

RANCANG BANGUN SISTEM REKOMENDASI PROPERTI UNTUK WEBSITE PERUSAHAAN REAL ESTATE

LAPORAN

Diajukan untuk memenuhi bagian dari syarat kelulusan
mata kuliah Praktik Pengalaman Lapangan



Oleh:

AHMAD IZZUDDIN

1908919

**PROGRAM STUDI ILMU KOMPUTER
DEPARTEMEN PENDIDIKAN ILMU KOMPUTER
FAKULTAS PENDIDIKAN MATEMATIKA DAN ILMU PENGETAHUAN
ALAM
UNIVERSITAS PENDIDIKAN INDONESIA
2023**

LEMBAR PENGESAHAN

**RANCANG BANGUN SISTEM REKOMENDASI PROPERTI UNTUK
WEBSITE PERUSAHAAN REAL ESTATE**

AHMAD IZZUDDIN

1908919

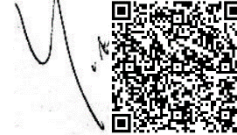
Menyetujui,

Pembimbing Lapangan



Arif Pribadi

Dosen Pembimbing



Dr. Yudi Wibisono, M.T.
NIP. 197507072003121003

Mengetahui,

Ketua Program Studi Ilmu Komputer



Dr. Rani Megasari, M.T.

NIP. 198705242014042002

KATA PENGANTAR

Puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa, karena atas berkat dan rahmat-Nya laporan ini dapat diselesaikan. Laporan ini merupakan salah satu syarat untuk menyelesaikan program Praktik Pengalaman Lapangan di Program Studi Ilmu Komputer, Fakultas Pendidikan Matematika dan Ilmu Pengetahuan Alam, Universitas Pendidikan Indonesia.

Laporan ini menjabarkan kegiatan Praktik Pengalaman Lapangan yang dilakukan selama tiga bulan di PT. Vanz Inovatif Teknologi. Laporan ini berisi tentang latar belakang perusahaan, deskripsi pekerjaan yang dilakukan, serta pengalaman yang diperoleh selama mengikuti program Program Praktik Lapangan.

Penulis menyadari bahwa laporan ini masih jauh dari sempurna, oleh karena itu penulis sangat mengharapkan masukan dan saran yang bersifat membangun dari pembaca agar laporan ini dapat diperbaiki di masa yang akan datang.

Akhir kata, penulis mengucapkan terima kasih kepada PT. Vanz Inovatif Teknologi yang telah memberikan kesempatan kepada penulis untuk mengikuti program Praktik Pengalaman Lapangan serta kepada semua pihak yang telah membantu dalam penyelesaian laporan ini.

Bandung, 22 Juni 2023

Penulis

DAFTAR ISI

LEMBAR PENGESAHAN	i
KATA PENGANTAR	ii
DAFTAR ISI.....	iii
BAB I PENDAHULUAN.....	1
1.1. Latar Belakang	1
1.2. Identifikasi Masalah	2
1.3. Tujuan.....	2
1.4. Ruang Lingkup.....	3
1.5. Sistematika Penulisan.....	3
BAB II GAMBARAN UMUM PERUSAHAAN DAN KAJIAN PUSTAKA.....	4
2.1. Profil Perusahaan.....	4
2.2. Kajian Pustaka.....	4
2.2.1. Bisnis Real Estate	4
2.2.2. Sistem Rekomendasi.....	5
2.2.3. NestJS	9
2.2.4. PostgreSQL	10
2.2.5. Redis	13
2.2.6. MySQL	13
2.2.7. NextJS	14
BAB III ANALISIS DAN PERANCANGAN	16
3.1. Analisis.....	16
3.1.1. Use Case Diagram.....	16
3.1.2. Activity Diagram	17
3.1.3. Spesifikasi Data	23
3.1.4. Menghindari Cold-Start Problem.....	24

3.2. Perancangan	24
3.2.1. Entity Relationship Diagram.....	24
3.2.2. Desain Sistem Rekomendasi.....	26
3.2.3. Deployment Diagram.....	28
BAB IV HASIL KEGIATAN PLA	30
4.1. Lingkungan Implementasi.....	30
4.2. Hasil Implementasi.....	32
4.2.1. Desain User Interface.....	32
4.2.2. Website Rekomendasi.....	33
4.2.3. API Rekomendasi	37
4.2.4. Logika Rekomendasi	39
4.3. Pengujian.....	40
BAB V PENUTUP	43
5.1. Kesimpulan.....	43
5.2. Saran.....	43
DAFTAR PUSTAKA	45
LAMPIRAN A.....	47
LAMPIRAN B	62
LAMPIRAN C.....	64
LAMPIRAN D.....	65

BAB I

PENDAHULUAN

1.1. Latar Belakang

Pasar properti di Indonesia mengalami perkembangan yang pesat seiring dengan pertumbuhan ekonomi yang semakin baik (Daryanto et al., 2018). Banyak orang mulai berinvestasi dalam bidang properti, baik untuk tempat tinggal maupun sebagai instrumen investasi. Karena sifatnya yang merupakan aset jangka panjang, pembelian properti dapat menjadi keputusan yang sangat penting dan memerlukan pertimbangan yang matang. Hal ini dapat mempersulit pengalaman pencarian properti bagi pembeli yang belum memiliki pengalaman yang cukup di bidang properti.

Kondisi ekonomi yang disebabkan pandemi COVID-19 juga mendorong semakin banyaknya penggunaan website real estate untuk mencari dan membeli properti (Arkandana et al., 2021). Namun, pada saat yang sama, meningkatnya jumlah listing pada website real estate juga mempersulit pengguna dalam memilih properti yang sesuai dengan kebutuhan dan preferensi mereka. Hal ini menimbulkan tantangan bagi pengelola website real estate untuk memberikan pengalaman pencarian properti yang lebih efisien dan efektif atau dalam kata lain lebih personal.

Dalam situasi seperti ini, sistem rekomendasi dapat menjadi solusi untuk membantu pengguna dalam mencari dan memilih properti yang sesuai dengan kebutuhan mereka (Nanou et al., 2010). Sistem rekomendasi dapat memberikan rekomendasi properti yang lebih akurat dan relevan dengan preferensi pengguna berdasarkan data historis dan informasi yang dikumpulkan dari pengguna. Sistem ini dapat membantu meningkatkan kepuasan pengguna dan mengoptimalkan pengalaman pencarian properti di website perusahaan real estate. Sistem rekomendasi juga dapat berdampak positif terhadap keinginan pengguna untuk melakukan transaksi (Ku & Tai, 2013).

Selain tantangan dalam memilih properti yang sesuai dengan preferensi pengguna, masih ada beberapa masalah yang dihadapi oleh pengguna pada website perusahaan real estate. Salah satunya adalah keterbatasan fitur pencarian yang hanya mengandalkan full text search berdasarkan similarity. Hal ini menyulitkan pengguna untuk menemukan properti yang sesuai dengan kriteria yang mereka inginkan, terutama jika kriteria tersebut terdiri dari banyak

variabel. Sebagai contoh, jika dicari rumah di daerah Sudirman Jakarta, maka kemungkinan besar pengunjung juga terbuka untuk melihat rumah lain di daerah Senayan.

Oleh karena itu, dengan menghadirkan sistem rekomendasi pada website perusahaan real estate, diharapkan dapat membantu mengatasi masalah tersebut dan memberikan pengalaman pencarian properti yang lebih baik bagi pengguna. Dengan sistem rekomendasi, pengguna dapat melihat rekomendasi properti yang personalisasi dan relevan dengan preferensi mereka, sehingga proses pencarian properti dapat menjadi lebih efisien dan efektif.

Berdasarkan hal tersebut, sistem rekomendasi untuk website perusahaan real estate sangat penting untuk membantu memperbaiki pengalaman pengguna dan meningkatkan efisiensi dalam pencarian properti. Laporan ini akan membahas rancangan dan implementasi sistem rekomendasi yang dapat digunakan pada website perusahaan real estate, sehingga diharapkan dapat meningkatkan kualitas pengalaman pencarian properti bagi pengguna.

1.2. Identifikasi Masalah

Kegiatan Program Praktik Lapangan ini dilaksanakan berdasarkan kebutuhan yang ada di PT. Vanz Inovatif Teknologi (qbit). Berdasarkan ini, permasalahan yang akan diselesaikan adalah sistem rekomendasi sebagai bagian dari pengembangan infrastruktur web klien perusahaan real estate. Dari permasalahan di atas, terdapat beberapa rumusan masalah antara lain adalah:

1. Bagaimana rancangan untuk sebuah sistem rekomendasi bagi website perusahaan real estate?
2. Bagaimana pengembangan dan mekanisme implementasi sistem rekomendasi bagi website perusahaan real estate yang dalam pengembangan?
3. Bagaimana pengujian dan evaluasi sistem rekomendasi bagi website perusahaan real estate yang dalam pengembangan?

1.3. Tujuan

Berdasarkan rumusan masalah yang dijelaskan sebelumnya, tujuan dari kegiatan PPL ini adalah:

1. Merancang sebuah sistem rekomendasi bagi website perusahaan real estate.
2. Mengembangkan dan Mengimplementasi sistem rekomendasi bagi website perusahaan real estate yang dalam pengembangan.
3. Menguji dan Mengevaluasi sistem rekomendasi bagi website perusahaan real estate yang dalam pengembangan.

1.4. Ruang Lingkup

Kegiatan PPL ini dibatasi pada ruang lingkup sebagai berikut:

1. Implementasi antramuka (UI) dengan sistem rekomendasi pada website menggunakan nextjs dan reactjs.
2. Implementasi sistem rekomendasi pada backend dalam bentuk API dan infrastruktur database menggunakan nestjs, postgresql, dan redis.
3. Pengembangan sistem rekomendasi untuk website perusahaan real estate sebagai bagian dari proyek pengembangan website perusahaan real estate dengan tim dari QBIT.

1.5. Sistematika Penulisan

Laporan PPL ini terdiri atas 5 bab, ditambah dengan halaman awal yang terdiri atas lembar judul, kata pengantar, daftar isi, daftar tabel, dan daftar gambar. Batang tubuh laporan PPL ini mengikuti struktur organisasi berikut:

BAB I PENDAHULUAN

Bab ini terdiri dari latar belakang, identifikasi masalah, tujuan, ruang lingkup, serta sistematika penulisan. Dijelaskan ide dan permasalahan yang menjadi dasar pelaksanaan kegiatan PPL.

BAB II GAMBARAN UMUM PERUSAHAAN DAN KAJIAN PUSTAKA

Membahas secara singkat lembaga tempat kegiatan PPL dilaksanakan dan informasi lain berupa teori, dokumentasi, serta hasil penelitian lain yang mendukung penyelesaian proyek ini.

BAB III ANALISIS DAN PERANCANGAN

Membahas metode penelitian yang digunakan, analisis, serta perancangan secara umum dan secara khusus. Analisis yang digunakan meliputi use case diagram dan activity diagram dan dilampirkan perancangan.

BAB IV HASIL KEGIATAN PPL

Membahas luaran yang diperoleh selama menjalankan kegiatan PPL serta membahas sistem yang telah dibangun dimulai dari persiapan, hasil akhir dan pengujian.

BAB V PENUTUP

Berisi tentang pencapaian tujuan dari sistem yang dikembangkan. Dipaparkan pula saran berisi hal-hal yang dirasa perlu dilakukan pada pengembangan selanjutnya.

BAB II

GAMBARAN UMUM PERUSAHAAN DAN KAJIAN PUSTAKA

2.1. Profil Perusahaan

PT. Vanz Inovatif Teknologi atau lebih umum dikenal sebagai qbit adalah perusahaan teknologi yang memiliki spesialisasi dalam quantum, blockchain, IoT, dan transportasi. Perusahaan ini berfokus pada pengembangan solusi teknologi untuk membantu bisnis dalam mengoptimalkan operasi mereka. qbit menawarkan berbagai layanan termasuk pengembangan perangkat lunak, desain produk, blockchain, Internet of Things, manage services, dan games. PT. Vanz Inovatif Teknologi didirikan pada tahun 2017 oleh sekelompok ahli teknologi yang memiliki pengalaman dalam pengembangan perangkat lunak dan bisnis. Perusahaan ini berbasis di Ruko Sentra Niaga Green Lake L7, Jl. Green Lake City, Boulevard, Cengkareng, DKI Jakarta 11750. PT. Vanz Inovatif Teknologi saat ini memiliki logo sebagaimana pada Gambar 2.1.



Gambar 2.1. Logo PT. Vanz Inovatif Teknologi

PT. Vanz Inovatif Teknologi memiliki struktur dengan core team yang terdiri atas:

1. CEO: Ronald Gunawan
2. COO: Irvan Chandra
3. CTO: Evan Leonardi

Secara garis besar, tim developer di qbit terdiri atas project manager, lead developer, tim developer, tim QA, dan tim UI/UX. Tim developer sendiri terbagi berdasarkan project. Sedangkan tim QA dan tim UI/UX mengerjakan task berdasarkan timnya.

2.2. Kajian Pustaka

2.2.1. Bisnis Real Estate

Bisnis real estate atau properti adalah bisnis yang bergerak di bidang jual-beli, sewa-menyewa, dan pengelolaan properti seperti rumah, apartemen, ruko, gedung perkantoran, dan lain sebagainya. Bisnis ini memainkan peran penting dalam perekonomian Indonesia karena investasi di bidang properti masih menjadi pilihan yang menarik bagi banyak orang.

Sebagai perusahaan di era digital, perusahaan real estat beradaptasi dengan perkembangan teknologi dan tren pasar terbaru dengan memanfaatkan platform online untuk memperluas jangkauan bisnis. Ini termasuk website yang dapat dikunjungi oleh beberapa jenis pengguna mulai dari klien jual/beli, titip sewa, dan potensi mitra.

Untuk melindungi privasi semua klien Qbit, nama atau adanya klien yang mungkin bersangkutan atau tidak dengan proyek ini tidak dapat disebutkan dalam laporan ini.

2.2.2. Sistem Rekomendasi

Masalah yang ingin diselesaikan oleh sistem rekomendasi adalah masalah dari banyaknya informasi atau pilihan yang tersedia dalam suatu platform (Al-Oud, 2008). Masalah terlalu banyak pilihan adalah masalah yang muncul ketika ada terlalu banyak opsi atau alternatif yang tersedia, sehingga membuat pengguna sulit untuk memilih yang terbaik. Dalam konteks bisnis real estate, masalah ini dapat terjadi ketika ada terlalu banyak listing properti yang tersedia, sehingga calon pembeli atau penyewa kesulitan untuk menemukan properti yang sesuai dengan kebutuhan dan preferensi mereka.

Sistem search engine adalah salah satu solusi yang umum digunakan untuk membantu pengguna menemukan properti yang sesuai dengan kriteria mereka. Namun, meskipun search engine dapat memberikan hasil pencarian yang relevan, namun masih ada beberapa limitasi (Al-Oud, 2008). Beberapa limitasi tersebut antara lain:

1. **Keyword-based search:** Sistem search engine biasanya hanya dapat melakukan pencarian berdasarkan keyword atau kata kunci yang dimasukkan oleh pengguna. Hal ini bisa menjadi limitasi karena pengguna harus tahu persis apa yang mereka cari dan bagaimana mengungkapkan keinginan mereka dengan kata-kata.
2. **Keterbatasan dalam menampilkan hasil pencarian:** Sistem search engine biasanya hanya dapat menampilkan hasil pencarian dalam jumlah terbatas. Hal ini bisa menjadi limitasi jika ada terlalu banyak properti yang tersedia dan pengguna ingin melihat semua opsi yang tersedia.
3. **Tidak bisa memberikan rekomendasi personalisasi:** Sistem search engine hanya dapat memberikan hasil pencarian berdasarkan kata kunci dan kriteria yang dimasukkan oleh pengguna. Hal ini berbeda dengan sistem rekomendasi yang dapat memberikan rekomendasi yang lebih personalisasi berdasarkan preferensi dan perilaku pengguna.

Karena itulah, sistem rekomendasi dapat menjadi solusi yang lebih efektif dalam membantu pengguna menemukan properti yang sesuai dengan preferensi dan kebutuhan mereka.

Dalam hal ini, sistem rekomendasi bertujuan untuk membantu pengguna dalam mempersempit pilihan properti yang tersedia (Aggarwal, 2016; Al-Oud, 2008; Nanou et al., 2010), sehingga pengguna dapat lebih efisien dalam mencari dan menemukan properti yang cocok dengan kebutuhan dan preferensi mereka. Sistem rekomendasi dapat membantu mengurangi waktu yang dibutuhkan pengguna untuk mencari properti yang sesuai dengan kriteria mereka, serta meningkatkan kemungkinan bahwa pengguna akan menemukan properti yang cocok dengan preferensi mereka.

Sistem rekomendasi adalah sebuah teknologi yang digunakan untuk memberikan rekomendasi atau rekomendasi produk, layanan, atau konten yang sesuai dengan preferensi atau minat pengguna. Sistem ini berfungsi dengan cara menganalisis data pengguna seperti riwayat transaksi, riwayat penelusuran, dan perilaku pengguna lainnya untuk memahami preferensi dan minat mereka.

Sistem rekomendasi biasanya menggunakan algoritma machine learning dan analisis data untuk menemukan pola dan tren dalam perilaku pengguna. Ada beberapa jenis sistem rekomendasi, termasuk sistem berbasis konten, sistem berbasis kolaboratif, dan sistem rekomendasi hybrid.

Sistem rekomendasi sangat populer di berbagai industri, termasuk e-commerce, layanan streaming, dan sosial media. Mereka membantu meningkatkan pengalaman pengguna dengan menyediakan rekomendasi yang relevan dan dapat membantu pengguna menemukan produk, layanan, atau konten yang sesuai dengan kebutuhan dan minat mereka. Selain itu, sistem rekomendasi juga membantu perusahaan meningkatkan konversi dan penjualan dengan mempromosikan produk dan layanan mereka kepada pengguna yang berpotensi tertarik.

2.2.2.1. Collaborative Filtering

Model Collaborative Filtering adalah model yang digunakan dalam sistem rekomendasi untuk menghasilkan rekomendasi berdasarkan perilaku pengguna yang serupa (Aggarwal, 2016; Al-Oud, 2008). Model ini mencoba untuk menemukan pola dan preferensi yang sama antara pengguna berbeda untuk memperoleh rekomendasi yang sesuai untuk setiap pengguna.

Dalam model Collaborative Filtering, sistem rekomendasi menciptakan matriks yang merepresentasikan hubungan antara pengguna dan item. Baris matriks merepresentasikan

pengguna dan kolom merepresentasikan item. Setiap sel pada matriks merepresentasikan rating yang diberikan oleh pengguna pada item tertentu.

Model ini memiliki dua jenis, yaitu User-Based Collaborative Filtering dan Item-Based Collaborative Filtering. User-Based Collaborative Filtering bekerja dengan cara mencari pengguna dengan perilaku dan preferensi yang serupa, lalu merekomendasikan item yang disukai oleh pengguna tersebut. Sedangkan, Item-Based Collaborative Filtering mencari item yang memiliki hubungan dekat dengan item yang disukai oleh pengguna, lalu merekomendasikan item tersebut ke pengguna.

Namun, model Collaborative Filtering juga memiliki kelemahan. Salah satu kelemahannya adalah cold-start problem, yaitu saat sistem rekomendasi sulit memberikan rekomendasi yang baik pada pengguna baru yang belum memiliki sejarah interaksi dengan sistem. Selain itu, model ini juga membutuhkan banyak data interaksi dari pengguna agar dapat memberikan rekomendasi yang akurat, dan dapat terjadi masalah jika tidak ada cukup data interaksi.

2.2.2.2. Content-based Filtering

Content-based filtering adalah teknik dalam sistem rekomendasi yang menggunakan informasi tentang atribut atau karakteristik item untuk merekomendasikan item yang mirip dengan item yang disukai pengguna sebelumnya (Aggarwal, 2016; Al-Oud, 2008). Teknik ini didasarkan pada asumsi bahwa pengguna cenderung menyukai item yang memiliki atribut atau karakteristik serupa dengan item yang telah mereka sukai sebelumnya.

Dalam content-based filtering, item yang disukai oleh pengguna sebelumnya dianalisis dan fitur atau atribut dari item tersebut diekstraksi. Kemudian, sistem mencari item lain dengan atribut atau karakteristik serupa dan merekomendasikannya kepada pengguna.

Contoh implementasi content-based filtering adalah pada platform streaming musik seperti Spotify. Sistem merekomendasikan lagu berdasarkan genre, artis, dan lagu yang pernah didengarkan oleh pengguna sebelumnya. Jika pengguna sering mendengarkan lagu-lagu dengan genre pop, maka sistem akan merekomendasikan lagu-lagu pop lainnya dengan artis atau karakteristik serupa.

2.2.2.3. Knowledge-based Filtering

Knowledge-based recommender system (sistem rekomendasi berbasis pengetahuan) adalah sistem yang memberikan rekomendasi produk atau layanan kepada pengguna berdasarkan spesifikasi yang disebutkan oleh pengguna (Aggarwal, 2016). Sistem ini berbeda dengan

sistem rekomendasi lainnya seperti collaborative filtering dan content-based filtering yang hanya mengandalkan data riwayat pengguna atau kesamaan produk dalam memberikan rekomendasi.

Knowledge-based recommender system menggunakan knowledge base sebagai sumber informasi untuk menghasilkan rekomendasi. Knowledge base adalah database pengetahuan yang berisi aturan dan fungsi kesamaan (similarity functions) yang digunakan untuk mengambil informasi tentang produk atau layanan yang dibutuhkan pengguna.

Knowledge-based recommender system digunakan khususnya untuk produk yang jarang dibeli seperti real estate, mobil, permintaan wisata, layanan keuangan, atau barang mewah yang mahal. Karena produk tersebut jarang dibeli, tidak selalu tersedia banyak rating atau ulasan dari pengguna, sehingga sulit untuk menggunakan collaborative filtering atau content-based filtering dalam memberikan rekomendasi.

Dalam knowledge-based recommender system, pengguna dapat secara eksplisit menentukan spesifikasi yang dibutuhkan. Ada dua jenis knowledge-based recommender system, yaitu constraint-based recommender system dan case-based recommender system.

1. Constraint-based recommender system: dalam sistem ini, pengguna menentukan batasan atau persyaratan (constraints) pada atribut produk yang dibutuhkan. Contoh interface dari sistem ini bisa dilihat pada gambar 1.5 dalam teks di atas. Aturan domain-specific digunakan untuk mencocokkan persyaratan pengguna dengan atribut produk yang tersedia. Aturan-aturan ini bisa berupa batasan domain-specific pada atribut produk (misalnya "mobil sebelum tahun 1970 tidak memiliki cruise control") atau hubungan antara atribut pengguna dengan atribut produk (misalnya "investor tua tidak berinvestasi pada produk ultra high-risk").
2. Case-based recommender system: dalam sistem ini, pengguna menentukan kasus atau titik referensi yang spesifik. Fungsi kesamaan (similarity metrics) didefinisikan pada atribut produk untuk mencari produk yang serupa dengan kasus yang ditentukan. Produk yang ditemukan sering digunakan sebagai kasus baru dengan modifikasi interaktif oleh pengguna.

Knowledge-based recommender system menggunakan beberapa metode untuk interaktifitas, yaitu diantaranya conversational system, search-based system, atau navigation-based system. Conversational system memungkinkan preferensi pengguna ditentukan secara iteratif melalui feedback loop, sedangkan search-based system menggunakan urutan pertanyaan yang telah

ditentukan atau antarmuka pencarian yang disediakan oleh sistem. Dalam navigation-based system, pengguna menentukan sejumlah permintaan perubahan pada item yang saat ini direkomendasikan. Melalui serangkaian permintaan perubahan secara iteratif, dimungkinkan untuk mencapai item yang diinginkan.

2.2.2.4. Hybrid or Ensemble Approaches

Sistem rekomendasi hybrid dan ensemble adalah dua jenis sistem yang mampu menggabungkan beberapa jenis input yang berbeda untuk memberikan rekomendasi yang lebih baik (Aggarwal, 2016; Al-Oud, 2008). Tiga sistem sebelumnya yaitu sistem collaborative filtering, content-based, dan knowledge-based masing-masing menggunakan sumber input yang berbeda dan mampu bekerja dengan baik dalam skenario yang berbeda pula. Misalnya, sistem collaborative filtering bergantung pada rating komunitas, sedangkan sistem content-based bergantung pada deskripsi teks dan rating pengguna. Sementara itu, sistem knowledge-based bergantung pada interaksi dengan pengguna di dalam basis pengetahuan.

Berbeda dengan sistem rekomendasi tunggal yang hanya mengandalkan satu jenis input, sistem rekomendasi hybrid dan ensemble dapat menggabungkan beberapa jenis input yang berbeda untuk memberikan hasil yang lebih baik. Dalam banyak kasus, ketika berbagai jenis input tersedia, kita dapat menggunakan berbagai jenis sistem rekomendasi untuk tugas yang sama. Dalam hal ini, ada banyak peluang untuk melakukan hibridisasi, di mana berbagai aspek dari berbagai jenis sistem dikombinasikan untuk mencapai yang terbaik dari semua dunia.

Sistem rekomendasi hybrid erat hubungannya dengan bidang analisis ensemble, di mana kekuatan dari beberapa jenis algoritma pembelajaran mesin digabungkan untuk menciptakan model yang lebih kuat. Sistem rekomendasi ensemble mampu menggabungkan tidak hanya kekuatan dari beberapa sumber data, tetapi juga mampu meningkatkan efektivitas kelas tertentu sistem rekomendasi (misalnya sistem kolaboratif) dengan menggabungkan beberapa model dari jenis yang sama. Skenario ini tidak berbeda jauh dengan analisis ensemble dalam bidang klasifikasi data.

2.2.3. NestJS

NestJS adalah sebuah framework untuk membangun aplikasi web berbasis Node.js dengan gaya arsitektur MVC (Model-View-Controller) yang mengikuti pola desain Object Oriented Programming (OOP) (NestJS, 2023). NestJS dikembangkan untuk mempercepat proses pengembangan aplikasi dengan menyediakan fitur-fitur yang membantu dalam membangun aplikasi yang scalable dan maintainable.

NestJS berfokus pada pengembangan aplikasi server-side dengan memanfaatkan teknologi modern seperti Typescript, HTTP Server Express, dan Websockets. Framework ini memungkinkan developer untuk mengembangkan aplikasi dengan mudah dan cepat dengan menggunakan Dependency Injection, Pipes, Middleware, dan Guards.

Dalam NestJS, semua fitur yang disediakan dibangun di atas konsep-modul. Modul adalah bagian dari aplikasi yang memiliki tujuan spesifik seperti mengurus autentikasi, manajemen pengguna, manajemen produk, dll. Modul dapat berisi beberapa komponen seperti controller, service, dan provider.

NestJS memiliki dokumentasi yang lengkap dan mudah dipahami, sehingga memudahkan pengembang untuk memulai mengembangkan aplikasi dengan cepat. Selain itu, framework ini juga menyediakan fitur-fitur seperti CLI (Command Line Interface) untuk mempercepat proses pengembangan dan pengujian aplikasi.

2.2.4. PostgreSQL

PostgreSQL adalah sistem manajemen basis data relasional open-source yang bersifat objek-relasional (Juba et al., 2015). Dalam PostgreSQL, data diatur dalam tabel, yang terdiri dari baris dan kolom. Basis data relasional seperti PostgreSQL adalah cara yang umum digunakan untuk menyimpan dan mengelola data dalam aplikasi. PostgreSQL mendukung berbagai fitur seperti transaksi, indeks, fungsi, tampilan, dan trigger.

PostgreSQL merupakan salah satu basis data open-source paling populer dan kuat di dunia. Ia terkenal karena keandalannya, keamanannya, dan kemampuan untuk menangani beban kerja yang besar. PostgreSQL juga mendukung bahasa pemrograman seperti C, C++, Java, Perl, Python, Ruby, dan lainnya, sehingga memudahkan pengembang dalam membangun aplikasi yang kompleks.

PostgreSQL dapat dijalankan di berbagai sistem operasi seperti Windows, Linux, macOS, dan BSD. Selain itu, PostgreSQL menyediakan antarmuka klien-server yang memungkinkan pengguna untuk mengakses basis data melalui jaringan, dan juga mendukung protokol komunikasi standar seperti TCP/IP. PostgreSQL juga mendukung standar SQL, sehingga memudahkan pengembang dalam membuat dan mengelola basis data relasional.

2.2.4.1. tsvector

Dalam PostgreSQL, tipe data khusus yang disebut "tsvector" digunakan untuk menyimpan dokumen teks yang sudah diproses secara penuh atau "tokenized". "tsvector" merupakan

singkatan dari "text search vector" yang dapat digunakan untuk meningkatkan kemampuan pencarian teks pada kolom teks dalam tabel.

"tsvector" pada dasarnya adalah representasi vektor yang mengandung daftar kata-kata unik yang muncul dalam teks bersama dengan posisi kata tersebut dalam dokumen. Tipe data ini memungkinkan pengindeksan dan pencarian teks yang lebih efisien dalam PostgreSQL.

Proses penciptaan "tsvector" melibatkan dua tahap utama: tokenisasi dan normalisasi. Pertama, teks yang dimasukkan akan dipecah menjadi kata-kata individu, mengabaikan karakter non-alphanumeric seperti tanda baca dan spasi. Selanjutnya, kata-kata tersebut akan dinormalisasi, yang meliputi penghapusan akhiran kata, pengubahan huruf besar menjadi huruf kecil, dan penghapusan kata-kata umum (stop words) yang tidak memberikan informasi penting dalam konteks pencarian.

Setelah tahap pemrosesan selesai, hasil akhir adalah "tsvector" yang terdiri dari daftar kata-kata unik dan posisi kata-kata tersebut dalam dokumen. Dengan menggunakan indeks yang sesuai pada kolom "tsvector", pencarian teks dapat dilakukan lebih efisien, karena tidak perlu memindai setiap baris dalam tabel.

Misalnya, dengan menggunakan indeks pada kolom "tsvector" yang berisi data teks dari dokumen, kita dapat dengan mudah mencari kata-kata tertentu dalam dokumen, menghitung frekuensi kata-kata, atau bahkan melakukan pencarian teks berdasarkan relevansi dengan menggunakan fungsi-fungsi pencarian teks bawaan PostgreSQL seperti "ts_query".

Dalam rangka memanfaatkan fitur pencarian teks yang kuat ini, PostgreSQL menyediakan sejumlah operator dan fungsi yang dapat digunakan untuk memanipulasi dan mencari data teks dengan menggunakan tipe data "tsvector" dan "tsquery" yang saling berinteraksi.

2.2.4.2. PostgresML

Karena kebutuhan system resource dari sebuah sistem rekomendasi, dibutuhkan platform yang cukup performan untuk memungkinkan. PostgresML adalah sebuah aplikasi dan database vektor, simpanan model, simpanan fitur, server inferensi, dan jalur implementasi ML (Machine Learning) yang merupakan perluasan dari PostgreSQL. PostgresML didesain untuk mendukung semua framework machine learning utama dan memungkinkan pengguna melakukan pelatihan dan inferensi pada data teks dan data tabel menggunakan kueri SQL.

Sebagai solusi machine learning end-to-end, PostgresML terintegrasi dengan sempurna dengan PostgreSQL, memungkinkan pengembang untuk dengan cepat membuat prototipe dan

menerapkan aplikasi AI. PostgresML menawarkan berbagai kemampuan seperti membuat dan mengindeks embedding menggunakan model open source dari Huggingface, atau melatih model custom.

Salah satu keunggulan utama dari PostgresML adalah skalabilitasnya. Dengan memanfaatkan replika yang didedikasikan, PostgresML dapat dengan cepat mengatasi beban kerja yang semakin besar. Hal ini memastikan bahwa tugas machine learning dapat dilaksanakan dengan efisien bahkan dengan dataset yang besar dan model yang kompleks.

PostgresML adalah sebuah perluasan open-source untuk PostgreSQL, yang berarti mendapat manfaat dari komunitas pengembang yang aktif dan peningkatan yang terus-menerus. Ini memberikan para pengembang dengan seperangkat alat dan fitur yang komprehensif untuk menyederhanakan alur kerja machine learning dalam lingkungan yang sudah dikenal dari PostgreSQL.

2.2.4.3. pg_trgm

Pg_trgm (PostgreSQL Trigram) adalah ekstensi yang disediakan oleh PostgreSQL untuk melakukan pencocokan string berdasarkan trigram. Trigram adalah tiga karakter berurutan dalam sebuah string. Ekstensi ini memungkinkan Anda untuk melakukan pencarian atau pemadanan string dengan menggunakan metode trigram similarity.

Pg_trgm menyediakan fungsi dan operator untuk menghitung kesamaan trigram antara dua string. Fungsi-fungsi ini memungkinkan Anda untuk mengukur sejauh mana dua string mirip atau seberapa banyak trigram yang sama di antara keduanya.

Dalam penggunaannya, Anda dapat membuat indeks pada kolom yang berisi string dengan menggunakan ekstensi pg_trgm. Indeks trigram ini memungkinkan pencocokan cepat dan efisien saat melakukan pencarian string dengan menggunakan operator dan fungsi pg_trgm.

Contoh penggunaan pg_trgm adalah ketika Anda ingin mencari string yang mirip atau mendekati string yang sudah ada dalam database. Dengan menggunakan pg_trgm, Anda dapat menghitung tingkat kesamaan antara string yang dicari dengan string yang ada dalam database dan mengembalikan hasil yang paling relevan.

Selain itu, pg_trgm juga dapat digunakan dalam operasi lain seperti penggabungan, pemisahan, dan perbandingan string berdasarkan trigram similarity. Hal ini memungkinkan Anda untuk melakukan berbagai operasi pemadanan dan analisis string dengan mudah menggunakan ekstensi ini.

Pg_trgm adalah salah satu dari banyak ekstensi yang tersedia dalam PostgreSQL yang memperluas fungsionalitas sistem manajemen basis data tersebut. Dengan menggunakan pg_trgm, Anda dapat meningkatkan kemampuan pencarian dan pemadanan string dalam aplikasi yang menggunakan PostgreSQL.

2.2.5. Redis

Redis adalah sebuah sistem penyimpanan data open-source yang menggunakan struktur data berbasis key-value (Da Silva & Tavares, 2015). Redis dapat digunakan sebagai database, cache, dan message broker. Redis memiliki kecepatan yang sangat tinggi dan dapat menangani banyak jenis data, termasuk string, hash, list, set, dan sorted set.

Redis juga dapat digunakan untuk memecahkan masalah dalam aplikasi yang kompleks, seperti mempercepat waktu akses ke data yang sering digunakan, mengurangi beban database utama dengan menyimpan data di dalam cache, dan menyimpan pesan antara komponen aplikasi yang berbeda.

Redis dapat diintegrasikan dengan berbagai bahasa pemrograman dan framework, seperti Node.js, Python, Ruby, PHP, dan Java. Redis juga memiliki dukungan untuk replikasi data dan sharding, sehingga memungkinkan untuk membuat sistem yang lebih skala horizontal dengan mudah.

2.2.6. MySQL

MySQL adalah sistem manajemen basis data relasional (RDBMS) yang sangat populer dan banyak digunakan di seluruh dunia. MySQL dikembangkan oleh perusahaan Oracle Corporation dan merupakan salah satu RDBMS yang paling banyak digunakan dalam pengembangan aplikasi web, terutama untuk mengelola dan menyimpan data dalam basis data.

MySQL didasarkan pada bahasa pemrograman SQL (Structured Query Language) yang digunakan untuk mengelola basis data. SQL memungkinkan pengguna untuk membuat, mengubah, menghapus, dan mengambil data dari basis data. MySQL menyediakan berbagai fitur dan fungsi yang kuat untuk mengelola basis data, termasuk dukungan untuk transaksi, indeks, tampilan, trigger, prosedur tersimpan, dan banyak lagi.

MySQL dapat diintegrasikan dengan berbagai bahasa pemrograman dan framework pengembangan aplikasi, seperti PHP, Java, Python, dan Ruby. Hal ini membuat MySQL menjadi pilihan yang populer bagi pengembang aplikasi web untuk menyimpan dan mengelola data.

Keuntungan menggunakan MySQL antara lain:

1. Kinerja yang tinggi: MySQL dirancang untuk memberikan kinerja yang cepat dan efisien dalam pemrosesan data, dengan kemampuan untuk mengelola basis data yang besar.
2. Skalabilitas: MySQL mendukung replikasi dan partisi data, memungkinkan pengguna untuk meningkatkan kapasitas dan kinerja basis data sesuai dengan kebutuhan aplikasi.
3. Keamanan: MySQL menyediakan berbagai fitur keamanan, termasuk otentikasi pengguna, akses kontrol, enkripsi data, dan audit log.
4. Mudah digunakan: MySQL memiliki antarmuka yang sederhana dan intuitif, sehingga mudah dipelajari dan digunakan oleh pengembang pemula.
5. Komunitas dan dokumentasi yang luas: MySQL memiliki komunitas pengguna yang besar dan aktif, serta dokumentasi yang lengkap dan mendetail, sehingga memudahkan pengguna dalam mempelajari dan menyelesaikan masalah.

MySQL memiliki beberapa edisi, termasuk edisi komunitas (MySQL Community Edition) yang dapat digunakan secara gratis, dan edisi berbayar (MySQL Enterprise Edition) yang menyediakan fitur dan dukungan tambahan untuk kebutuhan bisnis yang lebih canggih.

Dalam kesimpulannya, MySQL adalah sistem manajemen basis data relasional yang populer dan kuat yang digunakan secara luas dalam pengembangan aplikasi web. Keunggulannya termasuk kinerja tinggi, skalabilitas, keamanan, dan mudah digunakan. MySQL menjadi pilihan yang baik bagi pengembang untuk mengelola dan menyimpan data dalam aplikasi mereka.

2.2.7. NextJS

Next.js adalah sebuah kerangka kerja (framework) JavaScript yang digunakan untuk mengembangkan aplikasi web modern. Ini adalah pengembangan lebih lanjut dari React.js yang memberikan fitur-fitur tambahan dan kemudahan dalam pembuatan aplikasi web.

Berikut adalah beberapa poin penting tentang Next.js:

1. Server-side Rendering (Rendering di Sisi Server): Salah satu fitur utama dari Next.js adalah kemampuannya untuk melakukan server-side rendering (SSR). SSR memungkinkan pembuatan tampilan awal aplikasi dilakukan di sisi server, sehingga pengguna dapat melihat konten yang dirender dengan cepat sejak awal. Hal ini

membantu meningkatkan kecepatan dan kinerja aplikasi, serta membantu dalam hal SEO (Search Engine Optimization).

2. Static Site Generation (Generasi Situs Statis): Next.js juga mendukung static site generation (SSG), yang memungkinkan pembuatan situs web statis dengan menghasilkan halaman HTML statis pada waktu pembuatan (build time). Ini sangat berguna untuk situs web yang kontennya tidak berubah secara teratur, karena halaman HTML yang dihasilkan dapat di-cache dan disajikan secara cepat kepada pengguna.
3. Routing Dinamis: Next.js menyediakan routing yang mudah dan fleksibel. Anda dapat membuat route yang dinamis dengan menggunakan tanda kurung kurawal ({}) dalam definisi path, yang kemudian dapat digunakan untuk menentukan parameter dinamis dalam URL.
4. Hot Module Replacement (HMR): Next.js mendukung Hot Module Replacement, yang memungkinkan perubahan kode yang dilakukan pada waktu pengembangan dapat dilihat secara langsung tanpa perlu melakukan refresh pada browser. Hal ini mempercepat proses pengembangan dan memudahkan pembuatan perubahan dalam waktu nyata.
5. Ekosistem React: Next.js dibangun di atas React.js, yang merupakan salah satu library JavaScript yang paling populer untuk pengembangan antarmuka pengguna. Dengan menggunakan Next.js, Anda dapat memanfaatkan semua kelebihan React.js, termasuk komponen yang dapat digunakan kembali (reusable components), manajemen state yang efisien, dan ekosistem yang luas.
6. Pengaturan Konfigurasi yang Mudah: Next.js menyediakan file konfigurasi sederhana yang memungkinkan Anda untuk mengatur berbagai opsi pengembangan, seperti mengatur direktori sumber, mengaktifkan modus pengembangan atau produksi, dan menambahkan pengaturan kustom sesuai kebutuhan proyek.

Next.js telah menjadi pilihan populer bagi pengembang untuk membangun aplikasi web modern. Dengan kombinasi SSR, SSG, routing yang fleksibel, dan dukungan penuh untuk React.js, Next.js memudahkan pengembangan aplikasi web yang cepat, tanggap, dan mudah di-maintain.

BAB III

ANALISIS DAN PERANCANGAN

3.1. Analisis

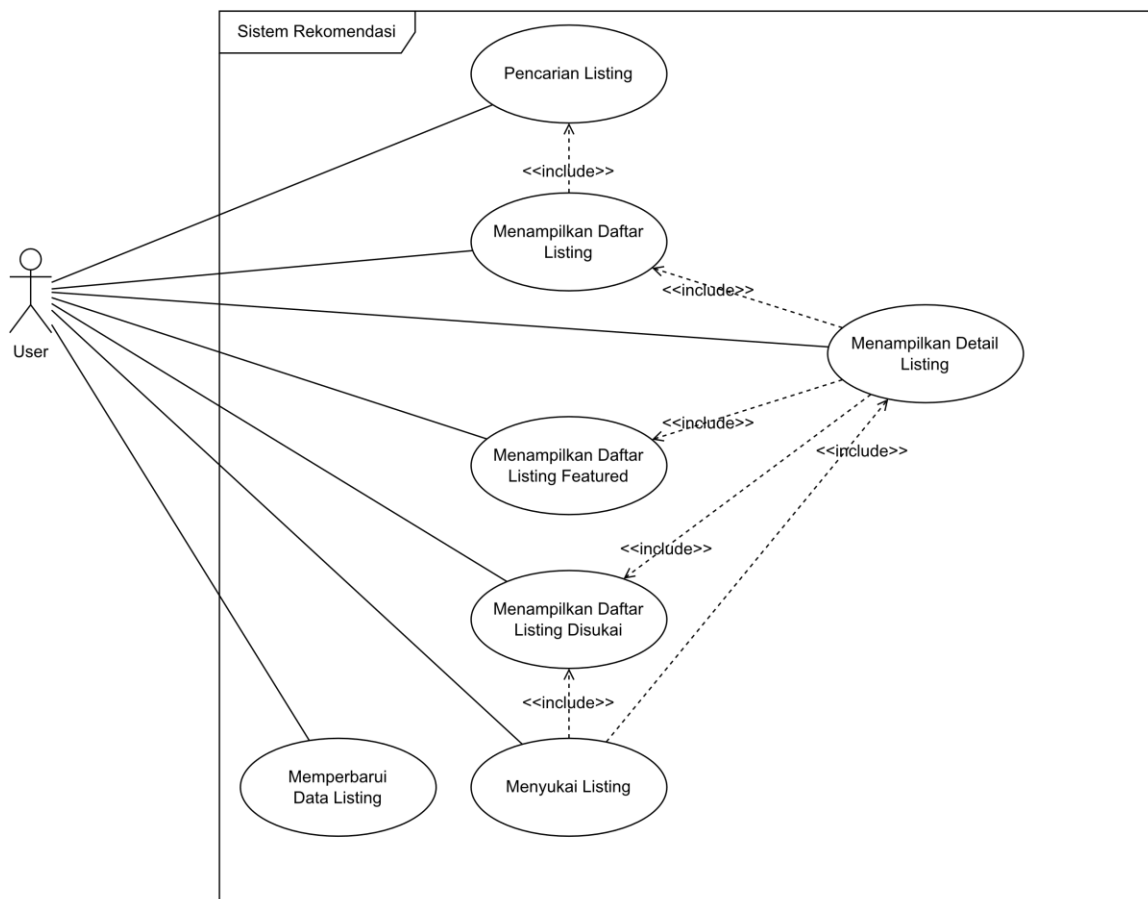
3.1.1. Use Case Diagram

Berikut adalah use case untuk sistem rekomendasi website perusahaan real estate:

1. Pencarian Properti
 - a. User melakukan pencarian properti berdasarkan kriteria tertentu seperti lokasi, harga, tipe properti, luas tanah, dan sebagainya.
 - b. Sistem Rekomendasi menampilkan daftar properti yang sesuai dengan kriteria yang dimasukkan oleh pengguna.
2. Menampilkan Daftar Listing
 - a. Setelah melakukan pencarian, user dapat melihat daftar properti yang tersedia.
 - b. Sistem Rekomendasi menampilkan informasi dasar tentang setiap properti seperti foto, lokasi, harga, ukuran, dan deskripsi.
3. Menampilkan Detail Listing
 - a. Setelah melihat daftar properti, user dapat memilih salah satu properti untuk melihat informasi detail.
 - b. Menampilkan informasi detail tentang properti seperti fasilitas, kondisi bangunan, riwayat perawatan, dan sebagainya.
 - c. User dapat melihat daftar listing yang direkomendasikan oleh sistem rekomendasi berdasarkan listing yang sedang dilihat.
 - d. Sistem Rekomendasi menampilkan pada halaman detail listing sebuah daftar listing yang dipilih berdasarkan kriteria tertentu seperti relevansi terhadap pencarian, listing yang detailnya sedang dilihat, harga, atau kualitas properti.
4. Menyukai Listing
 - a. User dapat menandai listing properti yang disukai atau dianggap menarik dengan tombol "like" atau "disukai".
 - b. Sistem Rekomendasi menyimpan informasi tentang listing properti yang telah disukai oleh pengguna.
5. Menampilkan Daftar Listing Disukai
 - a. User dapat melihat daftar listing properti yang telah disukai atau dianggap menarik.

- b. Sistem Rekomendasi menampilkan daftar listing properti yang telah disukai oleh pengguna.
6. Menampilkan Daftar Listing Featured
 - a. Sistem Rekomendasi menampilkan daftar listing featured pada halaman utama website real estate.
 - b. Listing featured dipilih berdasarkan kriteria tertentu seperti popularitas, lokasi, atau tipe properti.

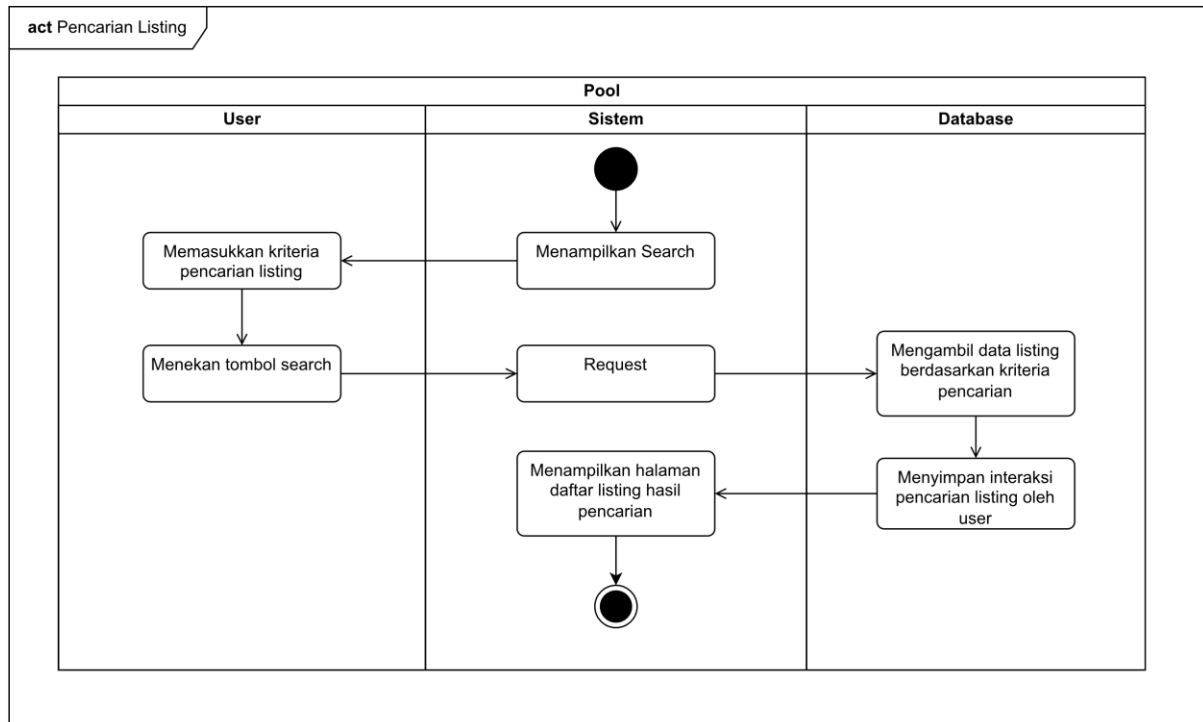
Sistem Rekomendasi dapat menggunakan data dari daftar pencarian, kunjungan halaman detail listing, listing disukai, dan interaksi lain untuk memberikan rekomendasi yang lebih tepat dan relevan bagi pengguna. Secara keseluruhan, use case sistem rekomendasi ditampilkan sebagai diagram pada Gambar 3.1.



Gambar 3.1 Diagram Use Case Sistem Rekomendasi

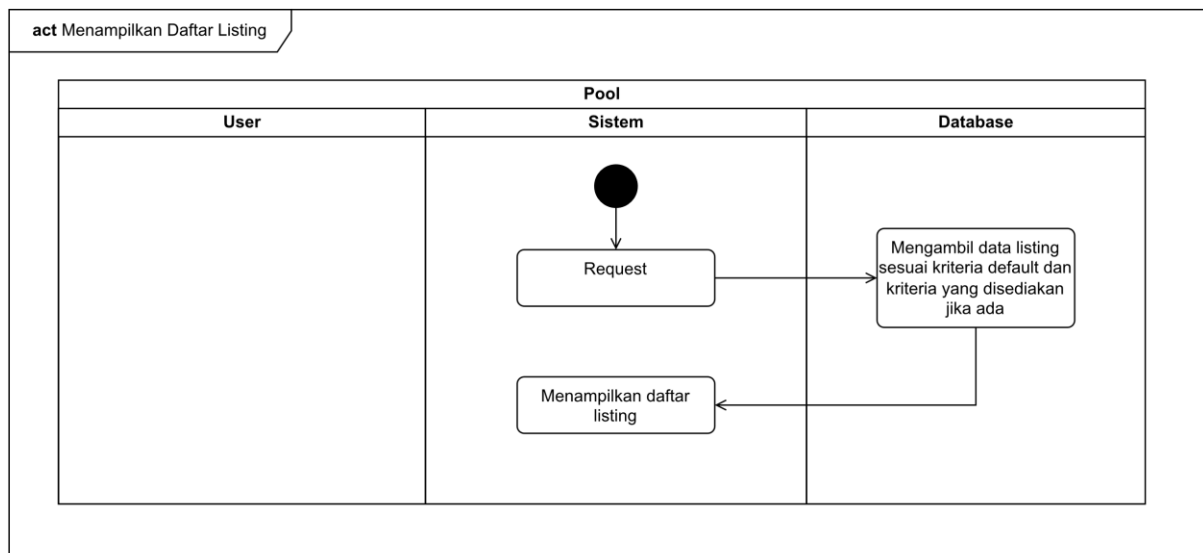
3.1.2. Activity Diagram

Berdasarkan use case diagram pada Gambar 3.1, berikut ini adalah activity diagram untuk tiap use case:



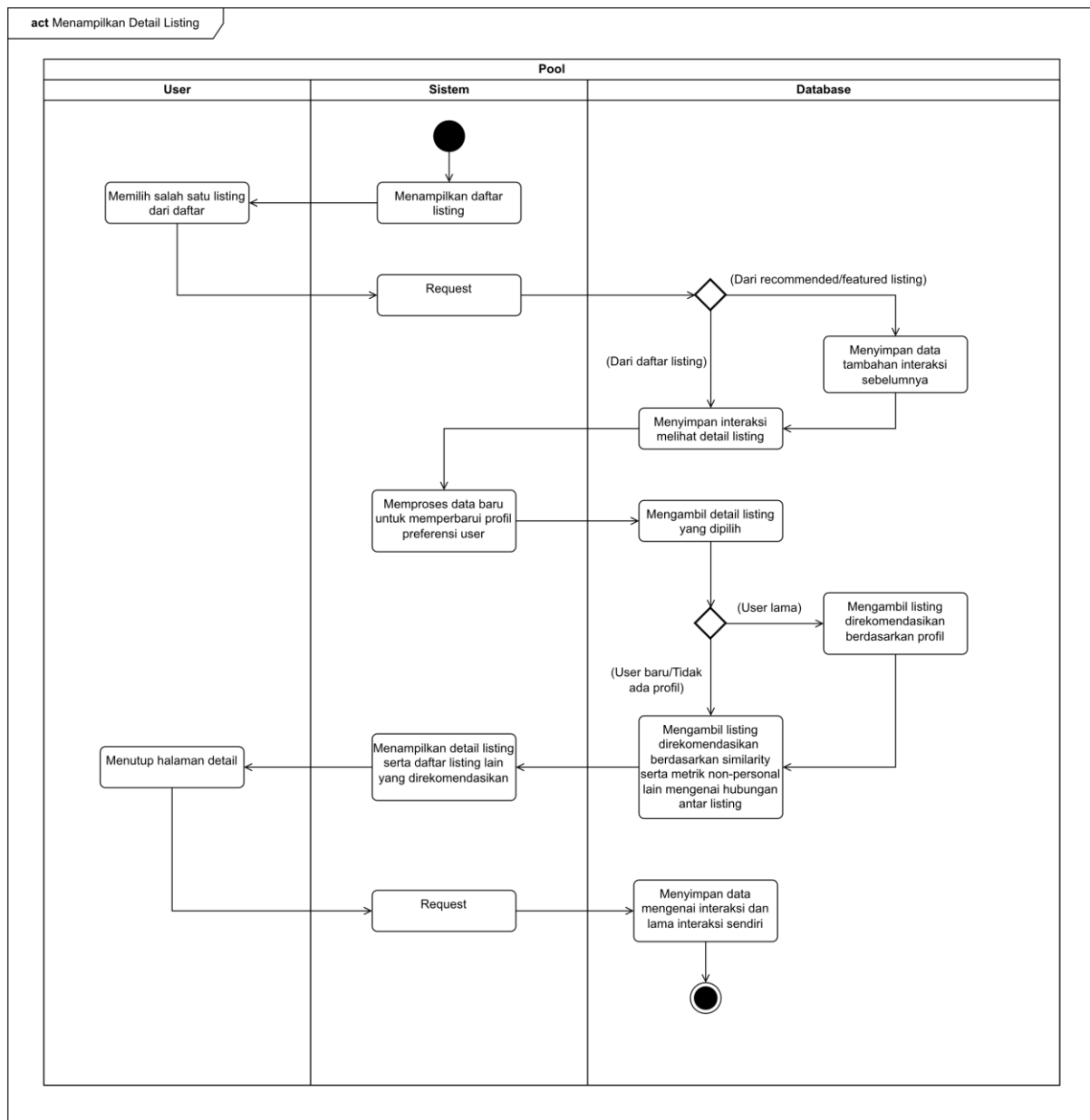
Gambar 3.2 Activity Diagram Pencarian Listing

Salah satu kebutuhan utama dari sistem rekomendasi adalah untuk mengumpulkan data mengenai perilaku pengguna dan hubungannya dengan listing. Ini dapat dilakukan dengan beberapa cara. Cara yang pertama digunakan pada rancangan ini adalah saat user menggunakan fitur search di website. Kriteria pencarian yang digunakan dan hasil dan interaksi dengan hasil dari pencarian dapat digunakan untuk menambah pemahaman sistem rekomendasi maka dari itu kedua interaksi tersebut disimpan sebagai mana ditunjukkan pada Gambar 3.2. Hasil dari pencarian sendiri memiliki alur seperti yang ditunjukkan dalam Gambar 3.3.



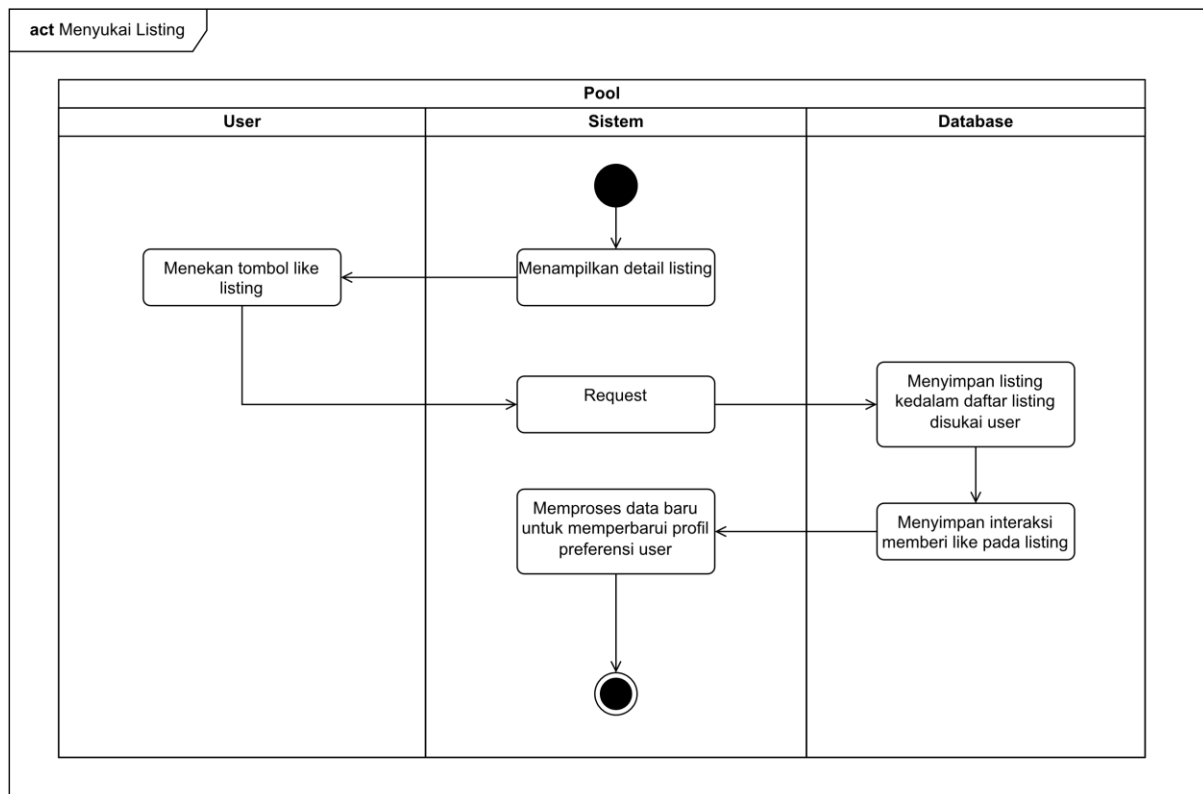
Gambar 3.3 Activity Diagram Menampilkan Daftar Listing

Dari daftar listing, baik itu daftar hasil pencarian, daftar keseluruhan, ataupun daftar rekomendasi, tiap listing dapat dilihat detailnya. Interaksi memilih dan membuka detail sebuah listing secara implisit berarti user memberi penilaian lebih pada listing yang telah dibuka detailnya. Ini merupakan sumber informasi untuk sistem rekomendasi. Untuk memberi akurasi lebih tinggi, penilaian oleh user tadi diberi bobot menggunakan beberapa hal lain, termasuk lama interaksi di halaman detail dan jumlah tiap jenis interaksi yang terjadi pada halaman detail. Selain itu, karena halaman detail juga memuat rekomendasi untuk properti yang serupa maka interaksi lanjut dengan listing yang direkomendasikan tersebut dapat digunakan untuk menilai seberapa berguna rekomendasi tersebut. Ini digunakan untuk memperbaiki sistem rekomendasi. Kedua penyimpanan data interaksi pada detail listing dan penggunaan data interaksi rekomendasi listing alurnya digambarkan pada Gambar 3.4.

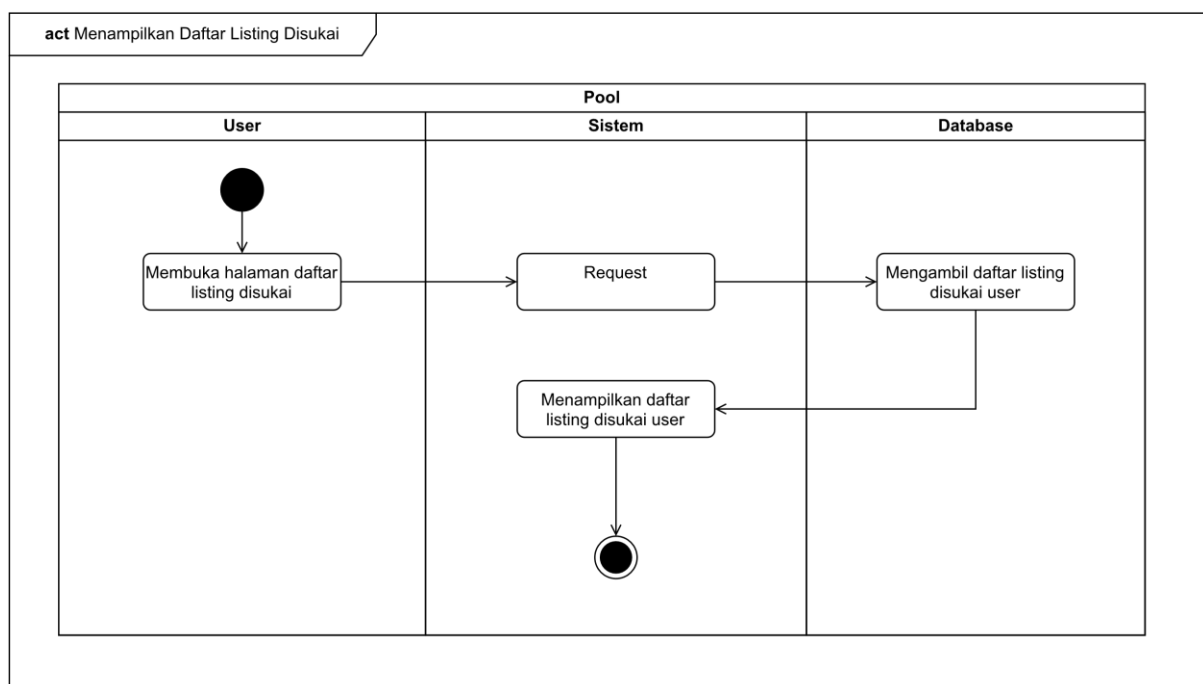


Gambar 3.4 Activity Diagram Menampilkan Detail Listing

Selain interaksi dan penilaian implisit, sistem rekomendasi juga akan lebih baik jika dapat menggunakan penilaian eksplisit. Dalam kasus ini, karena sifat listing yang tidak memungkinkan mendapatkan banyak penilaian untuk sebuah listing spesifik, sistem like yang bertindak seperti bookmark memungkinkan sistem rekomendasi untuk mendapatkan penilaian tersebut tanpa properti dibeli. Sistem “like” ini digambarkan pada Gambar 3.5 dan Gambar 3.6.

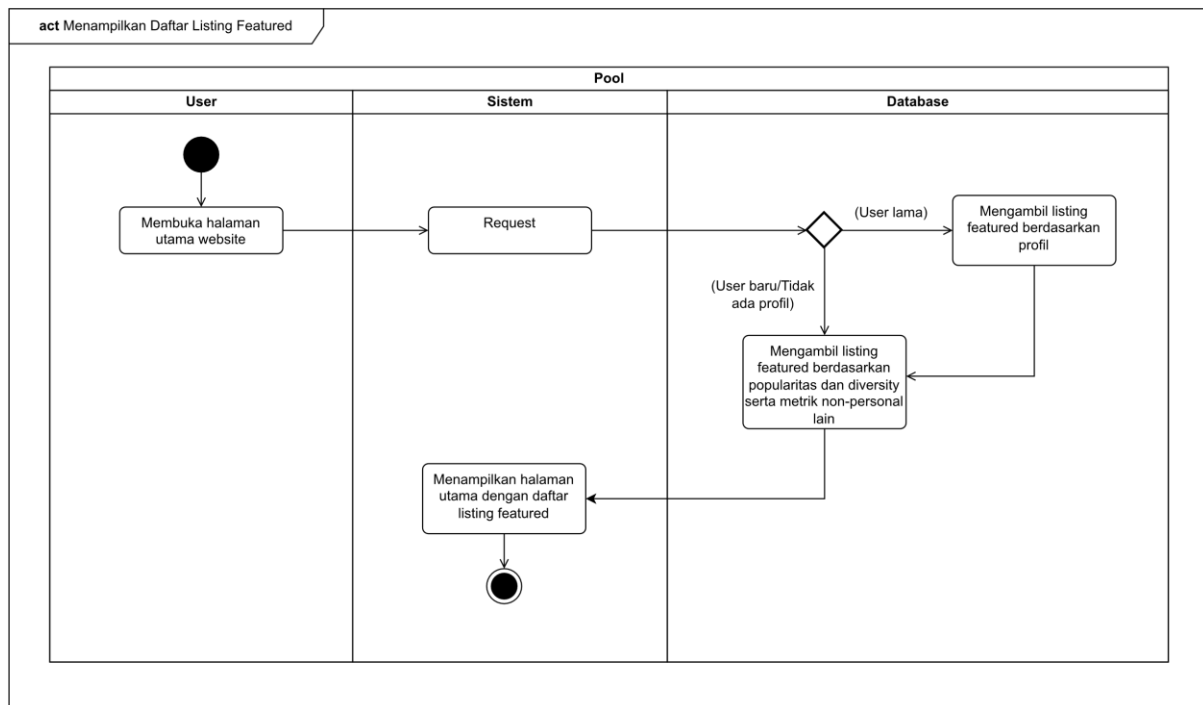


Gambar 3.5 Activity Diagram Menyukai Listing



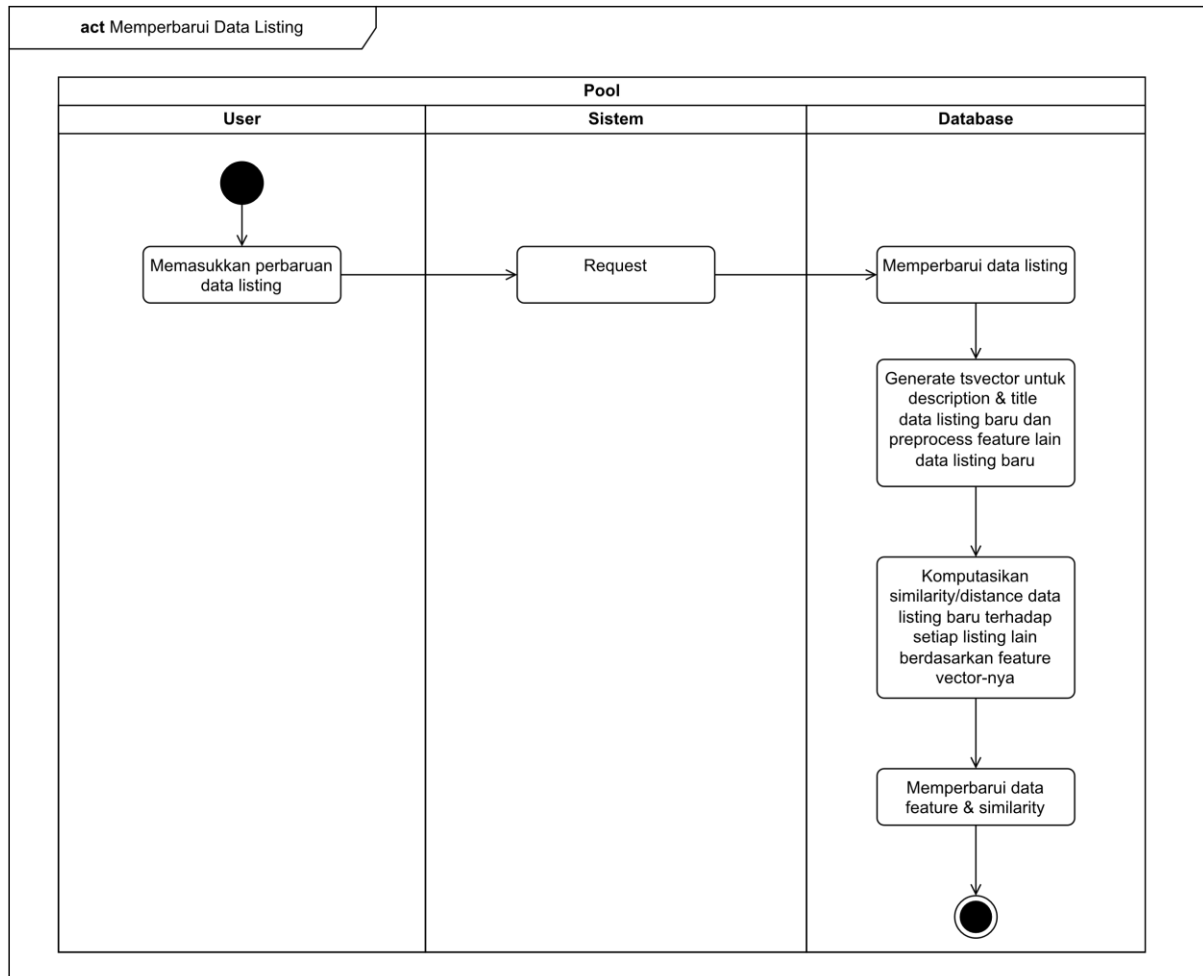
Gambar 3.6 Activity Diagram Menampilkan Daftar Listing Disukai

Selain rekomendasi berdasarkan listing detail yang sedang di lihat, terdapat juga rekomendasi listing lebih umum yang hanya berdasarkan data interaksi user dengan listing dan listing populer sebagaimana digambarkan pada Gambar 3.7.



Gambar 3.7 Activity Diagram Menampilkan Daftar Listing Featured

Untuk sistem rekomendasi hybrid, selain rekomendasi dari collaborative filtering, juga dibutuhkan nilai rekomendasi dari content-based recommendation. Nilai ini di komputasi pada saat data listing telah diubah. Ini berarti setelah listing baru dimasukkan, atau data listing lama di update atau di hapus. Proses ini dapat dilihat pada Gambar 3.8.



Gambar 3.8 Activity Diagram Memperbarui Data Listing

3.1.3. Spesifikasi Data

Terdapat tiga tipe data sumber informasi yang digunakan untuk sistem rekomendasi yang dimaksud dalam laporan ini, yaitu:

1. Listing

Merupakan data mengenai iklan properti dan properti itu sendiri. Antara lain, atribut listing termasuk lokasi, karakteristik properti, deskripsi, title, dan identifikasi unik dari listing.

2. Visitor Preference Profile

Merupakan data yang dibuat saat user mengunjungi website yang memiliki atribut untuk mengidentifikasi user unik. Data ini adalah agregasi dari semua interaksi user dengan halaman detail listing. Interaksi tersebut adalah sebagai berikut:

- a. ViewDetail: 1
- b. ViewPictures: 2
- c. PlayVideo: 3
- d. OpenMoreDetails: 3
- e. KPR: 5
- f. SavePDF: 5
- g. Share: 7
- h. Whatsapp: 8
- i. Like: 8

3. Event

Merupakan data mengenai interaksi antara user dan listing yang terdiri atas atribut mendeskripsikan waktu, jenis interaksi, dan informasi lebih mengenai interaksi tersebut seperti halaman sebelumnya yang juga telah direkomendasikan oleh sistem sebagaimana pada use case menampilkan detail listing di Gambar 3.4.

3.1.4. Menghindari Cold-Start Problem

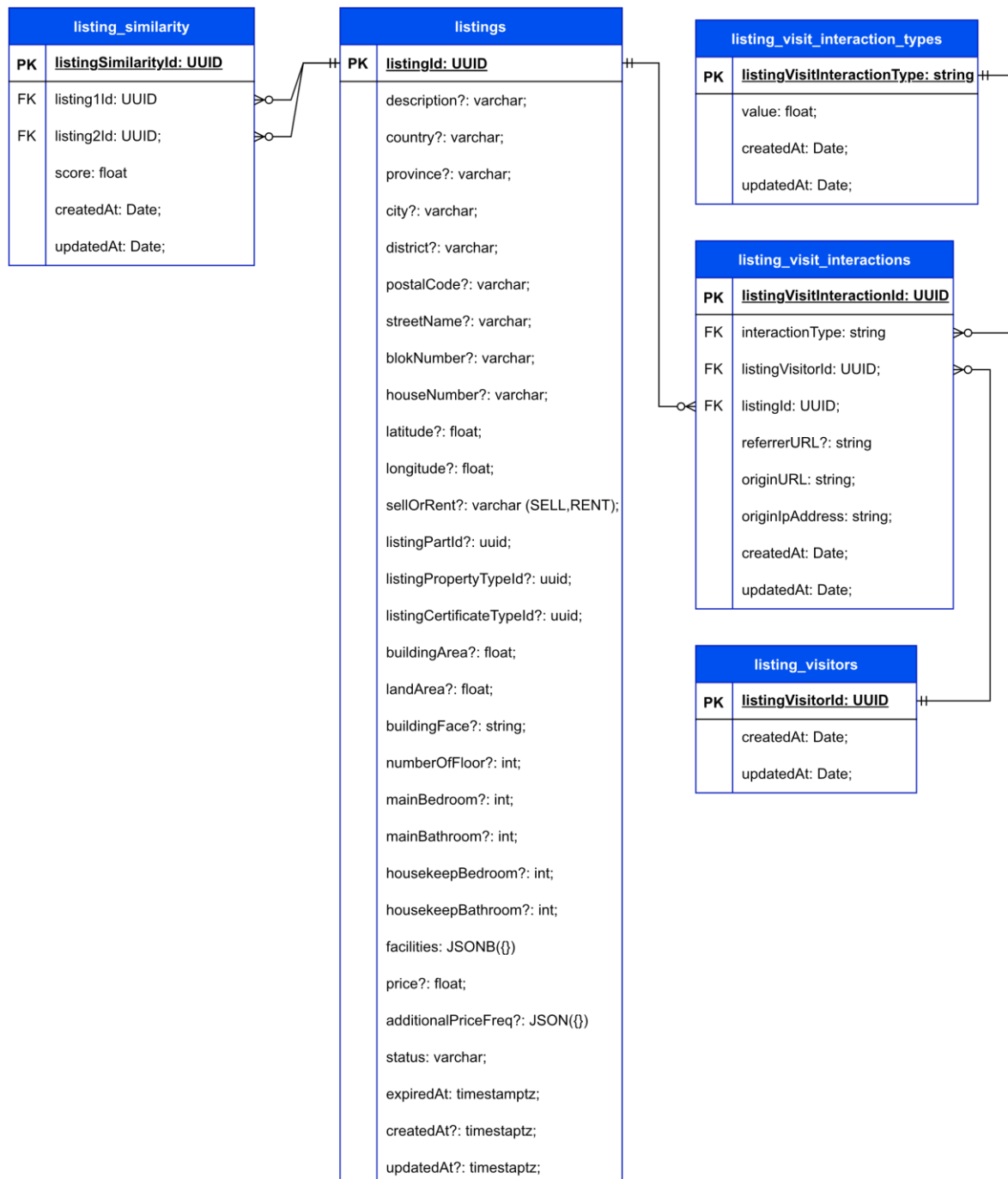
Karena website perusahaan klien sebelumnya tidak menggunakan sistem rekomendasi, maka permasalahan cold-start akan membatasi secara besar kemampuan sistem rekomendasi collaborative filtering untuk berfungsi. Namun, manfaat dari collaborative filtering, ternama menggunakan fitur yang tidak terdapat pada data seperti preferensi sosial membuatnya dapat memberikan rekomendasi baru yang dapat memikat pengguna setelah cukup banyak pengguna memberikan feedback. Oleh karena itu, untuk mendapatkan manfaat kedua jenis sistem, pendekatan hybrid akan digunakan dalam sistem rekomendasi untuk website perusahaan real estate (Burke, 2002).

3.2. Perancangan

3.2.1. Entity Relationship Diagram

Data yang digunakan untuk sistem rekomendasi adalah data listing dan data interaksi pengguna serta identifikasi pengguna itu sendiri. Data listing terdiri atas informasi relevan mengenai sebuah listing seperti lokasi, jenis properti, jenis sertifikat legal yang berhubungan, spesifikasi properti dalam listing, dan deskripsi teks listing. Data listing ini disimpan dalam table listings. Setelah detail listing diproses untuk kedekatan antara listing hasil kedekatannya disimpan di

dalam table listing_similarity. Selain ini, juga terdapat data yang digunakan untuk mencatat aktivitas pengunjung. Table listing_visitors digunakan untuk menyimpan identitas unik pengunjung. Table listing_visit_interactions menyimpan data interaksi seorang pengunjung dengan listing. Jenis interaksi disimpan dalam table list_visit_interaction_types. Entity Relationship Diagram (ERD) untuk sistem rekomendasi ditampilkan pada Gambar 3.9.



Gambar 3.9. Entity Relationship Diagram Sistem Rekomendasi

3.2.2. Desain Sistem Rekomendasi

3.2.2.1. Rekomendasi Content-based

Rekomendasi content-based, seperti namanya, menggunakan content atau isi listing untuk memberikan rekomendasi listing. Yang pertama adalah membuat fitur yang dapat digunakan untuk pengukuran similarity antar listing. Untuk data lokasi, dibuat fitur yang dihitung sebagai kesamaan lokasi berdasarkan kategori dalam hierarki (country, province, city, district). Sebagai contoh, listing yang berada dalam kota yang sama akan memiliki nilai kedekatan yang lebih tinggi dibandingkan dengan listing yang hanya sama provinsi.

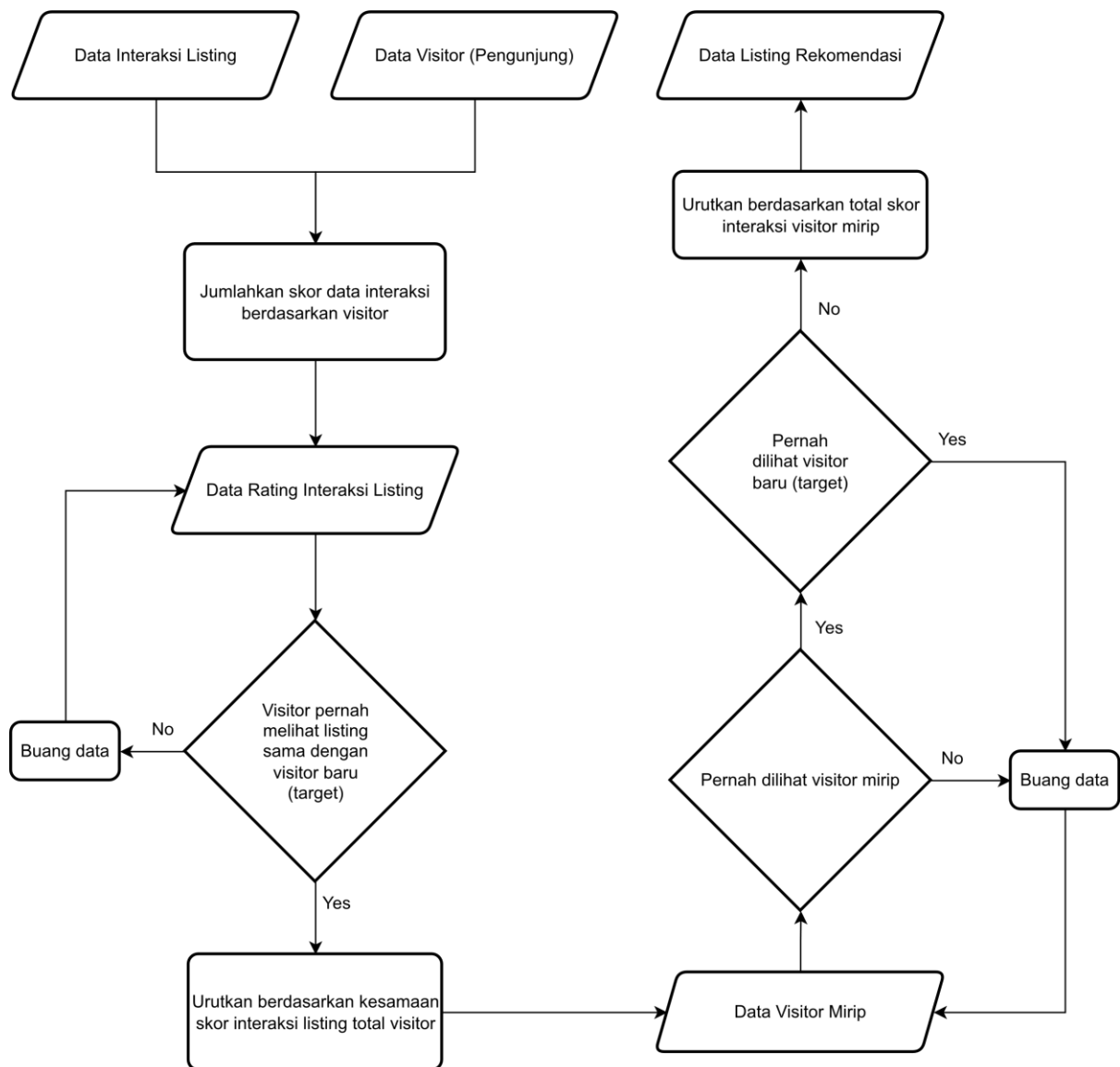
```
score_location = (listing1.country = listing2. country)::Int +2*(listing1.province = listing2.  
province)::Int + 3*(listing1.city = listing2. city)::Int + 4*(listing1.district =  
listing2.district)::Int
```

Data text seperti description dan judul, digunakan similarity dari ts_rank menggunakan tsvector yang dihasilkan dari data text sebelumnya. Untuk data lain score similarity didapatkan dari cosine similarity sederhana dari vector yang dibuat dari data di atas.

Kemudian, semua score similarity dibobot dan dijumlahkan untuk mendapatkan score similarity keseluruhan.

3.2.2.2. Rekomendasi Collaborative

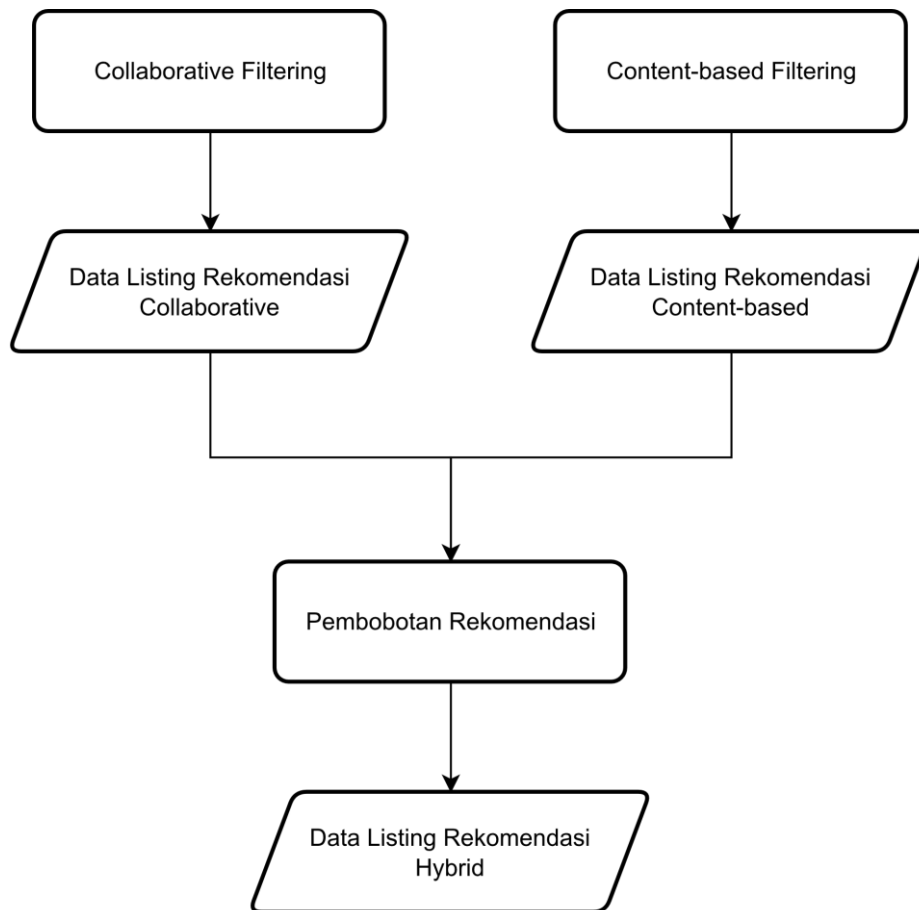
Rekomendasi collaborative dihasilkan menggunakan interaksi pengunjung dan pengunjung lain dengan listing untuk memberikan rekomendasi listing. Setelah seorang pengunjung berinteraksi dengan beberapa listing, interaksi tersebut digunakan untuk mencari pengunjung lain yang memiliki skor interaksi yang mirip dengan pengunjung baru. Kemudian, listing yang pernah dilihat pengunjung lain tapi belum pernah dilihat pengunjung baru akan direkomendasikan dan diurutkan berdasarkan total skor interaksi pengunjung lain masing-masing listing. Proses ini diilustrasikan pada Gambar 3.10.



Gambar 3.10. Flowchart Collaborative Filtering Recommendation

3.2.2.3. Rekomendasi Hybrid

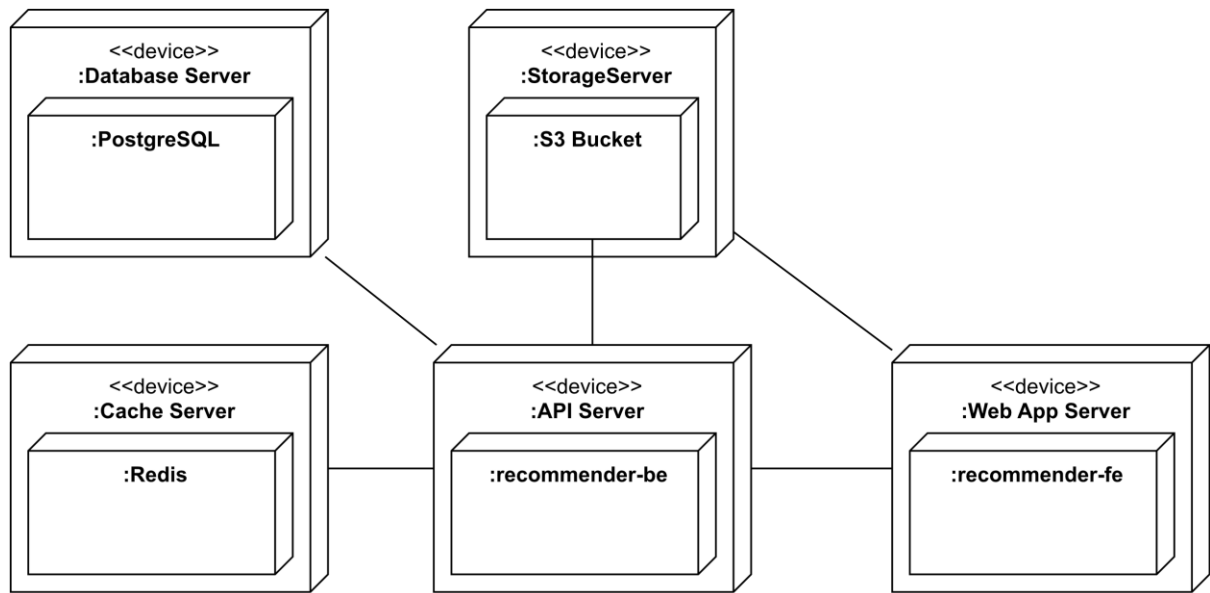
Untuk kebutuhan rekomendasi yang mirip dengan suatu listing referensi namun perlu tetap terpersonalisasi, digunakan rekomendasi hybrid. Ini dilakukan dengan menjumlahkan skor rekomendasi collaborative dengan score similarity dengan bobot. Dalam kasus sebuah listing tidak direkomendasikan oleh rekomendasi collaborative, maka nilainya hanya didapatkan dari nilai rekomendasi content-based. Alur pembobotan ini dapat dilihat di Gambar 3.11.



Gambar 3.11. Hybrid recommendation flowchart

3.2.3. Deployment Diagram

Berikut ini adalah deployment untuk sistem rekomendasi yang telah di rancang. Aplikasi ini menggunakan 5 device untuk semua server yang dibutuhkannya.



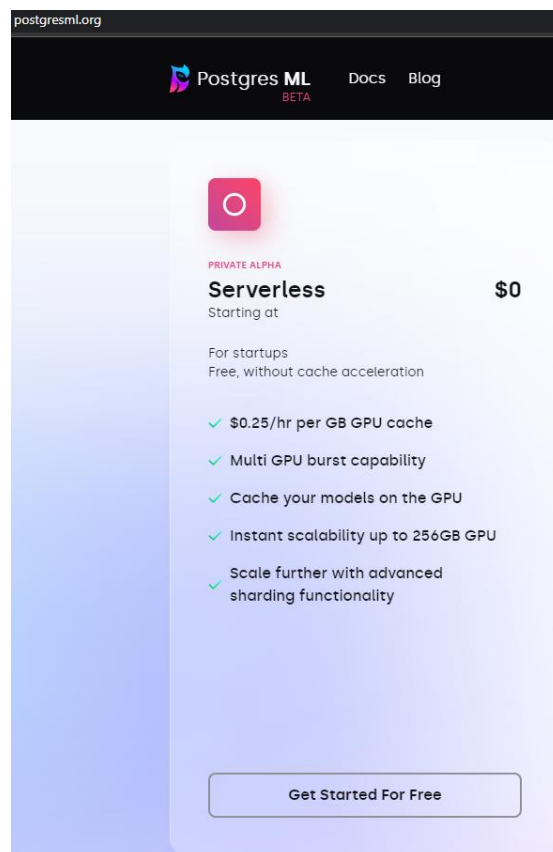
Gambar 3.12 Deployment Diagram Aplikasi Web Dengan Sistem Rekomendasi

BAB IV

HASIL KEGIATAN PLA

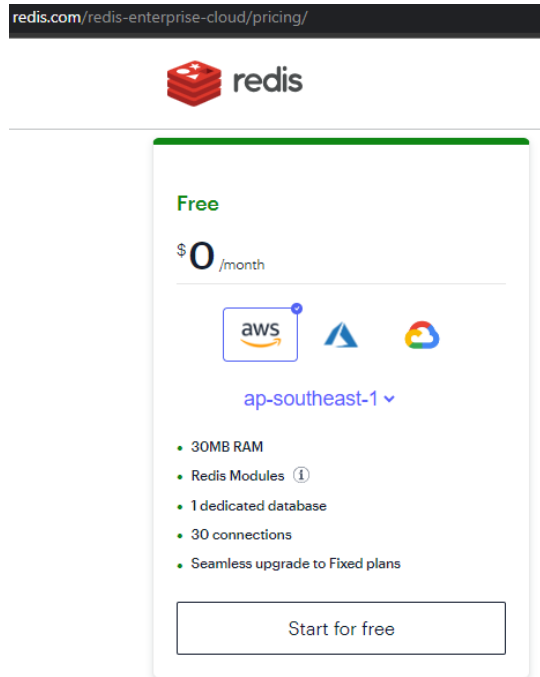
4.1. Lingkungan Implementasi

Implementasi dari sistem rekomendasi untuk website real estate dideploy menggunakan beberapa layanan dan solusi hosting. Karena sistem ini belum masuk ke production full untuk digunakan, maka hosting untuk sistem rekomendasi ini menggunakan layanan free beberapa solusi hosting dan database.




Gambar 4.1. Layanan PostgreSQL dari PostgresML

Database postgresql yang digunakan adalah layanan gratis dari postgresml.org. Layanan ini pada saat penulisan memiliki 1 CPU, 1 GPU, 8GB RAM, dan 5GB storage. Layanan ini dipilih agar kebutuhan storage dan RAM yang cukup tinggi sistem rekomendasi berbasis SQL dapat terpenuhi. Ini dapat dilihat pada Gambar 4.1.



Gambar 4. 2. Layanan Redis dari Redislabs

Untuk Redis, digunakan layanan free dari redislabs.com atau lebih umum dikenal sebagai redis.com. Layanan ini memiliki memory maksimum 30MB dan persistence. Layanan ini dapat dilihat pada Gambar 4. 2.

 Services From \$0 USD / month		<ul style="list-style-type: none"> ✓ Web services with HTTP/2 and full TLS ✓ Private services ✓ Background workers 	<ul style="list-style-type: none"> ✓ Node, Python, Go, Rust, Ruby, and Elixir ✓ Custom Docker containers ✓ SSD disks for \$0.25/GB per month
Web Services	Private Services	Background Workers	
Instance Type	Pricing	RAM	CPU
Free	\$0/month <i>with limits</i>	512 MB	0.1

Gambar 4.3. Layanan webservice dari Render.com

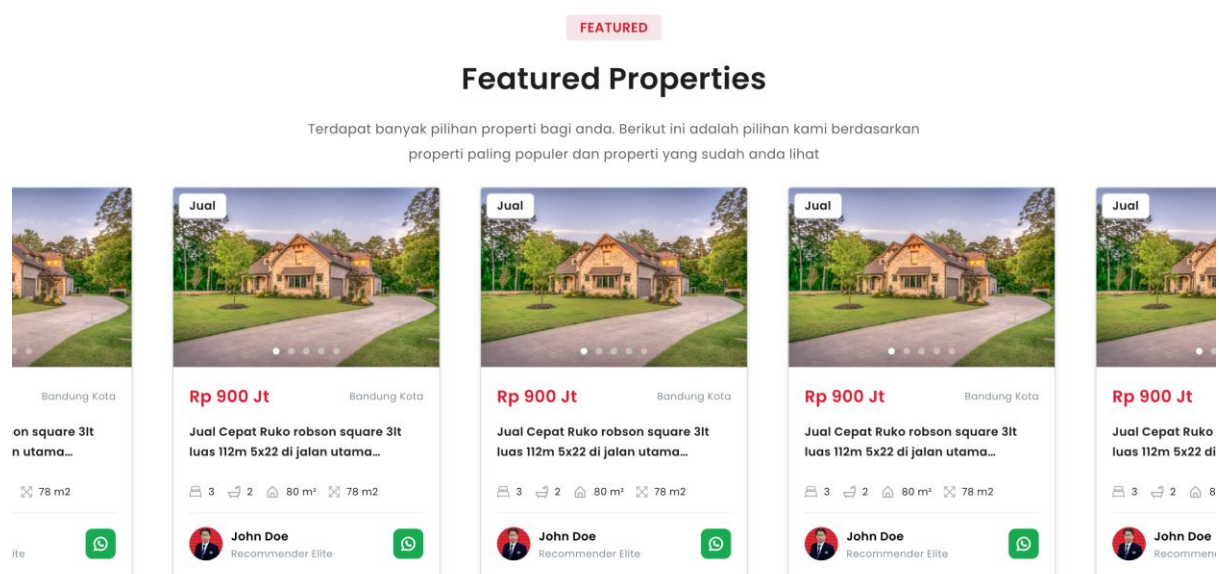
Untuk hosting api dan web, dilakukan dengan layanan dari render.com yang memiliki beberapa fitur penting yaitu auto deploy dari github setelah konfigurasi, aplikasi otomatis HTTP/2 dan TLS untuk HTTPS, dan kemampuan untuk menjalankan environment linux yang berbeda dengan layanan edge gratis seperti vercel yang tidak dapat menjalankan beberapa binary linux tanpa konfigurasi ekstensif. Selain ini, render.com memberikan running time 750 jam perbulan dan RAM 512MB. Fitur HTTPS otomatis penting karena beberapa browser modern seperti chrome membutuhkannya untuk dapat memberi client cookie dari third party domain yang dalam kasus ini adalah API sistem rekomendasi. Cookie ini diperlukan untuk mencatat

pengunjung mana yang berinteraksi dengan website. Layanan ini dapat dilihat pada Gambar 4.3.

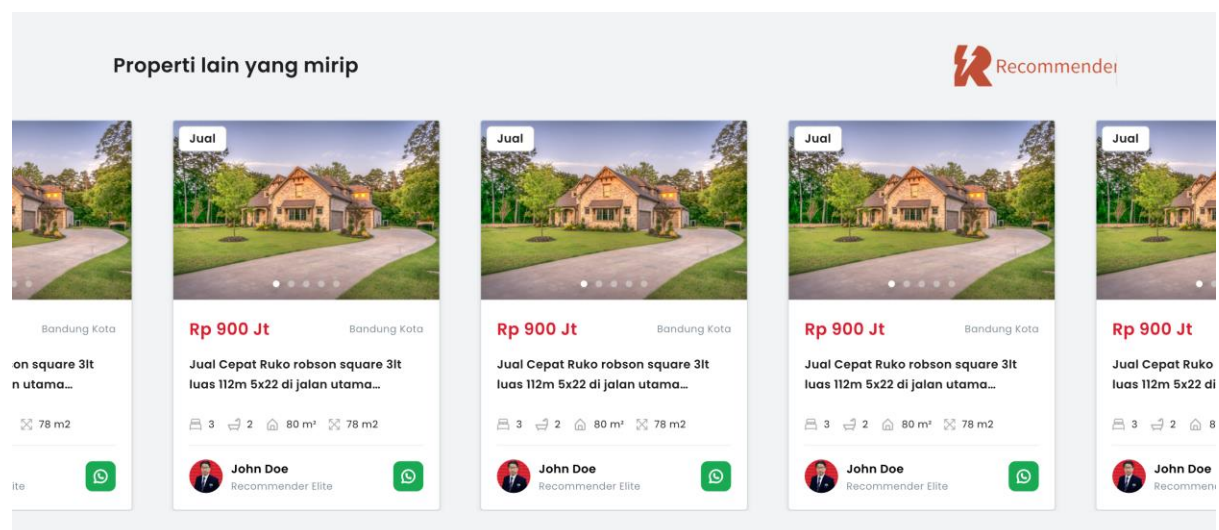
4.2. Hasil Implementasi

4.2.1. Desain User Interface

Karena batasan kerahasiaan untuk klien, desain figma untuk website tidak dapat diperlihatkan dalam laporan ini. Namun untuk desain user interface (UI) sistem rekomendasi sendiri dapat ditampilkan dalam laporan ini. Desainnya adalah sebagaimana ditampilkan pada Gambar 4.4 dan Gambar 4.5.



Gambar 4.4. UI featured properties di landing page

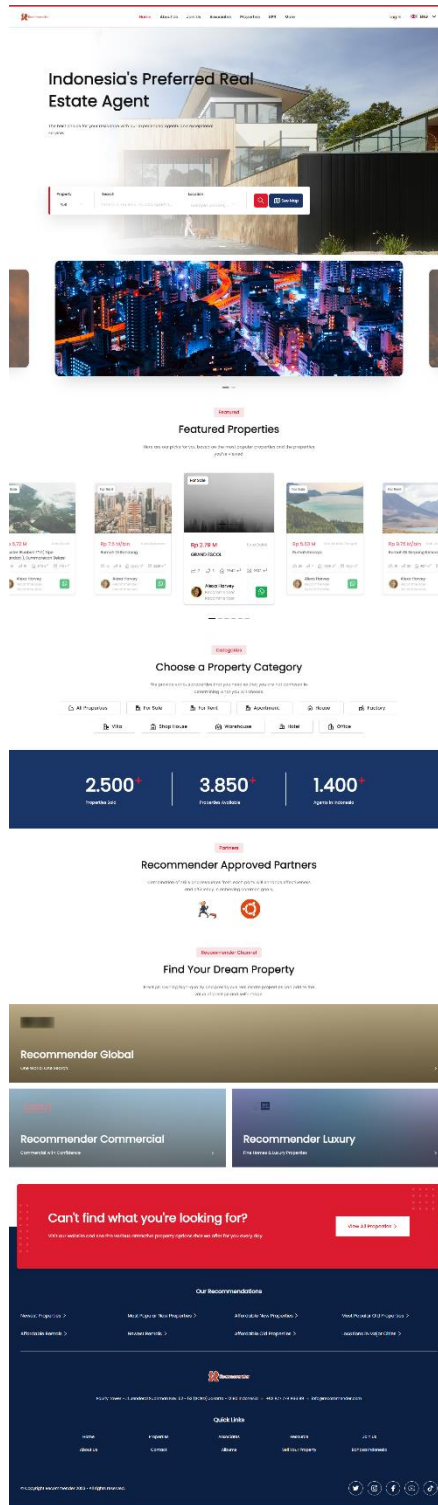


Gambar 4.5. UI recommended listings di halaman listing detail

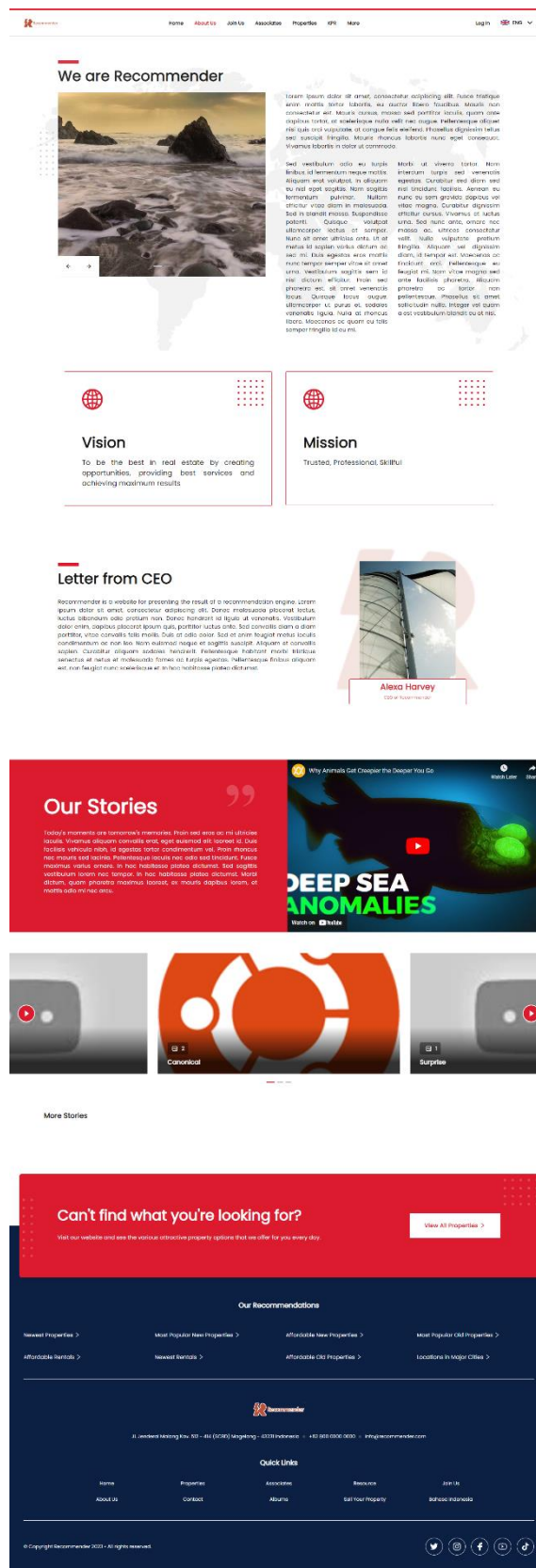
4.2.2. Website Rekomendasi

Hasil dari PPL adalah dalam bentuk website. Website tersebut dapat dilihat pada <https://recommender-web.onrender.com>. Website sistem rekomendasi ini dikerjakan dengan developer lain dari Qbit. Beberapa halaman yang telah dibikin oleh penulis, yaitu:

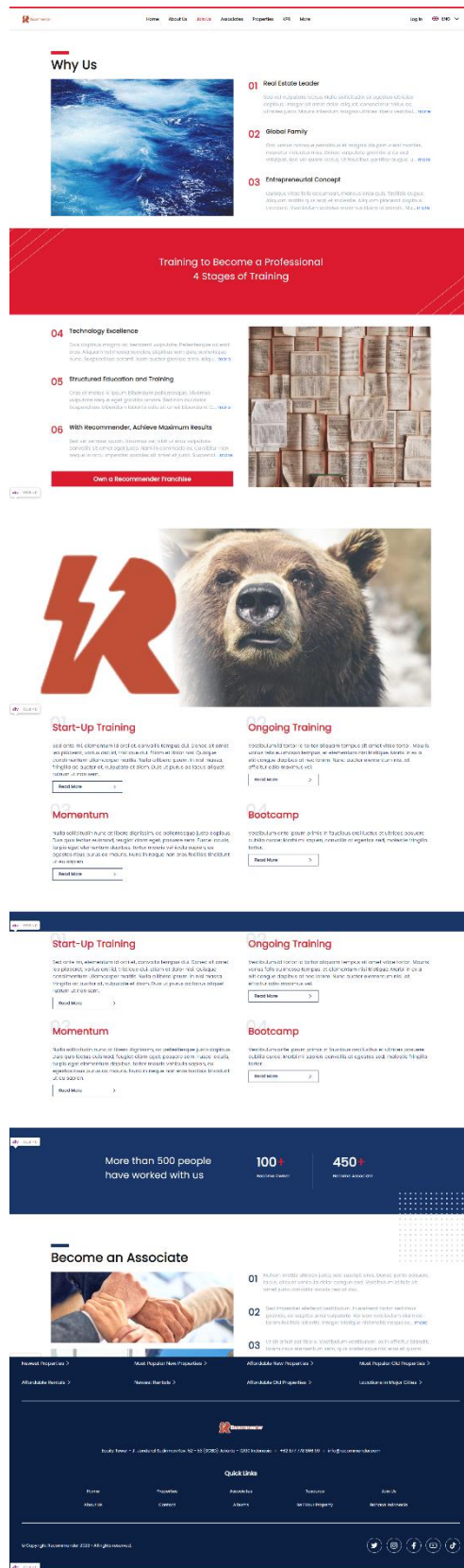
1. Home: /



2. About us: /about-us, dan Albums: /about-us#album



3. Join us: /join-us, dan Associates: /join-us#associates



4. Property Terms: /istilah-property

Recommend

Home

About Us

Join Us

Associates

Properties

KPR

More

Log in

ENG

Terms

AJB, Agen property, Agunan ...

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

Y

A

There are 9 terms

AJB

Akta Jual Beli. AJB Merupakan surat keterangan bahwa satu properti sudah dipindahtangankan dari satu pihak ke pihak lain karena aktivitas jual beli.

Agen property

Pihak atau orang yang menjembatani antara penjual dan pembeli properti, pada dasarnya agen properti ada yang tergabung dalam sebuah kantor agen properti ataupun individu.

Agunan (collateral)

Aset milik peminjam yang diberikan kepada pihak bank, aset tersebut nantinya akan menjadi milik bank jika peminjam gagal untuk membayar cicilan.

Dalam skema pengajuan KPR biasanya agunan yang diberikan kepada bank adalah sertifikat tanah.

Amortisasi

Pengurangan utang berjalan dengan membayar hutang secara tetap dengan jumlah yang sama. Dengan amortisasi, pembayaran utang terdiri atas pembayaran pokok (principal) dan pembayaran bunga (interest).

Can't find what you're looking for?

Visit our website and see the various attractive property options that we offer for you every day.

View All Properties >

Our Recommendations

Newest Properties >

Most Popular New Properties >

Affordable New Properties >

Most Popular Old Properties >

Affordable Rentals >

Newest Rentals >

Affordable Old Properties >

Locations in Major Cities >

Recommend

Equity Tower - Jl. Jenderal Sudirman Kav. 52 - 53 (SCBD) Jakarta - 12190 Indonesia • +62 877 778 886 99 • info@recommender.com

Quick Links

Home

About Us

Properties

Contact

Associates

Albums

Resource

Sell Your Property

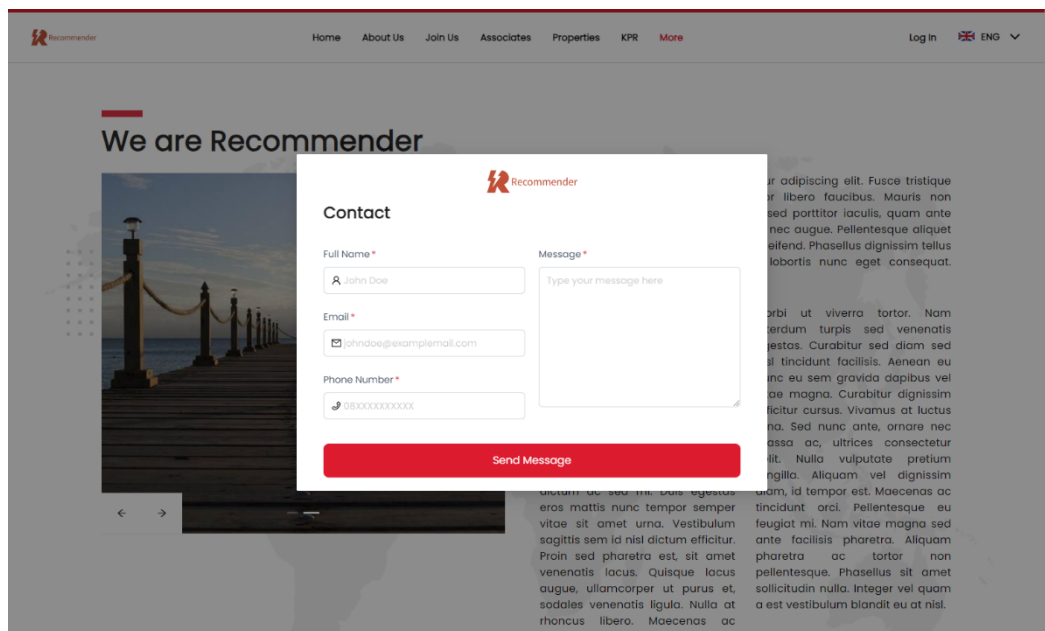
Join Us

Bahasa Indonesia

© Copyright Recommender 2023 • All rights reserved.

36

5. Contact: modal in menu



UI untuk rekomendasi diimplementasi di halaman Home sebagai featured recommendation yang berbasis collaborative filtering. UI rekomendasi juga diimplementasi di halaman detail listing. Rekomendasi disini berbasis hybrid recommendation berdasarkan listing yang halaman detailnya sedang dibuka.

4.2.3. API Rekomendasi

Di sisi backend, terdapat beberapa endpoint diimplementasikan khusus untuk rekomendasi. API ini dapat di akses di <https://recommender-api-b5o1.onrender.com>. Yang pertama adalah untuk mencatat interaksi pengunjung dengan website. Endpoint ini terdapat di POST /listing-visitors/interaction. Endpoint ini memerlukan beberapa parameter sebagai input dalam body requestnya, yaitu:

```
{  
  "listingId": UUID Listing yang pengunjung sedang berinteraksi,  
  "referrerURL": URL sebelumnya yang mengarahkan ke halaman ini,  
  "originURL": URL yang mengirimkan request interaksi tersebut,  
  "originIpAddress": IP Address pengunjung,  
  "listingVisitInteractionType": Jenis interaksi yang terjadi  
}
```

Request di atas harus dikirim dengan cookie berisi visitorId pengunjung jika sudah memiliki. Response dari endpoint ini adalah code jika success atau tidak, message, dan payload yang berisi data interaksi serta data dari kolom yang di autogenerate seperti listingVisitInteractionId. Selain body, response ini juga memberikan cookie untuk mengidentifikasi visitor yang sama pada interaksi selanjutnya.

Rekomendasi sendiri didapatkan dengan request ke GET /listings/recommendation. Endpoint ini memiliki beberapa parameter query yaitu:

```
{  
  "limit": Jumlah maksimum rekomendasi yang diperlukan atau kosong jika tidak perlu batas  
  spesifik,  
  "listingId": UUID listing yang digunakan sebagai referensi untuk rekomendasi jika perlu  
}
```

Request di atas juga harus dikirim dengan cookie berisi visitorId pengunjung jika sudah memiliki. Response dari endpoint di atas adalah object dengan code success atau tidak, message, dan payload berisi count dari jumlah listing yang direkomendasikan dan array result yang berisi listing dan data detailnya.

Selain sistem rekomendasi, penulis juga mengerjakan fitur lain yaitu migrasi data klien. Untuk menjaga privasi klien, detail spesifik migrasi tidak dapat dituliskan dalam laporan ini. Namun secara umum, setelah beberapa revisi, sistem migrasi berkerja sebagai berikut:

1. Data memberikan dump dari database lama
2. Data lama diproses menjadi bentuk excel
 - a. Data boolean diubah jadi ya tidak
 - b. Data categorical diubah dari id nya ke string literal spesifik untuk tiap kategori, seperti "rumah", "toko", dst.
3. File excel ini di upload ke backend melalui endpoint di API

Proses import ini dibagi menjadi dua:

- a. Dry run
 - a. File di validasi bentuk nya dan format datanya sudah sesuai
 - b. Data yang tidak sesuai di catat

- c. Data juga divalidasi relasinya pada dengan transaksi yang mencoba insert data tersebut dan kemudian langsung di rollback
- d.
- e. Jika transaksi gagal, maka dicatat juga kegagalan tersebut
- f. Jika ada catatan masalah dengan data maka di file dikembalikan dengan catatan kesalahan tersebut
- b. Wet run
 - a. Jika tidak ada kesalahan pada data, maka dalam sebuah transaksi dijalankan bulk insert data.
 - b. Dan jika transaksi berhasil, user mendapat konfirmasi sukses.
 - c. Dan jika transaksi gagal maka eror akan dilaporkan ke user.
- 4. Kesalahan tercatat pada file excel yang dikembalikan diperbaiki oleh klien.
- 5. Diulangi dari No.3 hingga hasil upload sukses.

4.2.4. Logika Rekomendasi

Rekomendasi sendiri diimplementasi di dalam server PostgreSQL. Ini menghindari beban transfer data antara server dan database. Namun ini dengan catatan bahwa database PostgreSQL yang digunakan di host di PostgresML yang memiliki konfigurasi sehingga database memiliki sumber daya komputasi seperti RAM, CPU, dan GPU yang memadai untuk pekerjaan machine learning. Karena itu, dimungkinkan penggunaan database untuk komputasi rekomendasi. Logika ini dibagi menjadi dua bagian. Untuk komputasi content-based, ternama skor similarity antar listing, dilakukan dengan cara lazy yang hanya melakukan komputasi untuk suatu listing jika listing tersebut diminta similaritynya dengan listing lain. Kode ini diimplementasi dalam bentuk fungsi PostgreSQL yang badan dari fungsinya sebagaimana di LAMPIRAN A. Kode tersebut ditulis menggunakan bantuan dari code generation berdasarkan jawaban StackOverflow berikut (<https://stackoverflow.com/a/56068467>) untuk one hot encoding data kategori. Kemudian, untuk rekomendasi collaborative, digunakan query sebagaimana di LAMPIRAN B. Namun, karena collaborative filtering memiliki masalah cold start seperti yang dibahas pada 3.1.4, juga diimplementasi rekomendasi berdasarkan popularitas listing seperti di LAMPIRAN C. Setelah itu, untuk menyatukan rekomendasi collaborative dan content-based untuk di halaman detail listing, juga diimplementasi rekomendasi hybrid seperti di LAMPIRAN D.

4.3. Pengujian

Pengujian merupakan tahapan penting dalam menghasilkan sistem rekomendasi yang handal dan efektif. Pada sub bab ini, akan dibahas mengenai proses pengujian yang dilakukan dalam konteks proyek sistem rekomendasi yang telah dilakukan selama kegiatan magang. Tujuan dari pengujian adalah untuk memastikan bahwa sistem rekomendasi ini berfungsi dengan baik, memberikan rekomendasi yang relevan, serta memenuhi kebutuhan dan harapan pengguna.

Pada sub bab ini, akan dijelaskan mengenai metode pengujian yang digunakan, kriteria pengujian yang telah ditentukan, serta hasil pengujian yang diperoleh. Metode pengujian yang digunakan mencakup pengujian rekomendasi, pengujian interaksi pengguna, pengujian antarmuka pengguna (UI), dan pengujian kompatibilitas. Setiap metode pengujian akan dijelaskan secara rinci, termasuk langkah-langkah yang dilakukan, data yang digunakan, dan skenario pengujian yang dilakukan.

Selain itu, juga akan dibahas mengenai hasil pengujian yang diperoleh, termasuk evaluasi kinerja sistem rekomendasi berdasarkan kriteria pengujian yang telah ditetapkan. Hasil pengujian ini akan memberikan gambaran tentang sejauh mana sistem rekomendasi dapat memenuhi kebutuhan yang telah ditetapkan dalam proyek ini.

1. Pengujian Rekomendasi:

a.

Hasil diharapkan	Sistem mampu memberikan rekomendasi properti yang relevan berdasarkan sejarah user dan properti yang sedang dilihat.
Hasil	Sistem mampu memberikan rekomendasi properti yang relevan berdasarkan sejarah user dan properti yang sedang dilihat.
Catatan	Karena jumlah listing yang ada untuk testing terbatas, listing yang mirip dengan listing yang sedang dilihat terbatas pada listing pertama. Selain itu, teks title tidak digunakan untuk rekomendasi karena masalah dengan <code>ts_vector</code> yang memberikan rank rendah meskipun untuk string yang sama.

b.

Hasil diharapkan	Sistem mampu memilih properti yang relevan berdasarkan sejarah penggunaan user.
------------------	---

Hasil	Sistem mampu memilih properti yang relevan berdasarkan sejarah penggunaan user.
Catatan	Ini hanya di test dengan dua user dan user pertama memiliki beberapa listing yang belum pernah dilihat oleh user kedua.

2. Pengujian Interaksi Pengguna:

a.

Hasil diharapkan	Interaksi pengguna dengan sistem seperti melihat detail listing atau share listing disimpan oleh sistem.
Hasil	Interaksi pengguna dengan sistem seperti melihat detail listing atau share listing disimpan oleh sistem.
Catatan	

3. Pengujian Antarmuka Pengguna (UI):

a.

Hasil diharapkan	Memastikan antarmuka pengguna menampilkan informasi dengan benar dan sesuai dengan desain yang telah ditentukan.
Hasil	Memastikan antarmuka pengguna menampilkan informasi dengan benar dan sesuai dengan desain yang telah ditentukan.
Catatan	

b.

Hasil diharapkan	Fungsi interaktif antarmuka pengguna tombol pencarian, card listing, carousel rekomendasi, loading data, dan menu dapat digunakan sesuai spesifikasi.
Hasil	Fungsi interaktif antarmuka pengguna tombol pencarian, card listing, carousel rekomendasi, loading data, dan menu dapat digunakan sesuai spesifikasi.
Catatan	

4. Pengujian Kompatibilitas:

a.

Hasil diharapkan	Sistem rekomendasi dapat berfungsi dengan baik di browser desktop dan mobile yang umum digunakan.
Hasil	Sistem rekomendasi dapat berfungsi dengan baik di browser desktop dan mobile yang umum digunakan.
Catatan	

BAB V

PENUTUP

5.1. Kesimpulan

Dalam proyek sistem rekomendasi ini, tujuan utama yang berhasil dicapai adalah pengembangan sistem rekomendasi yang mampu memberikan rekomendasi properti yang relevan kepada pengguna. Sistem ini mampu menggabungkan dua metode rekomendasi utama, yaitu collaborative filtering dan content-based filtering sehingga menghasilkan rekomendasi yang lebih akurat dan personal. Terutama untuk rekomendasi di halaman detail listing yang menggunakan rekomendasi hybrid.

Selama proses pengembangan proyek, saya juga telah mempelajari dan mengimplementasikan berbagai teknologi dan alat yang digunakan oleh Perusahaan Qbit, termasuk penggunaan framework NestJS, NextJS (ReactJS), PostgreSQL, MySQL, dan Redis. Saya juga memperoleh pemahaman yang lebih baik mengenai dan penerapan pola desain controller, provider, & module di backend serta pola component & hooks di frontend dalam proyek ini. Selain itu, saya juga mendalami PostgreSQL untuk mengimplementasi logika sistem rekomendasi.

Dalam kesimpulan ini, saya juga ingin menyoroti beberapa pelajaran penting yang telah saya peroleh selama proses magang ini. Saya belajar mengenai manajemen proyek, pengembangan perangkat lunak secara tim, serta pemahaman yang lebih dalam mengenai industri real estate, dan sistem rekomendasi. Selain itu, saya juga belajar untuk beradaptasi dengan lingkungan kerja yang dinamis, mengatasi tantangan teknis, dan terus meningkatkan kemampuan komunikasi dan kolaborasi.

Secara keseluruhan, magang di Perusahaan Qbit telah memberikan pengalaman yang berharga dalam pengembangan proyek sistem rekomendasi. Saya berterima kasih kepada tim Perusahaan Qbit atas kesempatan ini dan pengarahan yang diberikan selama magang. Saya yakin pengalaman ini akan sangat berharga dalam perjalanan karir saya di bidang pengembangan perangkat lunak dan sistem rekomendasi.

5.2. Saran

Khusus untuk implementasi sistem rekomendasi, masih banyak yang dapat dilakukan untuk perbaikan kedepannya. Terdapat empat poin utama yang dapat diperbaiki, yaitu:

Di bagian saran pada bab penutup, Anda dapat menuliskan beberapa hal berikut terkait fitur yang belum diimplementasikan dan area yang dapat ditingkatkan:

1. Implementasi Fitur "Like" untuk Listing: Disarankan untuk melanjutkan implementasi fitur "Like" untuk listing properti. Fitur ini memungkinkan pengguna untuk menandai dan menyimpan listing properti yang disukai atau menarik bagi mereka. Dengan mengimplementasikan fitur ini, pengguna dapat memiliki daftar properti yang telah mereka sukai dan dapat mengaksesnya kapan saja.
2. Integrasi Recommendation System ke dalam Pencarian: Saran ini menyarankan untuk mengintegrasikan sistem rekomendasi ke dalam fitur pencarian properti. Dengan melibatkan sistem rekomendasi, hasil pencarian dapat ditingkatkan dengan memberikan rekomendasi properti yang sesuai berdasarkan preferensi pengguna dan perilaku mereka. Integrasi ini akan meningkatkan pengalaman pengguna dan membantu mereka menemukan properti yang lebih relevan dengan lebih mudah.
3. Peningkatan Tuning Bobot untuk Similarity pada Content-Based Recommendation: Saat ini, tuning bobot untuk similarity pada content-based recommendation dilakukan secara manual. Saran ini adalah untuk melakukan pengembangan lebih lanjut pada mekanisme tuning bobot dengan pendekatan yang lebih otomatis atau menggunakan metode pembelajaran mesin. Hal ini akan membantu meningkatkan akurasi dan keefektifan sistem rekomendasi dalam menemukan properti yang serupa berdasarkan konten.
4. Encoding Data Kategori yang Dinamis: Mengatasi masalah encoding data kategori yang statis dengan mengembangkan sistem yang dapat mengubah kategori secara dinamis berdasarkan perubahan di database. Ini akan memungkinkan sistem rekomendasi untuk tetap mengikuti perubahan dan pembaruan dalam kategori properti yang ada, memastikan keakuratan dan relevansi rekomendasi yang diberikan kepada pengguna.

Selain itu, juga disarankan untuk melakukan pengujian lebih lanjut pada sistem rekomendasi yang telah dikembangkan untuk mengukur kinerja, akurasi, dan kepuasan pengguna. Evaluasi secara teratur dan memperbaiki sistem rekomendasi berdasarkan umpan balik pengguna juga merupakan saran yang berguna.

DAFTAR PUSTAKA

- Aggarwal, C. C. (2016). *Recommender Systems*. Springer International Publishing.
<https://doi.org/10.1007/978-3-319-29659-3>
- Al-Oud, M. E. (2008). *Designing and Evaluating a Recommender System within the Book Domain*. University of Essex.
- Arkandana, M. T., Suganda, T. G., & Pramana, S. (2021). Hubungan Jumlah Tayangan Iklan Penawaran Penjualan dan Penyewaan Properti dengan PDRB Provinsi Bali Tahun 2019-2021 Dengan Menggunakan Big Data: Web Scraping. *Seminar Nasional Official Statistics*, 2021(1), 1083–1092. <https://doi.org/10.34123/semnasoffstat.v2021i1.1056>
- Burke, R. (2002). Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, 12(4), 331–370.
<https://doi.org/10.1023/A:1021240730564>
- Da Silva, M. D., & Tavares, H. L. (2015). *Redis essentials: Harness the power of Redis to integrate and manage your projects efficiently*. Packt Publishing.
- Daryanto, W. M., Samidi, S., & Siregar, D. J. (2018). The impact of financial liquidity and leverage on financial performance: Evidence from property and real estate enterprises in Indonesia. *Management Science Letters*, 1345–1352.
<https://doi.org/10.5267/j.msl.2018.9.005>
- Juba, S., Volkov, A., & Vannahme, A. (2015). *Learning PostgreSQL: Create, develop, and manage relational databases in real-world applications using PostgreSQL*. Packt Publishing.
- Ku, Y.-C., & Tai, Y.-M. (2013). What Happens When Recommendation System Meets Reputation System? The Impact of Recommendation Information on Purchase Intention. *2013 46th Hawaii International Conference on System Sciences*, 1376–1383.
<https://doi.org/10.1109/HICSS.2013.605>

- Nanou, T., Lekakos, G., & Fouskas, K. (2010). The effects of recommendations' presentation on persuasion and satisfaction in a movie recommender system. *Multimedia Systems*, 16(4–5), 219–230. <https://doi.org/10.1007/s00530-010-0190-0>
- NestJS. (2023). *Documentation / NestJS - A progressive Node.js framework*. <https://docs.nestjs.com/>

LAMPIRAN A

Kode badan fungsi komputasi similarity listing

```
with utils as (
  select
    "stats" .*,
    (((("stats"."minBuildingArea" * ${buildingAreaScaler})-
"stats"."maxBuildingArea")/("stats"."minBuildingArea"-
"stats"."maxBuildingArea")) "buildingAreaT1",
    "stats"."maxBuildingArea"- "stats"."minBuildingArea"
"rangeBuildingArea",
    ${buildingAreaScaler} as "buildingAreaScaler",
    ln(${buildingAreaScaler}) as "lnBuildingAreaScaler",
    (((("stats"."minLandArea" * ${landAreaScaler})-
"stats"."maxLandArea")/("stats"."minLandArea"- "stats"."maxLandArea"))
"landAreaT1",
    "stats"."maxLandArea"- "stats"."minLandArea" "rangeLandArea",
    ${landAreaScaler} as "landAreaScaler",
    ln(${landAreaScaler}) as "lnLandAreaScaler",
    (((("stats"."minNumberOfFloor" * ${numberOfFloorScaler})-
"stats"."maxNumberOfFloor")/("stats"."minNumberOfFloor"-
"stats"."maxNumberOfFloor")) "numberOfFloorT1",
    "stats"."maxNumberOfFloor"- "stats"."minNumberOfFloor"
"rangeNumberOfFloor",
    ${numberOfFloorScaler} as "numberOfFloorScaler",
    ln(${numberOfFloorScaler}) as "lnNumberOfFloorScaler",
    (((("stats"."minMainBedroom" * ${mainBedroomScaler})-
"stats"."maxMainBedroom")/("stats"."minMainBedroom"-
"stats"."maxMainBedroom")) "mainBedroomT1",
    "stats"."maxMainBedroom"- "stats"."minMainBedroom"
"rangeMainBedroom",
    ${mainBedroomScaler} as "mainBedroomScaler",
    ln(${mainBedroomScaler}) as "lnMainBedroomScaler",
```

```

        (((("stats"."minMainBathroom" * ${mainBathroomScaler})-
"stats"."maxMainBathroom")/("stats"."minMainBathroom"-
"stats"."maxMainBathroom")) "mainBathroomT1",
        "stats"."maxMainBathroom"- "stats"."minMainBathroom"
"rangeMainBathroom",
        ${mainBathroomScaler} as "mainBathroomScaler",
        ln(${mainBathroomScaler}) as "lnMainBathroomScaler",
        (((("stats"."minHousekeepBedroom" * ${housekeepBedroomScaler})-
"stats"."maxHousekeepBedroom")/("stats"."minHousekeepBedroom"-
"stats"."maxHousekeepBedroom")) "housekeepBedroomT1",
        "stats"."maxHousekeepBedroom"- "stats"."minHousekeepBedroom"
"rangeHousekeepBedroom",
        ${housekeepBedroomScaler} as "housekeepBedroomScaler",
        ln(${housekeepBedroomScaler}) as "lnHousekeepBedroomScaler",
        (((("stats"."minHousekeepBathroom" * ${housekeepBathroomScaler})-
"stats"."maxHousekeepBathroom")/("stats"."minHousekeepBathroom"-
"stats"."maxHousekeepBathroom")) "housekeepBathroomT1",
        ("stats"."maxHousekeepBathroom"- "stats"."minHousekeepBathroom")
"rangeHousekeepBathroom",
        ${housekeepBathroomScaler} as "housekeepBathroomScaler",
        ln(${housekeepBathroomScaler}) as "lnHousekeepBathroomScaler",
        (((("stats"."minPrice" * ${priceScaler})-
"stats"."maxPrice")/("stats"."minPrice"- "stats"."maxPrice")) "priceT1",
        "stats"."maxPrice"- "stats"."minPrice" "rangePrice",
        ${priceScaler} as "priceScaler",
        ln(${priceScaler}) as "lnPriceScaler"
from
    (
    select
        max("listings"."buildingArea") "maxBuildingArea",
        min("listings"."buildingArea") "minBuildingArea",
        max("listings"."landArea") "maxLandArea",
        min("listings"."landArea") "minLandArea",
        max("listings"."numberOfFloor") "maxNumberOfFloor",

```

```

        min("listings"."numberOfFloor") "minNumberOfFloor",
        max("listings"."mainBedroom") "maxMainBedroom",
        min("listings"."mainBedroom") "minMainBedroom",
        max("listings"."mainBathroom") "maxMainBathroom",
        min("listings"."mainBathroom") "minMainBathroom",
        max("listings"."housekeepBedroom") "maxHousekeepBedroom",
        min("listings"."housekeepBedroom") "minHousekeepBedroom",
        max("listings"."housekeepBathroom") "maxHousekeepBathroom",
        min("listings"."housekeepBathroom") "minHousekeepBathroom",
        max("listings"."price") "maxPrice",
        min("listings"."price") "minPrice"

    from

        listings) as "stats"
),
query as (
-- construct a query context with arguments that would typically be
-- passed in from the application layer
select
    -- a keyword query for "my" OR "search" OR "terms"
    "listings"."listingId",
    websearch_to_tsquery('simple',
    "listings"."title") as title_query,
    "listings"."title",
    websearch_to_tsquery('simple',
    "listings"."descriptionPurified") as description_query,
    "listings"."descriptionPurified",
    websearch_to_tsquery('simple',
    "listings"."keyword") as keyword_query,
    "listings"."keyword",
    "listings"."country",
    "listings"."province",
    "listings"."city",
    "listings"."district"
from

```

```

    listings
where
    "listings"."listingId" = ${listingIdParamName}
),
listing_ohe as (
-- construct a query one-hot-encoding
select
    "listings" .*,
        (case
            when "listings"."sellOrRent" = 'rent' then 1
            else 0
        end) "sellOrRent_rent",
        (case
            when "listings"."sellOrRent" = 'sell' then 1
            else 0
        end) "sellOrRent_sell",
        (case
            when "listings"."listingType" = 'primary' then 1
            else 0
        end) "listingType_primary",
        (case
            when "listings"."listingType" = 'secondary' then 1
            else 0
        end) "listingType_secondary",
        (case
            when "listings"."listingPartId" = '16ad93ae-5e13-4f1d-b972-
96e90762a141' then 1
            else 0
        end) "listingPartId_Residential",
        (case
            when "listings"."listingPartId" = '79a346e0-ddbf-42f7-90e9-
70458e019833' then 1
            else 0
        end) "listingPartId_Industrial",

```

```

        (case
            when "listings"."listingPartId" = '9d5b7082-d39d-4a3a-b18c-
54586c99a8f4' then 1
            else 0
        end) "listingPartId_Commercial",
        (case
            when "listings"."listingPartId" = 'dedfde9a-5abf-4513-a741-
212c2d04b839' then 1
            else 0
        end) "listingPartId_Luxury",
        (case
            when "listings"."listingPropertyTypeId" = '431e390d-6810-426f-
9043-45ab0882b15f' then 1
            else 0
        end) "listingPropertyTypeId_Gudang",
        (case
            when "listings"."listingPropertyTypeId" = '4e1d2fad-5c7f-48a2-
b661-ac99059a8e19' then 1
            else 0
        end) "listingPropertyTypeId_Pabrik",
        (case
            when "listings"."listingPropertyTypeId" = '66d1b1f2-44c7-48a0-
a8de-86f47811186f' then 1
            else 0
        end) "listingPropertyTypeId_Rumah",
        (case
            when "listings"."listingPropertyTypeId" = '7fe2a087-2455-476b-
b4f9-30b7a6e5ee9a' then 1
            else 0
        end) "listingPropertyTypeId_Apartment",
        (case
            when "listings"."listingPropertyTypeId" = '8d94f930-30cd-4903-
ad4c-0140620da316' then 1
            else 0

```



```

end) "listingPropertyTypeId_RukoRukan",
    (case
        when "listings"."listingPropertyTypeId" = 'a23bb102-b2cc-401c-
8255-0e1c1f6fc274' then 1
        else 0
    end) "listingPropertyTypeId_Kios",
    (case
        when "listings"."listingPropertyTypeId" = 'a4748d44-75a1-4bd9-
85eb-af24a4acbd3a' then 1
        else 0
    end) "listingPropertyTypeId_Perkantoran",
    (case
        when "listings"."listingPropertyTypeId" = 'c8278c9d-66aa-4996-
a02f-e1d6113e96a0' then 1
        else 0
    end) "listingPropertyTypeId_Condotel",
    (case
        when "listings"."listingPropertyTypeId" = 'de9d1857-4af1-490a-
9872-46dca83dcf09' then 1
        else 0
    end) "listingPropertyTypeId_VillaResort",
    (case
        when "listings"."listingCertificateTypeId" = '0cffdaaa-bc1b-
497e-825b-1d0435ce70b6' then 1
        else 0
    end) "listingCertificateTypeId_HakMilikSaruanStrataTitle",
    (case
        when "listings"."listingCertificateTypeId" = '636f26c7-47da-
49c9-aa08-aabe803aef8' then 1
        else 0
    end) "listingCertificateTypeId_HakPengelolaanLahanHPL",
    (case
        when "listings"."listingCertificateTypeId" = '6f6a512f-03b2-
4624-afe7-a50533297ebc' then 1

```

```

        else 0
    end) "listingCertificateTypeId_SertifikatHakMilikSHM",
        (case
            when "listings"."listingCertificateTypeId" = '9c4a21f2-925e-
43c1-b577-ab71a3241c30' then 1
            else 0
        end) "listingCertificateTypeId_HakPakaiLahan",
        (case
            when "listings"."listingCertificateTypeId" = 'a96f1692-d5d3-
4c1f-bea5-1febd520008b' then 1
            else 0
        end) "listingCertificateTypeId_Girik",
        (case
            when "listings"."listingCertificateTypeId" = 'd76141e4-c221-
46ff-a7ce-1b033b2d1756' then 1
            else 0
        end) "listingCertificateTypeId_PerjanjianPengikatanJualBelliPPJB",
        (case
            when "listings"."listingCertificateTypeId" = 'e6119074-7536-
40e7-a415-2b8e923b177f' then 1
            else 0
        end) "listingCertificateTypeId_HakGunaBangunanHGB",
        ln("utils"."buildingAreaT1" + ((1-
"utils"."buildingAreaScaler")* "listings"."buildingArea") /(-
"utils"."rangeBuildingArea"))/ "utils"."lnBuildingAreaScaler"
"buildingArea_scaled",
        ln("utils"."landAreaT1" + ((1-"utils"."landAreaScaler")*
"listings"."landArea") /(-"utils"."rangeLandArea"))/
"utils"."lnLandAreaScaler" "landArea_scaled",
        ln("utils"."numberOfFloorT1" + ((1-
"utils"."numberOfFloorScaler")* "listings"."numberOfFloor") /(-
"utils"."rangeNumberOfFloor"))/ "utils"."lnNumberOfFloorScaler"
"numberOfFloor_scaled",

```

```

        ln("utils"."mainBedroomT1" + ((1-"utils"."mainBedroomScaler")*
"listings"."mainBedroom") /(-"utils"."rangeMainBedroom"))/
"utils"."lnMainBedroomScaler" "mainBedroom_scaled",
        ln("utils"."mainBathroomT1" + ((1-
"utils"."mainBathroomScaler")* "listings"."mainBathroom") /(-
"utils"."rangeMainBathroom"))/ "utils"."lnMainBathroomScaler"
"mainBathroom_scaled",
        ln("utils"."housekeepBedroomT1" + ((1-
"utils"."housekeepBedroomScaler")* "listings"."housekeepBedroom") /(-
"utils"."rangeHousekeepBedroom"))/ "utils"."lnHousekeepBedroomScaler"
"housekeepBedroom_scaled",
        ln("utils"."housekeepBathroomT1" + ((1-
"utils"."housekeepBathroomScaler")* "listings"."housekeepBathroom") /(-
"utils"."rangeHousekeepBathroom"))/ "utils"."lnHousekeepBathroomScaler"
"housekeepBathroom_scaled",
        ln("utils"."priceT1" + ((1-"utils"."priceScaler")*
"listings"."price") /(-"utils"."rangePrice"))/ "utils"."lnPriceScaler"
"price_scaled"
    from
        listings,
        utils
    ),
    listing_data as (
    select
        "listing_ohe" .*,
        array["sellOrRent_rent"::real,
"sellOrRent_sell"] "embedding_sellOrRent",
        array["listingType_primary"::real,
"listingType_secondary",
"listingPartId_Residential",
"listingPartId_Industrial",
"listingPartId_Commercial",
"listingPartId_Luxury"] "embedding_listingPart",
        array["listingPropertyTypeId_Gudang"::real,

```

```

        "listingPropertyTypeId_Pabrik",
        "listingPropertyTypeId_Rumah",
        "listingPropertyTypeId_Apartment",
        "listingPropertyTypeId_RukoRukan",
        "listingPropertyTypeId_Kios",
        "listingPropertyTypeId_Perkantoran",
        "listingPropertyTypeId_Condotel",
        "listingPropertyTypeId_VillaResort"]
"embedding_listingPropertyType",
    array["listingCertificateTypeId_HakMilikSaruanStrataTitle "::real,
        "listingCertificateTypeId_HakPengelolaanLahanHPL",
        "listingCertificateTypeId_SertifikatHakMilikSHM",
        "listingCertificateTypeId_HakPakaiLahan",
        "listingCertificateTypeId_Girik",
        "listingCertificateTypeId_PerjanjianPengikatanJualBelliPPJB",
        "listingCertificateTypeId_HakGunaBangunanHGB"]
"embedding_listingCertificateType",
    array["buildingArea_scaled "::real,
        "landArea_scaled",
        "numberOfFloor_scaled",
        "mainBedroom_scaled",
        "mainBathroom_scaled",
        "housekeepBedroom_scaled",
        "housekeepBathroom_scaled",
        "price_scaled"] "embedding"
from
    listing_ohe
),
listing_data_query as (
select
    query.*,
    "listing_data"."country" "country_query",
    "listing_data"."province" "province_query",
    "listing_data"."city" "city_query",

```

```

        "listing_data"."district" "district_query",
        "listing_data"."buildingArea" "buildingArea_query",
        "listing_data"."landArea" "landArea_query",
        "listing_data"."buildingFace" "buildingFace_query",
        "listing_data"."sellOrRent" "sellOrRent_query",
        "listing_data"."listingType" "listingType_query",
        "listing_data"."listingPartId" "listingPartId_query",
        "listing_data"."listingPropertyTypeId"
"listingPropertyTypeId_query",
        "listing_data"."listingCertificateTypeId"
"listingCertificateTypeId_query",
        "listing_data"."embedding_sellOrRent" "embedding_sellOrRent_query",
        "listing_data"."embedding_listingPart"
"embedding_listingPart_query",
        "listing_data"."embedding_listingPropertyType"
"embedding_listingPropertyType_query",
        "listing_data"."embedding_listingCertificateType"
"embedding_listingCertificateType_query",
        "listing_data"."embedding" "embedding_query"
from
    listing_data,
    query
where
    "listing_data"."listingId" = "query"."listingId"
),
first_pass as (
select
    "ld_query"."listingId" as "listingId_query",
    "ld" ."listingId",
    "ld_query"."country_query",
    "ld"."country",
    "ld_query"."province_query",
    "ld"."province",
    "ld_query"."city_query",

```

```

"ld"."city",
"ld_query"."district_query",
"ld"."district",
"ld_query"."buildingArea_query",
"ld"."buildingArea",
"ld_query"."landArea_query",
"ld"."landArea",
"ld_query"."buildingFace_query",
"ld"."buildingFace",
"ld_query"."sellOrRent_query",
"ld"."sellOrRent",
"ld_query"."listingType_query",
"ld"."listingType",
"ld_query"."listingPartId_query",
"ld"."listingPartId",
"ld_query"."listingPropertyTypeId_query",
"ld"."listingPropertyTypeId",
"ld_query"."listingCertificateTypeId_query",
"ld"."listingCertificateTypeId",
"ld_query"."embedding_sellOrRent_query",
"ld"."embedding_sellOrRent",
"ld_query"."embedding_listingPart_query",
"ld"."embedding_listingPart",
"ld_query"."embedding_listingPropertyType_query",
"ld"."embedding_listingPropertyType",
"ld_query"."embedding_listingCertificateType_query",
"ld"."embedding_listingCertificateType",
"ld_query"."embedding_query",
"ld"."embedding",
-- calculate the term frequency of keywords in the document
  coalesce(ts_rank(to_tsvector("ld"."title"),
    "ld_query"."title_query"),
    0) as title_frequency,
SIMILARITY(coalesce("ld_query"."title",

```

```

    ''),
    coalesce("ld"."title",
    '')) as title_similarity,
    coalesce(ts_rank(to_tsvector("ld"."descriptionPurified"),
    "ld_query"."description_query"),
    0) as description_frequency,
    SIMILARITY(coalesce("ld_query"."descriptionPurified",
    ''),
    coalesce("ld"."descriptionPurified",
    '')) as description_similarity,
    coalesce(ts_rank(to_tsvector("ld"."keyword"),
    "ld_query"."keyword_query"),
    0) as keyword_frequency,
    SIMILARITY(coalesce("ld_query"."keyword",
    ''),
    coalesce("ld"."keyword",
    '')) as keyword_similarity,
    (SIMILARITY("ld_query"."country_query",
    "ld"."country")
    + 2 * SIMILARITY("ld_query"."province_query",
    "ld"."province")
    + 3 * SIMILARITY("ld_query"."city_query",
    "ld"."city")
    + 4 * SIMILARITY("ld_query"."district_query",
    "ld"."district"))/ 10.0 as location_similarity,
    pgml.cosine_similarity("ld"."embedding_sellOrRent",
    "ld_query"."embedding_sellOrRent_query") as
    "sellOrRent_similarity",
    pgml.cosine_similarity("ld"."embedding_listingPart",
    "ld_query"."embedding_listingPart_query") as
    "listingPart_similarity",
    pgml.cosine_similarity("ld"."embedding_listingPropertyType",
    "ld_query"."embedding_listingPropertyType_query") as
    "listingPropertyType_similarity",

```

```

pgml.cosine_similarity("ld"."embedding_listingCertificateType",
  "ld_query"."embedding_listingCertificateType_query") as
"listingCertificateType_similarity",
pgml.cosine_similarity("ld"."embedding",
  "ld_query"."embedding_query") as feature_similarity
-- our basic corpus is stored in the documents table
from
  listing_data ld,
  listing_data_query ld_query
-- ranked by term frequency
-- ORDER BY title_frequency, keyword_frequency,
description_frequency DESC
-- prune to a reasonably large candidate population
-- LIMIT 10000
),
second_pass as(
select
  -- create a second pass score of cosine_similarity across
embeddings
    ( ${titleSimilarityWeight} * "first_pass"."title_similarity"
    + ${descriptionFrequencyWeight} *
"first_pass"."description_frequency"
    + ${descriptionSimilarityWeight} *
"first_pass"."description_similarity"
    + ${keywordFrequencyWeight} * "first_pass"."keyword_frequency"
    + ${keywordSimilarityWeight} *
"first_pass"."keyword_similarity"
    + ${locationSimilarityWeight} *
"first_pass"."location_similarity"
    + ${sellOrRentSimilarityWeight} *
"first_pass"."sellOrRent_similarity"
    + ${listingPartSimilarityWeight} *
"first_pass"."listingPart_similarity"

```



```

        + ${listingPropertyTypeSimilarityWeight} *
"first_pass"."listingPropertyType_similarity"
        + ${listingCertificateTypeSimilarityWeight} *
"first_pass"."listingCertificateType_similarity"
        + ${featureSimilarityWeight} *
"first_pass"."feature_similarity")
    /( ${titleSimilarityWeight}
    + ${descriptionFrequencyWeight}
    + ${descriptionSimilarityWeight}
    + ${keywordFrequencyWeight}
    + ${keywordSimilarityWeight}
    + ${locationSimilarityWeight}
    + ${sellOrRentSimilarityWeight}
    + ${listingPartSimilarityWeight}
    + ${listingPropertyTypeSimilarityWeight}
    + ${listingCertificateTypeSimilarityWeight}
    + ${featureSimilarityWeight} ) as similarity,
    *
from
    first_pass
--order by
    --    similarity desc

)
update
    "public"."listing_similarities_mat"
set
    score = "second_pass"."similarity",
    stale = false
from
    second_pass
where
    "listing_similarities_mat"."listing1Id" = ${listingIdParamName}

```

```
    and "listing_similarities_mat"."listing2Id" =  
"second_pass"."listingId"  
    and "listing_similarities_mat"."stale" = true  
returning "listing_similarities_mat" .*;
```

LAMPIRAN B

Query collaborative filtering

```
with similar_visitor as (  
    select  
        "candidate"."listingVisitorId",  
        count("candidate"."listingId") as same_listing_count,  
        tanh(sqrt(sum(power("candidate"."visitorRating"-  
"target"."visitorRating", 2)))/ count("candidate"."listingId"))/ 2 +  
0.5 as deviation  
    from  
        "public"."listing_visitor_ratings" as target  
    join "public"."listing_visitor_ratings" as candidate  
        on  
            "target"."listingVisitorId" != "candidate"."listingVisitorId"  
        and "target"."listingId" = "candidate"."listingId"  
    where  
        "target"."listingVisitorId" = '${visitorId}'  
    group by  
        "candidate"."listingVisitorId"  
    )  
select  
    "candidate"."listingId",  
    tanh(avg(1 / "similar_visitor"."deviation")*  
sum("candidate"."visitorRating"))/ 2 + 0.5 as total_rank  
from  
    "similar_visitor"  
join "public"."listing_visitor_ratings" as candidate  
    on  
        "candidate"."listingVisitorId" =  
"similar_visitor"."listingVisitorId"  
left join "public"."listing_visitor_ratings" as target  
    on
```

```
    "target"."listingVisitorId" = '${visitorId}'  
    and "target"."listingId" = "candidate"."listingId"  
where  
    "target"."listingId" is null  
group by  
    "candidate"."listingId"  
order by  
    total_rank desc  
limit ${Math.round(limit * 10)};
```

LAMPIRAN C

Query listing populer

```
select
    lvr."listingId",
    count("lvr"."listingVisitorId") as popularity_rank,
    avg("lvr"."visitorRating") as rating_rank,
    sum("lvr"."visitorRating") as combined_rank
from
    listing_visitor_ratings lvr
group by
    lvr."listingId"
order by popularity_rank desc, rating_rank desc, combined_rank
desc
limit ${Math.round(limit * 10)};
```

LAMPIRAN D

Query rekomendasi hybrid

```
with similar_visitor as (  
    select  
        "candidate"."listingVisitorId",  
        count("candidate"."listingId") as same_listing_count,  
        tanh(sqrt(sum(power("candidate"."visitorRating"-  
"target"."visitorRating", 2)))/ count("candidate"."listingId"))/ 2 +  
0.5 as deviation  
    from  
        "public"."listing_visitor_ratings" as target  
    join "public"."listing_visitor_ratings" as candidate  
        on  
            "target"."listingVisitorId" != "candidate"."listingVisitorId"  
        and "target"."listingId" = "candidate"."listingId"  
    where  
        "target"."listingVisitorId" = '${visitorId}'  
    group by  
        "candidate"."listingVisitorId"  
),  
similar_user_listing as (  
    select  
        "candidate"."listingId",  
        tanh(avg(1 / "similar_visitor"."deviation")*  
sum("candidate"."visitorRating"))/ 2 + 0.5 as colaborative_rank  
    from  
        "similar_visitor"  
    join "public"."listing_visitor_ratings" as candidate  
        on  
            "candidate"."listingVisitorId" =  
"similar_visitor"."listingVisitorId"  
    left join "public"."listing_visitor_ratings" as target
```

```

        on
        "target"."listingVisitorId" = '${visitorId}'
        and "target"."listingId" = "candidate"."listingId"
    where
        "target"."listingId" is null
    group by
        "candidate"."listingId"
    ),
    similar_listing as (
    select
        "ls"."listing2Id" as "listingId",
        "ls"."score" as content_rank
    from
        "listing_similarity" ls
    where
        "ls"."listing1Id" = '${referenceListingId}'
        and "ls"."listing2Id" != '${referenceListingId}'
    order by content_rank desc
    limit ${Math.round(limit * 2)}
    )
    select
        "sl"."listingId",
        (${this.hybridContentWeight} * coalesce("sl"."content_rank",
0) +
        ${this.hybridColaborativeWeight} *
coalesce("sul"."colaborative_rank", 0)
        ) as total_rank
    from
        "similar_listing" as "sl"
    left join "similar_user_listing" "sul"
        on "sul"."listingId" = "sl"."listingId"
    order by
        total_rank desc
    limit ${Math.round(limit * 2)};

```

