VoyageAI *voyage-3-large* untuk pencarian semantik *e-commerce*, terdapat beberapa rekomendasi untuk pengembangan dan penelitian di masa depan. Berikut adalah empat saran yang dapat dijadikan acuan untuk kajian selanjutnya.

1. Menggunakan dataset *e-commerce* dengan metadata yang lebih besar dan lebih lengkap untuk melakukan pengujian. Penggunaan dataset yang lebih luas dan beragam akan memungkinkan pengujian yang lebih realistis.
2. Memperluas partisipan dari berbagai latar belakang. Hal ini penting untuk mendapatkan hasil yang lebih akurat karena basis pengguna *e-commerce* di dunia nyata sangat beragam dan tidak hanya terbatas pada domain tertentu.
3. Melakukan evaluasi terhadap *model embedding* yang secara spesifik dilatih (*fine-tuned*) pada domain *e-commerce*. Model yang terspesialisasi berpotensi menawarkan keseimbangan yang lebih baik antara presisi pencocokan dan pemahaman semantik dibandingkan model-model bertujuan umum yang digunakan dalam penelitian ini.

4. Melakukan pengujian sistem pada skala yang lebih besar atau skala produksi untuk mendapatkan hasil yang lebih akurat dan relevan.

DAFTAR PUSTAKA

Ahluwalia, A., Sutradhar, B., Ghosh, K., Yadav, I., Sheetal, A., & Patil, P. (2024). *Hybrid Semantic Search: Unveiling User Intent Beyond Keywords*. https://arxiv.org/pdf/2408.09236

Albujasim, Z., Inkpen, D., Han, X., & Guo, Y. (2023). Improving Word Embedding Using Variational Dropout. *The International FLAIRS Conference Proceedings*, *36*. https://doi.org/10.32473/FLAIRS.36.133326

Almunawar, M. N., Anshari, M., & Lim, S. A. (2021). A Framework for Observing Digital Marketplace. *International Journal of Hyperconnectivity and the Internet of Things*, *5*(2), 57–73. https://doi.org/10.4018/IJHIOT.2021070104

Asudani, D. S., Nagwani, N. K., & Singh, P. (2023). Impact of word embedding models on text analytics in deep learning environment: a review. *Artificial Intelligence Review*, *56*(9), 10345–10425. https://doi.org/10.1007/s10462-023-10419-1

Bianchi, F., Tagliabue, J., & Yu, B. (2021). Query2Prod2Vec Grounded Word Embeddings for eCommerce. *NAACL-HLT 2021 - 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Industry Papers*, 154–162. https://doi.org/10.18653/v1/2021.naacl-industry.20

Calleja, M. O., & Willoughby, A. R. (2023). The effects of search-irrelevant working memory content on visual search. *Attention, Perception, &Amp; Psychophysics*, *85*(2), 293–300. https://doi.org/10.3758/S13414-022-02634-9

Carrara, F., Vadicamo, L., Gennaro, C., & Amato, G. (2022). Approximate Nearest Neighbor Search on Standard Search Engines. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *13590 LNCS*, 214–221. https://doi.org/10.1007/978-3-031-17849-8_17

Choi, E., & Lee, K. C. (2019). Effect of Trust in Domain-Specific Information of Safety, Brand Loyalty, and Perceived Value for Cosmetics on Purchase Intentions in Mobile E-Commerce Context. *Sustainability 2019, Vol. 11, Page 6257*, *11*(22), 6257. https://doi.org/10.3390/SU11226257

Fife, A., Ramsey, C., Esculier, J. F., & Hébert-Losier, K. (2023). How do road runners select their shoes? A systematic review. *Footwear Science*, *15*(2), 103–112. https://doi.org/10.1080/19424280.2023.2180543;REQUESTEDJOURNAL:JOURNAL:TFWS20;JOURNAL:JOURNAL:TFWS20;WGROUP:STRING:PUBLICATION

Fife, A., Ramsey, C., Esculier, J. F., & Hébert-Losier, K. (2024). How do runners select their shoes? An in-store experience. *Footwear Science*, *16*(3), 191–199. https://doi.org/10.1080/19424280.2024.2353597;PAGE:STRING:ARTICLE/CHAPTER

Ghali, M. K., Farrag, A., Won, D., & Jin, Y. (2025). Enhancing knowledge retrieval with in-context learning and semantic search through generative AI. *Knowledge-Based Systems*, *311*. https://doi.org/10.1016/j.knosys.2025.113047

Gupta, A. (2025). *Virtual Online Fitness Market Research Report Forecast 2034*. https://www.marketresearchfuture.com/reports/virtual-online-fitness-market-24441. https://www.marketresearchfuture.com/reports/virtual-online-fitness-market-24441

Johnson, J., Douze, M., & Jegou, H. (2021). Billion-Scale Similarity Search with GPUs. *IEEE Transactions on Big Data*, *7*(3), 535–547. https://doi.org/10.1109/TBDATA.2019.2921572

Klotzman, V. (2024). *Enhancing Automated Medical Coding: Evaluating Embedding Models for ICD-10-CM Code Mapping*. https://doi.org/10.1101/2024.07.02.24309849

Lee, S. Y., & Lee, K. (2018). Factors that influence an individual's intention to adopt a wearable healthcare device: The case of a wearable fitness tracker. *Technological Forecasting and Social Change*, *129*, 154–163. https://doi.org/10.1016/J.TECHFORE.2018.01.002

Liu, R. (2024). Exploring the Impact of Word2Vec Embeddings Across Neural Network Architectures for Sentiment Analysis. *Applied and Computational Engineering*, *97*, 93–98. https://doi.org/10.54254/2755-2721/97/2024MELB0085

Liu, X., Wang, J., Sun, J., Yuan, X., Dong, G., Di, P., Wang, W., & Wang, D. (2023). *Prompting Frameworks for Large Language Models: A Survey*. https://arxiv.org/pdf/2311.12785

Ma, L., Zhang, R., Han, Y., Yu, S., Wang, Z., Ning, Z., Zhang, J., Xu, P., Li, P., Ju, W., Chen, C., Wang, D., Liu, K., Wang, P., Wang, P., Fu, Y., Liu, C., Zhou, Y., & Lu, C.-T. (2025). *A Comprehensive Survey on Vector Database: Storage and Retrieval Technique, Challenge*.

Maithili, K., Naveen Raja, S. M., Kumar, R. P. R., & Koli, S. (2023). A Survey (NLP) Natural Language Processing and Transactions on (NNL) Neural Networks and learning Systems. *E3S Web of Conferences*, *430*, 01148. https://doi.org/10.1051/E3SCONF/202343001148

Mitchell, T. (2024, Desember 20). *What are Embedding Models? An Overview - The Couchbase Blog*. https://www.couchbase.com/blog/embedding-models/?utm_medium=organic_social&utm_source=twitter

Naqvi, S. M. R., Ghufran, M., Varnier, C., Nicod, J. M., & Zerhouni, N. (2025). Enhancing semantic search using ontologies: A hybrid information retrieval approach for industrial text. *Journal of Industrial Information Integration*, *45*. https://doi.org/10.1016/j.jii.2025.100835

Neelakantan, A., Xu, T., Puri, R., Radford, A., Han, J. M., Tworek, J., Yuan, Q., Tezak, N., Kim, J. W., Hallacy, C., Heidecke, J., Shyam, P., Power, B., Nekoul, T. E., Sastry, G., Krueger, G., Schnurr, D., Such, F. P., Hsu, K., … Weng, L. (t.t.). *Text and Code Embeddings by Contrastive Pre-Training*. https://doi.org/10.48550/ARXIV.2201.10005

Novotná, T. (2021). Human Evaluation Experiment of Legal Information Retrieval Methods. *Frontiers in Artificial Intelligence and Applications*, *346*, 131–137. https://doi.org/10.3233/FAIA210328

Parab, L. (2023). *Amazon Products Sales Dataset 2023*. Kaggle.

Peikos, G., & Pasi, G. (2024). A systematic review of multidimensional relevance estimation in information retrieval. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *14*(5), e1541. https://doi.org/10.1002/WIDM.1541;REQUESTEDJOURNAL:JOURNAL:19424795;PAGEGROUP:STRING:PUBLICATION

Rawis, S. O., Nasution, R. A., Hadiansyah, L. S., Adiani, W., & Aprianingsih, A. (2022). Systematic Literature Review of Switching Behavior in Service Industry. *Binus Business Review*, *13*(1), 1–17. https://doi.org/10.21512/BBR.V13I1.7618

Ruthven, I. (2021). Resonance and the experience of relevance. *Journal of the Association for Information Science and Technology*, *72*(5), 554–569. https://doi.org/10.1002/ASI.24424;JOURNAL:JOURNAL:19366108;WEBSITE:WEBSITE:ASISTDL;REQUESTEDJOURNAL:JOURNAL:23301643;WGROUP:STRING:PUBLICATION

Sajid, S., Rashid, R. M., & Haider, W. (2022). Changing Trends of Consumers' Online Buying Behavior During COVID-19 Pandemic With Moderating Role of Payment Mode and Gender. *Frontiers in Psychology*, *13*. https://doi.org/10.3389/fpsyg.2022.919334

Seinen, T. M., Fridgeirsson, E. A., Ioannou, S., Jeannetot, D., John, L. H., Kors, J. A., Markus, A. F., Pera, V., Rekkas, A., Williams, R. D., Yang, C., Van Mulligen, E. M., & Rijnbeek,

P. R. (2022). Use of unstructured text in prognostic clinical prediction models: a systematic review. *Journal of the American Medical Informatics Association*, *29*(7), 1292–1302. https://doi.org/10.1093/JAMIA/OCAC058

Singh, S. (2025). *E Commerce Personal Care Product Market Size, Share, Trends and Analysis 2034*. https://www.marketresearchfuture.com/reports/e-commerce-personal-care-product-market-36683. https://www.marketresearchfuture.com/reports/e-commerce-personal-care-product-market-36683

Sultana, T., Mandal, A. K., Saha, H., Sultan, M. N., & Hossain, M. D. (2024). Intent Identification by Semantically Analyzing the Search Query. *Modelling*, *5*(1), 292–314. https://doi.org/10.3390/MODELLING5010016

Sun, X., Gao, Y., Sutcliffe, R., Guo, S. X., Wang, X., & Feng, J. (2021). Word Representation Learning Based on Bidirectional GRUs With Drop Loss for Sentiment Classification. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *51*(7), 4532–4542. https://doi.org/10.1109/TSMC.2019.2940097

Supabase. (2025, Juli 8). *Supabase.com*. supabase.com/docs

Tang, Y., & Yang, Y. (2025). *FinMTEB: Finance Massive Text Embedding Benchmark*. http://arxiv.org/abs/2502.10990

Urman, A., & Makhortykh, M. (2021). You Are How (and Where) You Search? Comparative Analysis of Web Search Behaviour Using Web Tracking Data. *Journal of Computational Social Science*, *6*(2), 741–756. https://doi.org/10.1007/s42001-023-00208-9

Urman, A., Makhortykh, M., & Ulloa, R. (2021). The Matter of Chance: Auditing Web Search Results Related to the 2020 U.S. Presidential Primary Elections Across Six Search Engines. Dalam *Social Science Computer Review*. https://doi.org/10.1177/08944393211006863

van Hoof, M., Meppelink, C. S., Moeller, J., & Trilling, D. (2024). Searching differently? How political attitudes impact search queries about political issues. *New Media &Amp; Society*, *26*(7), 3728–3750. https://doi.org/10.1177/14614448221104405

What is Extreme Programming (XP)? - GeeksforGeeks. (2025, Juli 11). https://www.geeksforgeeks.org/software-engineering/software-engineering-extreme-programming-xp/

Widianto, A., Pebriyanto, E., Fitriyanti, F., & Marna, M. (2024). Document Similarity Using Term Frequency-Inverse Document Frequency Representation and Cosine Similarity.

Journal of Dinda : Data Science, Information Technology, and Data Analytics, *4*(2), 149–153. https://doi.org/10.20895/DINDA.V4I2.1589

Yuan, Y., Abbasiantaeb, Z., Deng, Y., & Aliannejadi, M. (2025). Query Understanding in LLM-based Conversational Information Seeking. *Companion Proceedings of the ACM on Web Conference 2025*, 73–76. https://doi.org/10.1145/3701716.3715869

Zhan, J., Mao, J., Liu, Y., Guo, J., Zhang, M., & Ma, S. (t.t.). *Optimizing Dense Retrieval Model Training with Hard Negatives*. https://doi.org/10.48550/ARXIV.2104.08051

Zhang, Y., & Chai, Y. (1M). How Pinduoduo's E-Commerce Experience Shapes Customer Satisfaction and Loyalty: Leveraging Social Networks. *https://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/JECO.381236*, *23*(1), 1–17. https://doi.org/10.4018/JECO.381236

LAMPIRAN