PROPOSAL TUGAS AKHIR

Software Development

Analisis Perbandingan Performa TRPC dengan REST API pada Implementasi Sistem *E-Commerce* Berbasis Web

Diajukan untuk Memenuhi Salah Satu Persyaratan Mencapai Derajat Sarjana Komputer

Angello Khara Sitanggang

220711833

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ATMA JAYA YOGYAKARTA

2025

LEMBAR PENGESAHAN

Analisis Perbandingan Performa Connect-RPC dengan REST API pada Implementasi Sistem Informasi Akademik Mahasiswa Berbasis Web

Yogyakarta, 20 November 2025

Angello Khara Sitanggang

220711833

Menyetujui,

Dosen Pembimbing

Prof. Dr. Andi Wahju R. Emanuel, B.S.E.E., M.S.S.E.

PERNYATAAN PERSETUJUAN DARI INSTANSI ASAL PENELITIAN

Saya yang bertanda tangan di bawah ini:

Nama Lengkap Pembimbing : Prof. Dr. Andi Wahju R. Emanuel
Jabatan : DPM
Departemen : Informatika

Menyatakan dengan ini:

Nama Lengkap : Angello Khara Sitanggang
NPM : 220711833
Program Studi : Informatika
Fakultas : Teknologi Industri
Judul Penelitian : Analisis Perbandingan Performa TRPC dengan REST API pada Implementasi Sistem E-Commerce Berbasis Web

1. Topik penelitian telah disetujui oleh pihak perusahaan.
2. Perusahaan telah melakukan sidang internal berupa kelayakan penelitian ini dan akan mencantumkan lembar penilaian secara tertutup kepada pihak universitas sebagai bagian dari nilai akhir mahasiswa.
3. Memberikan persetujuan kepada pihak Universitas Atma Jaya Yogyakarta untuk menyimpan, mengelola, mendistribusikan, dan menampilkan hasil penelitian dalam bentuk produk hasil penelitian meliputi dokumen, perangkat lunak, dan data hasil penelitian selama tetap mencantumkan nama penulis.

Demikianlah pernyataan ini dibuat dan dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 20 November 2025

Yang menyatakan,

Prof. Andi Wahju R Emanuel
B.S.E.E., M.S.S.E.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

ABSTRAK

Analisis Perbandingan Performa TRPC dengan REST API pada Implementasi Sistem *E-Commerce* Berbasis Web

Angello Khara Sitanggang

220711833

Pertumbuhan pasar *e-commerce* global yang diperkirakan mencapai USD 70,9 triliun pada 2028 dengan tingkat pertumbuhan 27,5% mendorong kebutuhan optimasi arsitektur API untuk menangani volume transaksi dan data pelanggan yang kompleks. Meskipun REST API tetap dominan, tRPC menawarkan alternatif dengan end-to-end type safety melalui TypeScript namun masih minim bukti empiris performanya. Penelitian ini mengidentifikasi kekosongan data kuantitatif mengenai perbandingan tRPC dan REST dalam konteks *e-commerce*, sehingga perlu dilakukan evaluasi empiris untuk menentukan arsitektur API yang paling optimal bagi pengembangan sistem *e-commerce* modern.

Penelitian menggunakan pendekatan prototyping dengan eksperimen terkontrol untuk membangun dua implementasi sistem *e-commerce* yang identik secara fungsional satu menggunakan REST API dan satu menggunakan tRPC berbasis TypeScript. Kedua sistem mencakup modul *e-commerce* seperti katalog produk, keranjang belanja, dan manajemen pesanan. Pengujian performa dilakukan dengan Apache JMeter melalui skenario browse, shopping, dan checkout pada berbagai tingkat beban. Metrik yang diukur mencakup response time, latency (p95 dan p99), throughput, error rate, concurrency handling, dan resource utilization (CPU, memori, jaringan). Pengujian mencakup load testing, stress testing, soak testing, dan spike testing untuk memberikan gambaran menyeluruh mengenai perilaku sistem dalam kondisi beban berbeda.

Penelitian diharapkan menghasilkan data empiris kuantitatif yang membandingkan performa REST API dan tRPC pada sistem *e-commerce*, mencakup identifikasi kondisi operasional di mana tRPC menunjukkan efisiensi lebih tinggi dibandingkan REST. Analisis statistik yang diharapkan akan menentukan relevansi perbedaan performa secara praktis. Hasil penelitian diharapkan menjadi referensi akademis bagi penelitian lanjutan dalam optimasi arsitektur API, sekaligus memberikan panduan empiris bagi pengembang dan pengambil keputusan teknis dalam memilih arsitektur yang sesuai kebutuhan skalabilitas dan efisiensi pengembangan sistem *e-commerce* berbasis web.

Kata Kunci: tRPC, REST API, *e-commerce*, performa API, performance testing, comparative analysis, TypeScript

BAB I
PENDAHULUAN

1.1. Latar Belakang

Perkembangan teknologi informasi dalam industri perdagangan elektronik telah mendorong sistem *e-commerce* menjadi salah satu komponen penting yang mendukung pengelolaan transaksi serta data pelanggan secara keseluruhan. Sistem *e-commerce* tidak hanya berfungsi sebagai media pencatatan transaksi pengguna, tetapi juga sebagai dasar digital untuk mengelola inventori, administrasi pemesanan, kegiatan pembayaran, hingga analisis data pelanggan dan penjualan. Dengan demikian, sistem ini mencakup keseluruhan proses perdagangan elektronik berbasis web.[1].

Laporan Grand View Research memperkirakan pasar global *e-commerce* akan meningkat dari USD 16,6 triliun pada 2023 menjadi USD 70,9 triliun pada 2028 dengan tingkat pertumbuhan tahunan majemuk (CAGR) sebesar 27,5% [2]. Perkembangan ini menandakan adanya transformasi digital dalam perdagangan skala besar yang menuntut pengelolaan data pelanggan semakin kompleks. Oleh karena itu, sistem *e-commerce* berperan sebagai alat pengelola data pelanggan yang menyimpan berbagai informasi seperti riwayat pembelian, preferensi produk, dan transaksi keuangan secara sistematis agar mudah diakses kembali

Pada platform *e-commerce*, sistem diperlukan untuk mampu melakukan proses pengambilan data secara cepat, menjaga kualitas dan konsistensi informasi, serta menjamin ketepatan data untuk mendukung aktivitas transaksi dan administrasi harian. Penelitian pada *e-commerce* kemudian berfokus pada peran arsitektur API yang memungkinkan integrasi sistem backend dan pihak ketiga secara efisien untuk sinkronisasi data real-time, otomatisasi proses bisnis, pengurangan kesalahan, serta peningkatan skalabilitas dan pengalaman pelanggan, yang menjadi faktor kunci dalam mendukung kelancaran operasional dan daya saing platform *e-commerce* modern [3].

Sejalan dengan hal tersebut, penelitian terbaru menunjukkan bahwa tidak

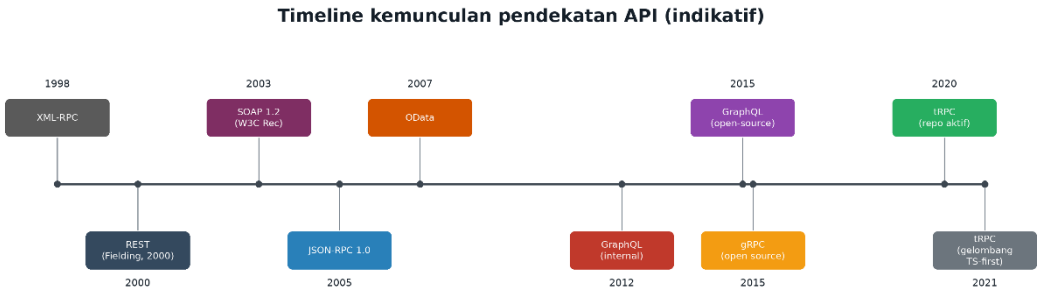
terdapat perbedaan signifikan secara statistik ($p > 0.05$) antara REST API yang menggunakan HTTP/1.1 dan gRPC API yang menggunakan HTTP/2 dalam metrik throughput, latency, dan response time. Analisis deskriptif mengungkapkan bahwa REST API memiliki kecenderungan throughput lebih tinggi pada kondisi low traffic serta respons time lebih stabil pada permintaan GET sederhana, yang disebabkan oleh overhead atau jumlah informasi tambahan dalam komunikasi yang lebih rendah dengan format teks JSON dan HTTP/1.1. [4].

Meski gRPC menunjukkan performa tinggi untuk komunikasi mikroservis, keterbatasannya pada browser menjadikannya kurang relevan untuk aplikasi *e-commerce* berbasis web [5], [6]. Di sisi lain, tRPC yang kompatibel dengan HTTP/1.1 dan berbasis TypeScript lebih sesuai untuk konteks tersebut, namun masih minim bukti empiris performanya dibandingkan REST API yang lebih mapan.

Maka dari itu, membandingkan tRPC dengan REST menjadi lebih relevan karena keduanya berbagi model komunikasi berbasis HTTP/1.1 dan JSON, yang sering digunakan pada aplikasi web modern. tRPC menawarkan keuntungan dari segi keamanan tipe dan efisiensi data dibandingkan REST, sehingga perbandingan ini penting untuk menentukan solusi terbaik sesuai kebutuhan pengembangan dan performa aplikasi.

Penelitian terhadap berbagai framework RPC, termasuk tRPC dan lainnya, menjadi sangat penting terutama dalam konteks aplikasi *e-commerce* yang menuntut interaksi real-time, skalabilitas tinggi, dan efisiensi komunikasi antara front-end dan back-end. Secara teori, protokol RPC seperti tRPC biasanya memberikan performa lebih baik dibandingkan REST karena RPC fokus pada pemanggilan fungsi langsung dengan overhead yang lebih rendah dan payload yang lebih kecil, sementara REST menggunakan pendekatan resource-oriented yang cenderung lebih verbose dan memiliki overhead lebih tinggi dalam komunikasi HTTP. Hal ini membuat RPC lebih efisien untuk operasi yang intensif dan komunikasi cepat antar komponen dalam sistem . Selain itu, kebutuhan e-commerce yang menuntut respons cepat dan stabil membuat pengujian langsung (misalnya mengukur latency dan throughput) menjadi penting untuk memastikan apakah

tRPC benar-benar lebih efisien daripada REST pada kondisi penggunaan nyata. Dengan begitu, hasil penelitian dapat menunjukkan kapan tRPC lebih tepat digunakan, bukan sekadar dinilai unggul secara teori.



Gambar 1. 1. Timeline Kemunculan Pendekatan API

.

Maka dari itu, justifikasi dan relevansi penelitian ini dapat dirumuskan secara lebih tegas. Pertama, meskipun REST masih menjadi arsitektur API yang paling banyak digunakan [7], sudah mulai banyak alternatif lain, salah satunya tRPC hal ini menunjukkan adanya pendekatan baru yang layak untuk diteliti lebih lanjut. tRPC menawarkan integrasi end-to-end dengan pemanfaatan TypeScript yang dapat memberikan efisiensi pada beberapa aspek pengembangan, namun sejauh mana hal tersebut berdampak terhadap performa sistem masih perlu diteliti secara empiris. Maka dari itu penelitian ini memilih REST sebagai baseline karena REST merupakan pendekatan yang paling mapan dan menjadi fondasi praktik API web modern sejak diperkenalkan sebagai gaya arsitektur pada 2000. Selain itu, berbagai pendekatan lain seperti GraphQL (dikembangkan internal sejak 2012 dan spesifikasinya di-open-source pada 2015) menunjukkan adanya evolusi kebutuhan, terutama pada sisi client-driven data fetching. Namun, kemunculan pendekatan yang lebih baru seperti tRPC yang menekankan end-to-end type safety berbasis TypeScript muncul di tahun 2021, hal ini mendorong kebutuhan penelitian empiris untuk menguji dampaknya terhadap performa sistem jika dibandingkan dengan

pendekatan yang lebih fundamental dan mapan seperti REST. Oleh karena itu, penelitian ini memposisikan tRPC bukan sebagai pengganti REST, melainkan sebagai alternatif pada konteks tertentu, sehingga perbandingan dengan REST sebagai dasar menjadi relevan dan metodologis.

Kedua, terbatasnya kajian akademik yang secara khusus membahas perbandingan antara tRPC dan REST API pada konteks *e-commerce*. Dengan meningkatnya kebutuhan integrasi real-time dan efisiensi komunikasi dalam platform *e-commerce* modern, diperlukan penelitian baru terhadap protokol alternatif seperti tRPC untuk memberikan gambaran empiris tambahan mengenai efektivitas dan potensi alternatif protokol lain dalam mendukung kebutuhan komunikasi data. Dengan demikian, penelitian ini diharapkan dapat memberikan kontribusi dalam memperkaya literatur terkait evaluasi performa API modern serta mendukung pemahaman yang lebih luas mengenai penerapan teknologi komunikasi data di lingkungan pengembangan web.

Sebagai tindak lanjut atas kondisi tersebut, penelitian ini difokuskan pada evaluasi performa tRPC dibandingkan dengan REST API dalam penerapan pada sistem *e-commerce* berbasis web. Mengingat REST masih menjadi standar dominan sementara tRPC menawarkan pendekatan alternatif yang menjanjikan efisiensi dan *type safety*, penelitian ini bertujuan untuk menilai sejauh mana tRPC dapat berperan sebagai solusi efektif dalam pengembangan sistem *e-commerce* modern. Hasil penelitian diharapkan mampu memberikan gambaran empiris bagi pengembang dan peneliti dalam menentukan teknologi API yang paling sesuai dengan kebutuhan serta karakteristik sistem di era digitalisasi perdagangan.

1.2. Rumusan Masalah

Pertumbuhan volume data di platform *e-commerce* menambah tekanan pada lapisan API, memicu fluktuasi waktu respons dan peningkatan error saat beban tinggi. REST API masih banyak digunakan karena kompatibilitasnya yang luas, tetapi kurang efisien untuk trafik besar. Sebaliknya, tRPC menawarkan integrasi TypeScript end-to-end yang lebih ringan, meski belum terbukti performanya dalam konteks *e-commerce*. Kekosongan data empiris ini menimbulkan kesenjangan atau

gap nyata untuk menentukan pilihan teknologi yang tepat. Penelitian ini difokuskan pada pengujian kedua API dalam skenario transaksi *e-commerce* berbasis web dengan konfigurasi server identik, menggunakan metrik seperti waktu respons, *latency*, *throughput*, *concurrency*, *error rate*, dan efisiensi sumber daya sebagai dasar evaluasi performa.

1.3. Batasan Penelitian

1. Penelitian tidak mencakup teknologi API lain seperti GraphQL atau gRPC.
2. Pengujian performa tidak mencakup aspek non-fungsional lain seperti keamanan, autentikasi, dan validasi data secara mendalam karena berada di luar ruang lingkup analisis performa inti.
3. Evaluasi dilakukan dalam lingkungan uji simulasi, sehingga penelitian ini tidak mempertimbangkan faktor eksternal pada sistem produksi nyata seperti variasi trafik pengguna atau kondisi jaringan tidak stabil.
4. Aspek skalabilitas lintas server dan integrasi dengan layanan pihak ketiga tidak dibahas dalam penelitian ini, mengingat fokus utama diarahkan pada pengukuran performa dasar kedua API dalam kondisi lingkungan terkendali.

1.4. Pertanyaan Penelitian

1. Bagaimana hasil perbandingan empiris kinerja antara tRPC dan REST API pada sistem e-commerce berbasis web berdasarkan indikator performa yang diukur?
2. Dalam kondisi beban atau skenario apa tRPC menunjukkan performa yang lebih optimal dibandingkan REST API dalam pengelolaan transaksi dan data pelanggan?
3. Berdasarkan hasil pengujian kinerja serta pengalaman implementasi pengembang, arsitektur API mana yang lebih tepat direkomendasikan untuk meningkatkan skalabilitas dan efisiensi sistem e-commerce modern?

1.5. Tujuan Penelitian

1. Melakukan analisis komparatif terhadap performa REST API dan tRPC pada sistem e-commerce berbasis web berdasarkan metrik *response time*, *throughput*, *latency (p95/p99)*, dan *resource utilization*. Melalui testing performa dengan metode *load, stress, spike,* dan *soak*
2. Mengidentifikasi kondisi operasional di mana tRPC menunjukkan efisiensi lebih tinggi dibandingkan REST API, baik dari aspek performa maupun produktivitas pengembang.
3. Menyediakan data empiris yang dapat menjadi dasar pertimbangan dalam pemilihan arsitektur API untuk pengembangan sistem *e-commerce* modern yang menuntut efisiensi dan skalabilitas tinggi.

1.6. Manfaat Penelitian

1) Manfaat Teoretis

1. Menambah literatur akademis dalam bidang rekayasa perangkat lunak dan arsitektur API, khususnya mengenai teknologi tRPC yang masih relatif baru.
2. Menjadi referensi awal dalam studi perbandingan performa API pada sistem *e-commerce* berbasis web, yang dapat dijadikan landasan penelitian lanjutan di bidang optimasi arsitektur API dan developer experience.

2) Manfaat Praktis

1. Memberikan panduan empiris bagi pengembang dan akademisi dalam memilih arsitektur API yang tepat sesuai kebutuhan performa dan efisiensi pengembangan.
2. Membantu industri *e-commerce* mengidentifikasi pendekatan API yang dapat meningkatkan kecepatan, reliabilitas, dan skalabilitas sistem.
3. Menjadi dasar bagi pengambil keputusan teknis dalam menentukan strategi migrasi arsitektur dari REST API menuju tRPC atau protokol modern lainnya.

BAB II

TINJAUAN PUSTAKA

2.1. Tinjauan Pustaka

2.1.1. Penelitian REST API

Berbagai penelitian terdahulu telah membahas performa dan efisiensi komunikasi antar mikroservis melalui pendekatan API. Niswar et al [8] menemukan bahwa gRPC memiliki waktu respons tercepat / terendah karena dukungan HTTP/2, sementara REST berada di posisi menengah dan GraphQL cenderung lebih lambat untuk data kompleks maupun sederhana. Namun, studi tersebut belum menilai *throughput*, tingkat kegagalan, atau konsumsi memori, sehingga belum memberikan gambaran menyeluruh tentang efisiensi sistem. Penelitian lain oleh Muzaki dan Salam [9] menyoroti masalah *over-fetching* dan *under-fetching* pada REST API, dan merekomendasikan GraphQL sebagai solusi untuk mengoptimalkan pengambilan data. Sementara itu, Brito dan Valente [10] menunjukkan bahwa GraphQL memiliki performa lebih baik daripada REST bahkan di kalangan peserta penelitian yang lebih berpengalaman dan di kalangan peserta yang memiliki pengalaman sebelumnya dengan REST, tetapi belum memiliki pengalaman dengan GraphQL. GraphQL membutuhkan usaha yang lebih sedikit untuk membuat kueri ke layanan jarak jauh dibandingkan. Keuntungan ini menjadi semakin besar ketika kueri REST harus berinteraksi dengan endpoint yang lebih kompleks dan melibatkan banyak parameter.

Narayanan et al [11] dan Quin et al. [12] menekankan pentingnya optimasi dan evaluasi berbasis metrik. Narayanan et al secara spesifik mengusulkan strategi sadar-struktur-halaman pada CDN untuk mengurangi latensi dengan memprioritaskan objek web penting. Di sisi lain, Quin et al, melalui tinjauan literatur sistematisnya, menegaskan bahwa A/B testing sangat penting untuk mengevaluasi dampak algoritma dan elemen visual terhadap metrik bisnis. Sementara itu, Addanki [13] serta Microsoft [6] menggambarkan pergeseran evolusi API dari SOAP ke REST, lalu menuju GraphQL dan gRPC, dengan fokus berbeda antara produktivitas developer dan efisiensi sistem .

Dengan demikian, meskipun berbagai studi seperti Niswar et al. [8], Muzaki dan Salam [9]serta Brito dan Valente [10] menunjukkan bahwa performa

API bergantung pada kecepatan respons, efisiensi pengambilan data, dan desain caching, namun masing-masing menonjolkan aspek berbeda tanpa menyajikan gambaran menyeluruh. Narayanan et al. [11] dan Quin et al. [12] menegaskan pentingnya evaluasi berbasis metrik nyata untuk menilai dampak performa terhadap hasil bisnis, sementara Addanki [13] dan Microsoft [6] menyoroti pergeseran paradigma API menuju efisiensi dan kemudahan integrasi. Dengan demikian, masih terdapat celah penelitian dalam menguji tRPC sebagai pendekatan end-to-end modern dibanding REST, khususnya pada sistem *e-commerce* dengan tuntutan performa tinggi dan efisiensi sumber daya terutama dengan minimnya data empiris mengenai tRPC dalam penelitian yang sudah ada.

2.1.2. Penelitian tRPC

tRPC unggul pada *end-to-end type safety* dengan deteksi error di *compile-time* dan eliminasi sinkronisasi tipe klien-server, tetapi bukti yang tersedia belum bersifat empiris karena belum ada evaluasi empiris kinerja yang mengaitkan keunggulan tersebut dengan metrik sistemik seperti *latency, throughput, error rate*, serta penggunaan CPU/memori pada beban e-commerce berbasis microservices; oleh karena itu, riset ini penting untuk menutup kekosongan dengan eksperimen terkontrol yang membandingkan tRPC terhadap REST/gRPC pada workload *e-commerce* representatif guna memverifikasi dampak nyata type safety terhadap performa dan stabilitas produksi [4] .

2.1.3. Penelitian E-Commerce

Studi komparatif REST–GraphQL–gRPC oleh Chandra dan Farisi [14] yang dipublikasikan oleh Atmaluhur melaporkan adanya *trade-off* dimana REST mencapai permintaan tinggi tinggi namun dengan konsumsi sumber daya dan tingkat kegagalan yang relatif lebih tinggi GraphQL lebih efisien dalam pemakaian sumber daya dengan hasil yang paling stabil sedangkan gRPC cenderung unggul pada pengunaan CPU terendah walaupun menunjukan spike beberapa kali hal ini berkat HTTP/2 dan Protobuf. Berdasarkan penelitian Hasil penelitian menunjukkan bahwa RESTful unggul dalam jumlah permintaan tetapi memiliki tingkat

keberhasilan lebih rendah, sementara GraphQL lebih efisien dan gRPC menyeimbangkan performa serta sumber daya meski memorinya meningkat di bawah tekanan. Namun, alur *e-commerce end-to-end* belum diuji secara khusus dalam pengukuran ini. Di luar itu, kajian perbandingan protokol seperti studi oleh Ain et al. [4] yang membandingkan gRPC–REST dalam layanan terdistribusi menyediakan bukti performa protokol, tetapi tidak mengaitkannya secara eksplisit dengan skenario *e-commerce*, sehingga memunculkan celah penelitian yang jelas di sini

Literatur *e-commerce* yang kuat justru berada pada sisi perilaku dan pengalaman pengguna, bukan perbandingan protokol API. Sebuah SLR(*systematic literature review*) tentang social commerce [15] memetakan pendorong interaksi sosial dan dampaknya terhadap perilaku belanja, tanpa menguji lapisan API atau performa teknis transaksi. Demikian pula, SLR mengenai *impulse buying* di *e-commerce* [16] menyoroti pemicu pembelian impulsif pada format live shopping, tetapi tidak melakukan pengukuran komparatif performa komunikasi layanan dalam funnel transaksi. Tinjauan tren *e-commerce* 2018–2024 juga membahas teknologi, logistik, dan keamanan, namun tidak menyajikan benchmark API pada skenario *browse* lalu *cart* lalu *checkout* yang dapat dihubungkan langsung ke metrik *latensi/throughput* transaksi.

Menurut studi Chandra dan Farisi [14],ada trade-off REST–GraphQL–gRPC yang relevan secara umum, tetapi belum teruji pada alur *e-commerce end-to-end*. Sementara itu, SLR *e-commerce* terbaru menekankan pada perilaku (*social/impulse buying*) dan tidak menyentuh perbandingan performa protokol API, sehingga tidak dapat dibandingkan langsung dengan studi protokol. Dengan demikian, hingga saat ini belum ditemukan penelitian akademik yang secara eksplisit dan kuantitatif membandingkan tRPC dengan REST di konteks *e-commerce*. Ini membuka celah riset untuk merancang benchmark terkendali tRPC vs. REST pada skenario *e-commerce* dengan metrik yang sesuai untuk memberikan data empiris yang dapat dijadikan bahan pertimbangan.

2.2. Pengembangan Sistem *E-Commerce* Berbasis Web

Pengembangan *e-commerce* sebaiknya menggabungkan sisi teknis dan bisnis secara seimbang. Kolaborasi antara tim pengembang perangkat lunak dan pemangku kepentingan komersial penting agar setiap fitur yang dibuat benar-benar mendukung tujuan seperti peningkatan konversi, keamanan transaksi, dan kelancaran operasional.

Dalam pengembangan *e-commerce*, penerapan antarmuka layanan umumnya menggunakan REST dengan format JSON karena mudah diintegrasikan, berorientasi pada sumber daya, dan didukung oleh beragam alat pengembang [17]. Pada tingkat sistem yang lebih kompleks, pendekatan arsitektur microservices sering digunakan dengan komunikasi asinkron berbasis event atau message broker, seperti Apache Kafka. Pendekatan ini membantu sistem tetap stabil ketika terjadi lonjakan trafik, mengurangi ketergantungan antar layanan seperti pesanan, pembayaran, dan manajemen stok, serta meningkatkan ketahanan dan skalabilitas secara keseluruhan [18].

Untuk Penelitian dan Pengembangan frontend *e-commerce* diawali dengan penggunaan React dan TypeScript lalu implementasi tRPC dan REST. Kombinasi ini memberikan konsistensi tipe data, memudahkan pembuatan komponen yang dapat digunakan kembali, serta didukung oleh ekosistem yang matang untuk aplikasi seperti katalog produk, keranjang belanja, dan proses checkout. Berbagai studi teknis dan praktik industri juga menunjukkan bahwa React dan TypeScript menjadi pilihan utama untuk membangun antarmuka *e-commerce* modern. Untuk memastikan performa sistem, pengujian menggunakan Apache JMeter sangat dianjurkan. Alat ini membantu mengukur throughput dan waktu respons pada berbagai skenario nyata, seperti pemuatan katalog, tampilan detail produk, interaksi keranjang, dan proses checkout. Penelitian konfigurasi 20 pengguna virtual oleh Khlamov et al [19], Dilakukan dengan serangkaian pengujian performa pada lima API publik menggunakan dua alat dengan berbagai profil beban dan konfigurasi operasi CRUD. Setiap alat dijalankan dengan setting pengguna virtual dan waktu tunda untuk meniru kondisi dunia nyata secara lebih akurat. Hasil menunjukkan Postman lebih efisien pada beban rendah, sedangkan JMeter unggul dalam

menghadapi intensitas permintaan tinggi dan beban ekstrem. Pemilihan alat yang tepat dapat meningkatkan efektivitas pengujian API sesuai kebutuhan dan skala aplikasi.

2.3. Celah Penelitian

Dari kajian pustaka di atas, dapat diidentifikasi bahwa:

1) Studi komparatif terbaru berfokus pada REST, GraphQL, dan gRPC seperti Niswar et al. [8] yang menilai response time dan CPU usage, tetapi tidak mengukur *throughput*, memori, atau stabilitas di bawah beban, sehingga kesimpulannya belum menyentuh efisiensi sistem menyeluruh.
2) Bukti empiris tRPC di e-commerce masih minim, sehingga belum jelas kapan tRPC unggul dari REST pada pola trafik nyata (*browse–add-to-cart–checkout*) dan ukuran payload dengan HTTP/1.1 dan JSON.
3) Dokumentasi dan tulisan praktik tentang tRPC menekankan end-to-end type safety dan deteksi error di compile-time, namun belum ada evaluasi empiris terstandar pada workload e-commerce yang mengaitkan klaim tersebut dengan metrik, throughput, error rate, serta penggunaan CPU/memori[20].

Celah penelitian yang muncul adalah perlunya eksperimen terkontrol yang membandingkan tRPC dengan REST API pada aplikasi e-commerce web di lingkungan deployment identik, mengukur metrik seperti *response time* ,*throughput, error rate, concurrency handling*, serta penggunaan CPU/memori pada pola trafik nyata melalui testing performa dengan *load, spike, soak,* dan *stress* testing agar menghasilkan rekomendasi praktis pemilihan arsitektur API di pada e-commerce modern.

Tabel 2. 1. Tabel Perbandingan Penelitian

Penulis	**Konteks**	**API**	**Metrik**	**Temuan Utama**
Niswar et al. [8]	Komunikasi microservices (REST vs GraphQL vs gRPC)	REST, GraphQL, gRPC	*Response time, CPU usage*	Hasil evaluasi performa menunjukkan gRPC mencapai waktu respons lebih cepat dan efisiensi CPU lebih tinggi dibandingkan REST dan GraphQL. Keunggulan ini terutama berasal dari penggunaan protokol HTTP/2 yang lebih modern dan efisien dalam mengelola komunikasi data.
Ain et al. [4]	Performa REST vs gRPC pada variasi trafik & ukuran data	REST, gRPC	*Throughput, Latency, Response Time , received data/payload*	Hasil menunjukkan REST API memiliki throughput lebih tinggi pada beban rendah , dan response time lebih cepat untuk metode GET pada low traffic (1 ms vs 20 ms). Sebaliknya, gRPC unggul dalam efisiensi payload) dan latency yang lebih stabil, sementara analisis ANOVA menunjukkan tidak ada perbedaan signifikan secara statistik ($p > 0,05$).
Golmohammadi et al. [17]	Survei pengujian RESTful APIs dan tooling industri	REST (survei)	Status Publikasi dalam testing, Pendekatan testing otomatis, Tools Tersedia, Tantangan	Penelitian ini menemukan bahwa minat terhadap pengujian RESTful API meningkat pesat sejak 2017, dengan berbagai metode dan alat open-source yang telah diuji pada API nyata dan berhasil menemukan

Penulis	Konteks	API	Metrik	Temuan Utama
				kesalahan.
Oyeniran et al. [18]	Microservices cloud-native dan messaging async (Kafka)	Pola komunikasi async (broker)	Ketahanan, skalabilitas, decoupling, fleksibilitas	Penelitian ini menegaskan bahwa arsitektur microservices telah menjadi strategi utama dalam pengembangan aplikasi modern karena meningkatkan skalabilitas, fleksibilitas, dan ketahanan sistem. Broker seperti Kafka berperan dalam mendukung pola publish–subscribe dan antrian untuk mendekopel layanan, menstabilkan lonjakan trafik, serta meningkatkan ketahanan dan skalabilitas sistem.
Khlamov et al. [19]	Perbandingan alat uji API performance (JMeter vs Postman)	N/A (tooling)	Kapabilitas uji performa API	Keduanya mendukung uji performa Postman lebih efisien pada beban rendah, sedangkan JMeter unggul dalam menghadapi intensitas permintaan tinggi dan beban ekstrem.
Penelitian ini	tRPC vs REST di e-commerce (web) dengan modul inti	REST, tRPC	*Response Time, Throughput, Latency (p95/p99), Error Rate, Concurrency, Resource Utilization*	Eksperimen dirancang dengan kontrol ketat dalam lingkungan identik dengan baseline simetris untuk memastikan perbandingan adil. Skenario pengujian mencakup alur *e-commerce* lengkap (*browse–add-to-cart–checkout*) yang bertujuan mengisi celah penelitian mengenai perbandingan tRPC versus REST yang masih belum ada di literatur.

2.4. Landasan Teori

Landasan teori penelitian ini mengkaji perbandingan performa antara arsitektur REST API dan tRPC dalam konteks sistem *e-commerce* modern. Analisis ini didasarkan pada enam pilar utama: karakteristik fundamental REST API dan tRPC, tantangan pengembangan spesifik untuk *e-commerce*, arsitektur frontend modern, serta metodologi dan metrik kuantitatif yang digunakan untuk pengujian kinerja API. Kerangka kerja ini bertujuan memberikan konteks menyeluruh tentang relevansi perbandingan kedua pendekatan tersebut dalam pengambilan keputusan teknis untuk aplikasi yang menuntut respons cepat dan efisiensi data.

2.4.1. REST API

REST API menggunakan protokol HTTP dengan sifat stateless, sehingga setiap permintaan diproses secara independen tanpa menyimpan konteks sesi sebelumnya [13]. Data umumnya dikirim dalam format JSON karena sederhana dan mudah dibaca, namun sebagai format teks json membawa overhead lebih besar dibanding format biner sehingga dapat menambah latency saat beban atau ketika terjadi banyak permintaan berulang untuk operasi kompleks [21]. REST memanfaatkan metode standar HTTP seperti GET, POST, PUT, dan DELETE untuk melakukan operasi pada resource yang diidentifikasi melalui URL [13].

Dalam praktiknya, REST API berfungsi sebagai jembatan integrasi lintas platform dan bahasa, memungkinkan berbagai sistem berkomunikasi dan mengakses data yang sama secara terstandar . Pada domain kesehatan, FHIR (*Fast Healthcare Interoperability Resources*) secara eksplisit mendefinisikan model data umum dan arsitektur RESTful berbasis HTTP sehingga berbagai sistem dapat berbagi dan mengintegrasikan data dengan cara yang konsisten [22]. Implementasi nasional seperti SATUSEHAT di Indonesia juga menyediakan REST API FHIR untuk resource pasien dan modul terkait sebagai contoh penerapan interoperabilitas klinis [23].

Untuk keamanan, mekanisme OAuth 2.0 memungkinkan sistem memantau penggunaan aplikasi oleh konsumen dan melacak token akses yang digunakan untuk permintaan, sehingga memberi wawasan operasional dan keamanan yang lebih baik [24]. Praktik modern juga melibatkan *monitoring* dan *auditing* token berbasis telemetry untuk mendeteksi anomali akses dan penyalahgunaan secara *real-time* [25].

REST mendukung arsitektur yang fleksibel dan modular, cocok dengan pola microservices yang menekankan layanan mandiri dan *loose coupling*. Namun, saat klien ingin mengambil beberapa resource terkait sekaligus, REST sering mengalami masalah *overfetching* dan *underfetching* yang memperbesar ukuran *payload* dan waktu respons, sehingga pada skenario tertentu GraphQL bisa lebih efisien [21].

Skalabilitas REST sangat dipengaruhi oleh kemampuan server dalam menangani koneksi bersamaan, yang bergantung pada faktor seperti CPU, memori, *bandwidth*, serta efisiensi implementasi server dan pola akses klien. Pengukuran *throughput* dan *latency* di berbagai beban lalu lintas menjadi kunci utama dalam menilai kapasitas sistem [21]. Karena itu, perbandingan sistematis antara REST dan alternatif modern seperti tRPC penting untuk memilih arsitektur optimal sesuai kebutuhan aplikasi terutama bagi aplikasi *e-commerce* yang memerlukan respons cepat dan pembaruan data secara real-time.

2.4.2. tRPC

Remote Procedure Call (RPC) adalah pola komunikasi yang memungkinkan aplikasi memanggil fungsi di server seolah-olah fungsi tersebut lokal, sehingga interaksi terasa seperti pemanggilan prosedur biasa daripada manipulasi HTTP tingkat rendah [26] [27]. tRPC adalah implementasi RPC berfokus pada sistem TypeScript yang memungkinkan pengembangan API *end-to-end* dengan *type safety* tanpa perlu membuat skema atau melakukan generasi kode, sehingga pemanggilan fungsi server dari client mendapatkan tipe secara otomatis [28] [29]. TypeScript bertindak sebagai *static type checker* di waktu kompilasi, sehingga tipe input, output, dan error diverifikasi sebelum runtime dan mengurangi potensi bug tipe di

produksi [30].

Keunggulan utama tRPC adalah jaminan keamanan tipe data secara menyeluruh dari ujung ke ujung serta pengalaman pengembangan yang lebih baik. Dokumentasi resmi tRPC menekankan bahwa klien memperoleh *static type safety* penuh beserta *autocompletion* untuk *input, output*, dan *error*, sehingga alur pengembangan menjadi lebih cepat dan konsisten di seluruh *stack* TypeScript [20].

Untuk kebutuhan real-time dan skalabilitas, tRPC mendukung fitur subscriptions dan *WebSockets* yang memungkinkan server melakukan push pembaruan secara langsung ke klien. Berbeda dengan pola REST yang umumnya menggunakan mekanisme pull, *WebSockets* pada tRPC menyediakan koneksi persisten untuk komunikasi dua arah secara efisien, sehingga cocok untuk aplikasi yang membutuhkan pembaruan data real-time dan interaksi responsif [20]. tRPC juga menyediakan *middleware* dan penanganan error terintegrasi memudahkan penerapan kontrol akses, validasi, dan logging secara konsisten di seluruh prosedur yang relevan untuk aplikasi *e-commerce* yang membutuhkan performa tinggi dan pembaruan data real-time [20]. Namun pendekatan REST tetap bernilai karena sifatnya yang umum, dokumentasi luas, dan kompatibilitas yang tinggi .

2.4.3. Pengembangan Sistem Web untuk E-Commerce

Pengembangan sistem web untuk *e-commerce* membutuhkan pendekatan multidisiplin yang mencakup rekayasa perangkat lunak, desain antarmuka, serta keandalan platform dalam mengelola transaksi dan interaksi pengguna. Salah satu metode yang banyak digunakan adalah *User-Centered Design* (UCD), yaitu pendekatan perancangan yang menempatkan pengguna sebagai fokus utama di setiap tahapan pengembangan, mengawali dari penelitian awal hingga pengujian akhir, dengan tujuan memastikan bahwa produk yang dikembangkan sesuai dengan kebutuhan dan keinginan pengguna [35].

User-Centered Design merupakan proses iteratif yang menitikberatkan pada pengalaman dan kepuasan pengguna dengan melalui empat fase utama: mengidentifikasi lingkungan penggunaan, merumuskan kebutuhan pengguna dan organisasi, mengembangkan solusi desain, dan melakukan evaluasi desain sesuai

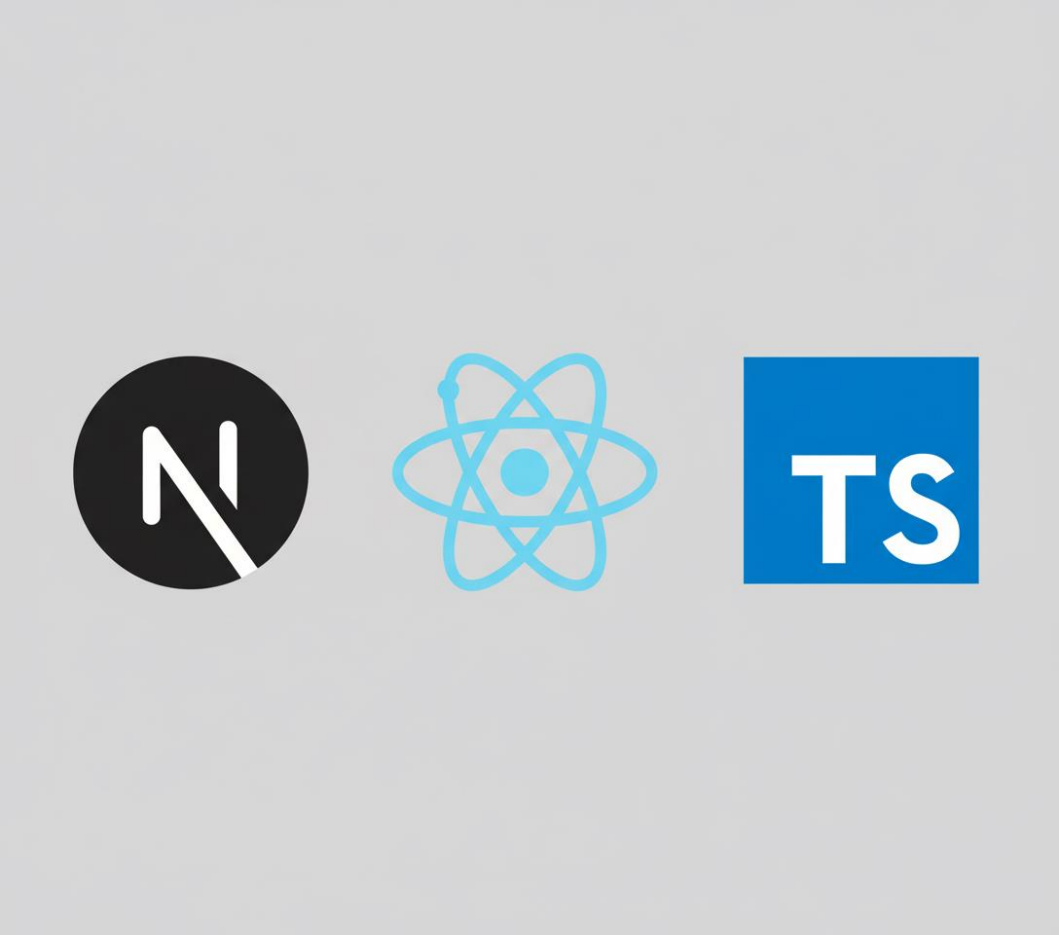
kebutuhan pengguna. [36]. Dengan pendekatan ini, aspek navigasi, penyampaian informasi, dan kualitas antarmuka dapat dioptimalkan agar sesuai dengan kebutuhan pengguna secara lebih efektif [35].

Secara teknis, kualitas perangkat lunak e-commerce perlu mengacu pada standar internasional seperti ISO/IEC 25010, yang menilai atribut penting seperti *Functional Suitability* (kecocokan fungsi), *Reliability* (keandalan), *Usability* (kegunaan), *Performance Efficiency* (efisiensi performa), dan *Security* (keamanan sistem) [37].

Dari sisi keamanan data, keamanan transaksi mencakup perlindungan informasi sensitif, transaksi keuangan, dan data pengguna melalui berbagai mekanisme, seperti enkripsi data dan penerapan kontrol akses yang ketat [38]. Oleh karena itu, peran ahli bisnis digital dan keamanan data sangat penting untuk menjaga kelancaran operasional serta membangun kepercayaan pengguna terhadap platform *e-commerce*.

2.4.4. Teknologi Frontend dan Arsitektur Aplikasi Web Modern

Pengembangan aplikasi *e-commerce* modern tidak terlepas dari arsitektur frontend yang digunakannya, karena sisi klien inilah yang bertanggung jawab untuk mengonsumsi API dan menampilkannya kepada pengguna, sehingga pilihan teknologi di lapisan ini berpengaruh langsung pada pengalaman dan kinerja sistem secara keseluruhan. Pilihan teknologi frontend secara langsung memengaruhi cara data diminta, diterima, dan dirender, yang pada akhirnya berdampak pada metrik performa yang dirasakan oleh pengguna seperti waktu muat awal, interaktivitas, dan respons terhadap permintaan data dinamis.



Gambar 2. 2. Teknologi Arsitektur Web

React adalah pustaka JavaScript untuk membangun antarmuka pengguna (UI) yang menyusun UI dari unit-unit kecil yang disebut komponen. Komponen dalam React merupakan potongan UI terisolasi yang dapat diawali dari elemen sederhana seperti tombol hingga satu halaman penuh, dan dapat dikombinasikan untuk membangun tampilan yang lebih kompleks. Pendekatan berbasis komponen ini mendukung pembuatan UI yang kompleks secara modular, di mana setiap komponen mengelola logika dan tampilannya sendiri sehingga dapat dikembangkan dan dipelihara tanpa memengaruhi komponen lain [39]. Praktik komponen yang dapat digunakan kembali ini banyak diterapkan dalam pengembangan *e-commerce*, misalnya untuk komponen kartu produk, daftar katalog, dan keranjang belanja, karena memudahkan konsistensi tampilan sekaligus

efisiensi pengembangan.

Next.js adalah framework React yang dirancang untuk penggunaan di produksi dan menyediakan fitur-fitur untuk membangun aplikasi web modern, termasuk pengambilan data dan strategi prerendering. Salah satu kemampuan utamanya adalah dukungan terhadap berbagai strategi rendering yang menentukan kapan dan di mana data dari API diambil dan diubah menjadi HTML sebelum dikirimkan ke pengguna [40].

TypeScript adalah *superset* JavaScript yang menambahkan sistem tipe statis, sehingga kode dapat dianalisis terlebih dahulu untuk mendeteksi kesalahan terkait tipe data sebelum dijalankan. Tipe statis ini memungkinkan pemeriksaan kesesuaian tipe pada saat kompilasi, sehingga banyak kesalahan yang biasanya baru muncul di runtime dapat ditemukan lebih awal dalam proses pengembangan [41].

Penggunaan TypeScript secara konsisten di frontend (React/Next.js) dan backend menciptakan dasar bagi pendekatan end-to-end type safety pada komunikasi antara klien dan server [41]. tRPC secara eksplisit memanfaatkan kemampuan TypeScript untuk membangun dan mengonsumsi API yang sepenuhnya bertipe tanpa memerlukan skema terpisah atau proses code generation, dengan membagikan definisi tipe yang sama antara sisi server dan klien. Dengan mekanisme ini, ketika skema atau tipe data di server berubah, pemanggilan API di sisi frontend yang tidak lagi sesuai akan terdeteksi sebagai kesalahan pada saat kompilasi atau proses build, yang sering kali langsung terlihat di editor melalui autocompletion dan pemeriksaan tipe. Pendekatan tersebut membantu menghilangkan kelas bug yang disebabkan oleh ketidaksesuaian kontrak API antara klien dan server sekaligus mengurangi kebutuhan dokumentasi manual, karena definisi tipe yang tertanam di dalam kode berfungsi sebagai dokumentasi yang selalu mutakhir dan konsisten di seluruh stack TypeScript [42].

2.4.5. Metrik Pengujian Kinerja Sistem

Pengujian kinerja umumnya didefinisikan sebagai proses mengevaluasi perangkat lunak ketika diberikan beban kerja tertentu untuk menilai stabilitas, kecepatan, dan efisiensi penggunaan sumber daya. Dalam penelitian backend API,

load testing sering digunakan untuk mensimulasikan beban normal atau beban yang direncanakan, dengan mengukur metrik seperti waktu respons, *throughput*, *latency*, *concurrency handling error rate* serta penggunaan CPU dan memori untuk menentukan kapasitas aman sistem sebelum performa mulai menurun [43].

Stress testing digunakan untuk mendorong sistem melampaui batas kapasitas normalnya dengan tujuan melihat bagaimana sistem mulai gagal dan bagaimana pola pemulihannya setelah berada pada kondisi ekstrem. Studi-studi kinerja backend menunjukkan bahwa kombinasi load dan stress testing memberikan gambaran dasar mengenai batas atas kemampuan layanan dan karakteristik skalabilitasnya di berbagai skenario beban [43].

Sebagai pelengkap, pengujian performa tidak hanya menggunakan load dan stress testing, tetapi juga *spike testing* untuk melihat respons sistem terhadap lonjakan beban yang sangat tiba-tiba, serta soak atau endurance testing untuk menilai kestabilan sistem pada beban yang relatif konstan dalam waktu lama. Kombinasi metode *load, stress, spike,* dan *soak* testing ini memberikan gambaran yang lebih utuh tentang performa dan keandalan aplikasi [43], [44].

2.4.6. Metrik Performa API

Evaluasi performa API umumnya dilakukan dengan menggunakan beberapa indikator kunci yang dapat diukur secara objektif. *Response Time* mengacu pada waktu rata-rata yang diperlukan API untuk menanggapi permintaan, yang berhubungan erat dengan efisiensi eksekusi operasi basis data serta kepuasan pengguna [45]. *Latency* mengukur jeda waktu antara pengiriman permintaan dan diterimanya respons pertama, sedangkan *Throughput* mencerminkan jumlah permintaan yang dapat diproses API per detik [4].

Selain itu, *Error Rate* menunjukkan persentase permintaan yang gagal diproses dan berguna untuk membantu pengembang mengidentifikasi permasalahan dalam implementasi API [45]. *Concurrency Handling* menilai kapasitas API dalam menangani permintaan bersamaan, yang dipengaruhi oleh kemampuan sumber daya server seperti CPU, memori, dan bandwidth jaringan, sehingga kapasitas server untuk menangani koneksi simultan dipengaruhi oleh

faktor-faktor tersebut . *Resource Utilization* memonitor penggunaan CPU, memori, dan bandwidth secara keseluruhan, menjadi tolok ukur penting dalam memastikan API berjalan optimal tanpa membebani infrastruktur .

Metrik-metrik tersebut menjadi dasar yang cukup dalam merancang eksperimen penelitian untuk membandingkan performa antara tRPC dan REST API dengan berbagai skenario beban kerja, guna mendapatkan gambaran yang komprehensif mengenai kelebihan dan kekurangan masing-masing pendekatan.

BAB III
METODOLOGI PENELITIAN

3.1. Pengantar

Penelitian ini merupakan penelitian rekayasa perangkat lunak yang bertujuan untuk melakukan analisis komparatif terhadap performa tRPC dan REST API dalam implementasi sistem *e-commerce* berbasis web. Jenis penelitian ini dipilih karena fokus utama adalah pada pengembangan dan pengujian empiris dua solusi teknologi komunikasi API yang dapat langsung diimplementasikan dalam sistem produksi. Penelitian bukan hanya sekadar mengangkat teori, melainkan menghasilkan bukti kuantitatif mengenai performa masing-masing protokol dalam skenario *e-commerce* yang nyata.

Pendekatan yang digunakan adalah *prototyping* dengan eksperimen terkontrol. Metode ini memungkinkan peneliti untuk membangun dua implementasi sistem *e-commerce* yang identik (satu dengan REST API, satu dengan tRPC), menjalankan serangkaian pengujian performa dalam lingkungan terkontrol, mendapatkan hasil empiris, dan kemudian melakukan analisis komparatif berdasarkan metrik yang telah ditentukan. *Prototyping* dipilih karena memungkinkan iterasi cepat dalam pengembangan fitur dan pengujian, sementara pendekatan eksperimental memastikan validitas dan reliabilitas hasil penelitian.

Penelitian ini melibatkan beberapa tahapan utama: identifikasi masalah spesifik terkait performa API dalam *e-commerce*, analisis kebutuhan fungsional sistem, perancangan arsitektur kedua protokol, implementasi kedua sistem, pengujian performa dengan berbagai skenario beban kerja, evaluasi hasil berdasarkan metrik kuantitatif, serta penyimpulan rekomendasi pemilihan arsitektur yang sesuai. Dengan pendekatan ini, hasil penelitian diharapkan dapat memberikan data empiris bagi pengembang sehingga dapat membantu menentukan teknologi API yang paling optimal untuk proyek e-commerce modern.

3.2. Tahapan Penelitian

3.2.1. Identifikasi Masalah

Pertama adalah kekosongan data empiris tRPC. Meskipun tRPC menawarkan *end-to-end type safety* dengan TypeScript dan potensi efisiensi komunikasi yang lebih baik dibanding REST, belum ada penelitian akademik yang secara kuantitatif membandingkan performa kedua protokol dalam konteks *e-commerce* berbasis web.

Kedua ada di relevansi praktis konteks *e-commerce.* Pertumbuhan pasar e-commerce global yang diperkirakan mencapai USD 70,9 triliun pada 2028 dengan CAGR 27,5% menunjukkan urgensi optimasi arsitektur API untuk menangani volume transaksi dan data pelanggan yang semakin besar.

Ketiga ada di *trade-off* yang belum dipetakan. Penelitian terdahulu (seperti Niswar et al., Chandra dan Farisi) telah menunjukkan *trade-off* antara REST, GraphQL, dan gRPC, namun belum memberikan perbandingan spesifik antara tRPC dan REST dalam skenario *e-commerce* modern.

Fokus identifikasi masalah diarahkan pada:

- Minimnya bukti kuantitatif mengenai performa tRPC dibandingkan REST pada beban e-commerce nyata dengan metrik spesifik (*response time, latency, throughput, error rate, concurrency handling, resource utilization*).
- Diperlukan evaluasi empiris dalam lingkungan terkontrol dengan konfigurasi server identik untuk memastikan validitas hasil perbandingan.

Hasil identifikasi ini menjadi justifikasi penelitian sehingga dapat memberikan tujuan penelitian yang spesifik. Untuk menjadi kontribusi data empiris untuk bisa menjadi pertimbangan dalam proses pengembangan *e-commerce* ataupun dalam proses migrasi REST ke tRPC atau sebaliknya

3.2.2. Analisis Kebutuhan

Kebutuhan fungsional sistem *e-commerce* dalam penelitian ini mencakup beberapa modul utama yang saling terintegrasi untuk mendukung alur belanja pengguna dari tahap penelusuran produk hingga penyelesaian transaksi. Modul

katalog produk menyediakan fitur untuk menampilkan daftar produk dalam bentuk halaman yang dibatasi sejumlah item per halaman, biasanya pada kisaran belasan produk, sehingga tampilan tetap ringan dan mudah diakses. Modul ini juga mendukung penyaringan berdasarkan kategori, rentang harga, maupun rating, serta pencarian produk menggunakan kata kunci, dan menampilkan detail produk secara lengkap, termasuk deskripsi, gambar, ketersediaan stok, dan harga, dengan ukuran *payload* JSON yang umumnya hanya beberapa *kilobyte* per produk agar tetap efisien.

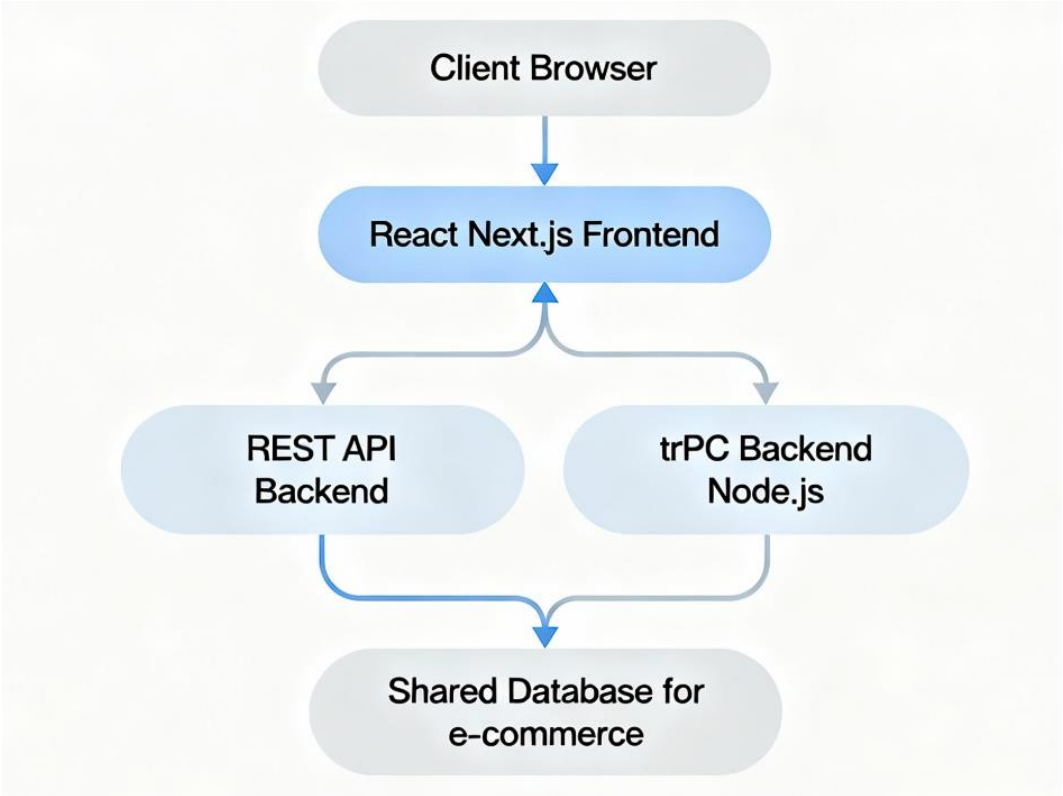
Modul keranjang belanja berfungsi untuk mengelola daftar produk yang dipilih pengguna sebelum melakukan *checkout*. Pengguna dapat menambahkan produk ke keranjang, mengubah jumlah item sesuai kebutuhan, maupun menghapus item yang tidak lagi diinginkan. Sistem secara otomatis melakukan perhitungan ulang total harga dan pajak secara hampir *real-time* setiap kali terjadi perubahan pada isi keranjang, serta menjaga agar kondisi keranjang tetap tersinkronisasi antara frontend dan backend sehingga informasi yang ditampilkan pada antarmuka selalu sesuai dengan data yang tersimpan di server.

Pada tahap berikutnya, modul *checkout* dan pembayaran menangani proses penyelesaian transaksi. Sistem melakukan validasi terhadap data pengiriman yang diisi pengguna, seperti nama, alamat lengkap, kota, dan kode pos, kemudian menyediakan beberapa pilihan metode pembayaran sesuai skenario yang dirancang. Setelah pengguna mengonfirmasi pesanan, sistem menyimpan data transaksi sebagai riwayat yang dapat diakses kembali, sehingga setiap pesanan terdokumentasi dengan baik untuk keperluan pelacakan maupun analisis. Modul riwayat pembelian kemudian memungkinkan pengguna melihat daftar pesanan yang pernah dilakukan, menampilkan detail setiap pesanan seperti status, tanggal, dan total biaya, sekaligus memfasilitasi pemantauan status pengiriman hingga pesanan dianggap selesai.

Untuk menguji performa kedua pendekatan API, disusun beberapa skenario pengujian yang merepresentasikan pola penggunaan nyata dengan intensitas berbeda. Skenario *browse* dikategorikan sebagai beban rendah, di mana pengguna hanya membuka halaman katalog, menerapkan filter produk, dan melihat detail

beberapa produk dalam satu sesi tipikal, skenario ini *menghasilkan* sekitar belasan panggilan API. Skenario *shopping* memiliki intensitas menengah, karena selain melakukan penelusuran, pengguna juga menambahkan beberapa produk ke keranjang dan melihat ringkasan keranjang, sehingga total permintaan ke server meningkat menjadi sekitar puluhan panggilan API dalam satu alur interaksi. Sementara itu, skenario checkout dikategorikan sebagai beban tinggi karena mencakup seluruh alur belanja lengkap, mulai dari penelusuran produk, penambahan ke keranjang, proses *checkout*, hingga pembayaran dan penayangan konfirmasi beserta riwayat pesanan, yang secara keseluruhan dapat menghasilkan puluhan hingga mendekati setengah ratus panggilan API untuk satu transaksi utuh.

3.2.3. Perancangan Sistem



Gambar 3. 3. Diagram Rancangan Sistem

Perancangan sistem dilakukan dengan dua pendekatan paralel: satu implementasi REST API dan satu implementasi tRPC. Kedua sistem dirancang

seidentik mungkin dalam hal fungsionalitas dan logika bisnis, namun berbeda dalam protokol komunikasi API. Desain sistem juga mencakup konfigurasi server identik untuk memastikan perbandingan performa yang adil.

Sistem pengujian dibangun dengan arsitektur multi-tier. Arsitektur yang digunakan terdiri dari lapisan client (browser pengguna), lapisan frontend web berbasis React, lapisan backend Node.js yang mengimplementasikan REST API dan tRPC secara paralel, serta satu basis data bersama yang menyimpan seluruh data e-commerce. Kedua varian backend di-deploy pada lingkungan server dengan spesifikasi perangkat keras dan perangkat lunak yang sama untuk menjaga keadilan dalam pengukuran performa.

Backend server dikembangkan menggunakan Node.js dengan dua varian API, yaitu tRPC dan REST. tRPC dimanfaatkan untuk menghadirkan *type safety* end-to-end melalui TypeScript, sehingga komunikasi antara frontend dan backend menjadi lebih terkontrol, konsisten, dan meminimalkan bug terkait tipe data. Pendekatan ini juga mempermudah proses pengembangan karena perubahan tipe di backend dapat langsung terpantau di sisi frontend. Sementara itu, REST API menggunakan pola HTTP/1.1 dengan format data JSON yang sudah sangat umum digunakan, sehingga tetap kompatibel dengan berbagai jenis klien dan tool integrasi yang ada saat ini.

Pada tahap perancangan API, setiap operasi utama seperti pendaftaran pengguna, autentikasi, pengelolaan produk, dan proses checkout didefinisikan baik sebagai endpoint REST maupun prosedur tRPC dengan struktur permintaan dan respon yang setara, sehingga hasil pengujian dapat dibandingkan secara langsung dan objektif.

Pada sisi frontend, aplikasi dibangun dengan React dalam framework Next.js dan TypeScript agar integrasi dengan kedua varian API tersebut berjalan mulus. Penggunaan TypeScript di frontend membantu menjaga keselarasan tipe data dengan backend, terutama ketika membandingkan perilaku antara tRPC dan REST. Desain UI/UX komponen React secara bertahap. Pendekatan ini membantu menjaga konsistensi tampilan, alur interaksi, dan pengalaman pengguna, meskipun terjadi perpindahan atau perbandingan penggunaan antara tRPC dan REST

3.2.4. Implementasi dan Pengujian

Tahap implementasi diawali dengan pengembangan dua varian sistem secara paralel:

Satu menggunakan REST API dan satu menggunakan tRPC, keduanya berbasis tech stack yang sama dari sisi frontend, backend, maupun basis data. Frontend dibangun menggunakan React dengan TypeScript dan didukung Next.js untuk menangani proses pengambilan dan penyegaran data dari server, dengan komunikasi HTTP menggunakan protokol HTTP/1.1 dan payload berbasis JSON sesuai praktik umum pengembangan aplikasi web modern. Pada sisi backend, kedua varian API diimplementasikan di atas Node.js dengan framework berbasis *middleware* seperti Express atau kerangka serupa, sehingga struktur aplikasi dapat dipisahkan ke dalam lapisan-lapisan logika bisnis, layanan, dan akses data. Basis data yang digunakan adalah PostgreSQL generasi terbaru yang stabil, dikonfigurasi sebagai penyimpanan utama seluruh data e-commerce.

Dalam implementasi REST API, server dikonfigurasi untuk menerima dan memproses request HTTP pada serangkaian *endpoint* yang dikelompokkan di bawah jalur versi tertentu. Endpoint disusun mengikuti pola yang konsisten yang disesuaikan dengan kebutuhan fungsional sistem *e-commerce*. Untuk meningkatkan kualitas error handling, disiapkan mekanisme penanganan error terpusat di akhir rantai middleware, sehingga setiap *error* yang tidak tertangani di level atas dapat dikonversi menjadi respons HTTP dengan kode status yang sesuai dan format pesan yang konsisten.

Pada implementasi tRPC, fokus utama adalah mendefinisikan router dan prosedur yang mencerminkan kebutuhan bisnis yang sama dengan REST, namun dikemas dalam bentuk pemanggilan prosedur *remote* yang *type-safe*. Struktur router dipecah ke dalam modul-modul. Setiap prosedur didefinisikan dengan tipe input dan output yang eksplisit menggunakan skema validasi, sehingga saat frontend melakukan pemanggilan, kesalahan tipe data dapat dideteksi lebih awal di sisi pengembangan melalui sistem tipe TypeScript. Integrasi tRPC dengan server HTTP dilakukan melalui middleware yang menghubungkan jalur tertentu, misalnya /trpc, dengan router aplikasi, sehingga alur request–response tRPC berjalan di atas

server Node.js yang sama dengan implementasi REST dan tetap dapat diukur dalam kondisi lingkungan yang identik.

Sebelum masuk ke pengujian performa, dilakukan pengujian fungsional dengan pendekatan *black-box* untuk memastikan bahwa kedua implementasi, baik REST maupun tRPC, memenuhi kebutuhan fungsional yang telah ditetapkan. Seluruh pengujian dilakukan dari perspektif pengguna tanpa melihat detail internal kode, dengan memeriksa apakah sistem memberikan respons yang benar untuk input yang valid maupun penanganan yang tepat untuk input yang tidak valid. Target dari tahap ini adalah memastikan bahwa semua fungsi utama berjalan sebagaimana rancangan tanpa error fungsional yang terdeteksi, sehingga perbedaan hasil pada tahap pengujian performa nantinya tidak disebabkan oleh perbedaan implementasi fitur.

Untuk menjaga keadilan dalam perbandingan performa, pengujian dijalankan pada sebuah server dengan spesifikasi kelas menengah yang cukup umum. Basis data diisi dengan data contoh dalam jumlah yang cukup besar untuk mensimulasikan lingkungan e-commerce nyata, misalnya pada kisaran puluhan ribu entri produk dengan atribut seperti nama, deskripsi, harga, stok, dan kategori. Untuk mendukung performa pada operasi yang sering digunakan, dibuat indeks pada kolom-kolom yang sering dijadikan kriteria pencarian atau relasi, seperti kategori produk, rentang harga, dan keterkaitan pesanan dengan pengguna, sehingga query yang dijalankan oleh kedua varian API dapat berjalan pada kondisi yang optimal dan setara.

Hasil akhir dari tahap implementasi dan pengujian ini berupa kode sumber lengkap untuk kedua pendekatan API yang mengimplementasikan fungsionalitas e-commerce yang sama, basis data yang telah terisi data uji dalam skala yang memadai, serta laporan pengujian fungsional yang menunjukkan bahwa kedua sistem berperilaku identik dari sudut pandang pengguna. Selain itu, terdokumentasi pula konfigurasi lingkungan pengujian, meliputi spesifikasi perangkat keras, perangkat lunak, dan parameter basis data, yang akan menjadi acuan ketika menganalisis dan menginterpretasikan hasil pengujian performa pada tahap berikutnya.

3.2.5. Evaluasi Sistem

Bagian evaluasi sistem pada penelitian ini berfokus pada pengujian performa kedua varian API (REST dan tRPC) dengan menggunakan Apache JMeter untuk mensimulasikan beban pengguna yang mendekati kondisi nyata. Pengujian dilakukan dengan menjalankan JMeter pada mesin terpisah dari server aplikasi, sehingga latensi jaringan dan *overhead* komunikasi ikut tercermin secara lebih realistis. Seluruh skenario dan konfigurasi disusun tertentu agar kedua API diuji dengan pola beban, jumlah virtual user, dan durasi yang setara, sehingga hasil evaluasi dapat dibandingkan secara adil.

Dari sisi rancangan beban, JMeter dikonfigurasi dalam satu test plan yang berisi beberapa kelompok beban yang merepresentasikan tahapan peningkatan tekanan terhadap sistem. Pengujian dimulai dengan fase awal menggunakan puluhan pengguna virtual selama beberapa menit untuk testing awal server dan membentuk koneksi awal, dilanjutkan dengan beban dasar yang mewakili kondisi operasi normal web tersebut. Setelah itu, beban dinaikkan ke level yang merefleksikan lalu lintas jam sibuk yang dijalankan dalam durasi yang lebih lama, kemudian dilanjutkan ke fase beban puncak yang lebih tinggi lagi. Tahap terakhir berupa pengujian stress dengan jumlah pengguna virtual yang melampaui beban yang diperkirakan, dengan tujuan melihat titik saat sistem mulai menunjukkan kegagalan atau penurunan performa yang signifikan.

Skenario perilaku pengguna dimodelkan menjadi tiga alur utama yang berjalan paralel dalam pengujian, yaitu *browse flow*, *shopping flow*, dan *checkout flow*. *Browse flow* menggambarkan pengguna yang hanya menelusuri katalog, memuat daftar produk, membuka beberapa detail produk, dan melakukan pencarian sederhana dengan jeda antarpemintaan untuk meniru waktu berpikir. *Shopping flow* memodelkan pengguna yang aktif mengelola keranjang, mulai dari menelusuri produk, menambahkan beberapa item, melihat ringkasan keranjang, hingga mengubah jumlah item sehingga menghasilkan beban operasi tulis yang lebih besar. *Checkout flow* merepresentasikan transaksi lengkap, mulai dari pengambilan riwayat pesanan, pembuatan pesanan baru dengan detail transaksi, hingga penayangan halaman konfirmasi, yang volumenya lebih kecil tetapi paling berat

dari sisi kompleksitas proses.

Pengujian dibagi ke dalam beberapa jenis performance testing yang berbeda untuk memberikan gambaran menyeluruh mengenai perilaku sistem. *Load* testing dilakukan dengan menjalankan beban pada tingkat yang mendekati beban operasi normal dan jam sibuk yang realistis, dengan jumlah pengguna virtual dan durasi yang cukup untuk melihat stabilitas respons pada kondisi yang diharapkan. *Stress* testing kemudian mendorong sistem melampaui kapasitas normal, dengan menaikkan jumlah pengguna virtual secara bertahap hingga sistem menunjukkan peningkatan error atau lonjakan waktu respons, sehingga dapat diidentifikasi titik jenuh dan batas toleransi infrastruktur. *Soak* atau *endurance* testing diterapkan dengan menjalankan beban mendekati puncak dalam durasi yang lebih panjang, untuk melihat apakah terdapat masalah atau degradasi performa yang muncul perlahan seiring waktu. Terakhir, *spike* testing mensimulasikan lonjakan beban yang sangat cepat misalnya peningkatan tajam jumlah pengguna dalam waktu singkat untuk mengevaluasi kemampuan sistem merespons dan kembali stabil setelah lonjakan tersebut.

Selama seluruh pengujian, berbagai metrik performa dikumpulkan secara sistematis. Dari sisi waktu respons, dicatat nilai rata-rata, minimum, maksimum, serta beberapa persentil penting seperti median (sekitar persentil ke-50), persentil ke-95, dan persentil ke-99 untuk menggambarkan pengalaman mayoritas pengguna sekaligus mengamati ekor distribusi yang paling lambat. Dari sisi throughput, diamati jumlah permintaan yang dapat diproses per detik, persentase permintaan yang berhasil, serta tingkat error yang muncul pada tiap tingkat beban hal ini dilakukan untuk setiap metrik yang akan diukur. Di luar metrik aplikasi, pemantauan sumber daya server dilakukan dengan alat bantu sistem untuk merekam penggunaan CPU, konsumsi memori, pemakaian bandwidth jaringan, dan aktivitas I/O disk.

Prosedur pengujian untuk setiap skenario disusun dengan langkah yang sama agar hasilnya dapat dibandingkan secara adil. Sebelum pengujian dimulai, server di-restart, cache dibersihkan, dan kondisi basis data disetel ke keadaan awal yang seragam, misalnya dengan memuat data dummy produk dalam jumlah yang

sama untuk setiap siklus uji. Sistem kemudian diberi beban warm-up ringan dalam waktu singkat agar layanan berada pada kondisi stabil sebelum pengukuran utama dilakukan.

Selama pengujian, seluruh metrik dari JMeter disimpan ke berkas hasil, sementara pemantauan sumber daya server (CPU, memori, jaringan, dan I/O disk) dijalankan di latar belakang. Setelah satu siklus pengujian selesai, server diberi jeda cool-down sebelum siklus berikutnya dimulai, dan seluruh data uji diekstraksi serta dibersihkan untuk dikompilasi menjadi dataset yang siap dianalisis untuk membandingkan REST API dan tRPC pada berbagai jenis pengujian dan tingkat beban..

3.3. Usulan Solusi

Membangun dua aplikasi e-commerce berbasis web yang berjalan secara paralel, satu menggunakan REST API konvensional dengan HTTP verb standar dan dokumentasi OpenAPI/Swagger, dan satu lagi menggunakan tRPC dengan arsitektur berbasis prosedur serta dukungan type safety end-to-end.

Aplikasi ini memiliki fitur utama:

1) pengelolaan katalog produk dengan fungsi browse menggunakan paginasi, filter, dan pencarian, serta tampilan detail produk
2) pengelolaan keranjang belanja untuk membuat dan mengatur keranjang, menambah, mengubah, dan menghapus item dengan perhitungan total secara mendekati *real-time*
3) pengelolaan pesanan yang mencakup pembuatan pesanan baru dari keranjang, konfirmasi pesanan, serta tampilan dan print riwayat pembelian pengguna.

Kedua solusi ini mengimplementasikan fitur inti yang sama, yaitu manajemen katalog produk (penelusuran dengan paginasi, fitur filter dan pencarian, serta tampilan detail produk), pengelolaan keranjang belanja (membuat dan mengelola keranjang, menambah, mengubah, dan menghapus item, serta perhitungan nilai keranjang secara mendekati real-time), dan manajemen pesanan (membuat pesanan baru dari keranjang, menampilkan konfirmasi pesanan, serta

melihat riwayat pesanan pengguna).

Arsitektur kedua sistem dirancang sedemikian rupa sehingga alur bisnis dan pengalaman pengguna tetap konsisten, sementara perbedaannya terletak pada protokol API yang digunakan solusi ini ditujukan untuk menghasilkan data empiris perbandingan performa REST dan tRPC serta memberikan rekomendasi dan panduan praktis bagi pengembang dalam memilih teknologi API yang paling sesuai dengan karakteristik aplikasi e-commerce berbasis web.

3.4. Pengumpulan Data

Penelitian ini mengumpulkan data primer dan sekunder dari proses pengembangan dan pengujian dua sistem *e-commerce* berbasis REST dan tRPC sebagai dasar analisis komparatif performa dan kualitas implementasi. Data primer terutama berupa metrik performa hasil pengujian menggunakan Apache JMeter yang disimpan dalam format CSV dan laporan statistik agregat untuk setiap skenario beban, serta log pemantauan sumber daya server seperti CPU, memori, jaringan, dan I/O. Untuk melengkapi data teknis tersebut, hasil uji fungsional (log eksekusi test case beserta status lulus/gagal) dan catatan proses pengembangan atau umpan balik pengguna/pengembang juga dapat dicatat sebagai bahan pendukung interpretasi hasil. Data sekunder diperoleh melalui studi dokumentasi dan literatur yang relevan. Sumber ini mencakup praktik terbaik perancangan REST API, dokumentasi resmi dan sumber komunitas terkait tRPC, serta penelitian akademik dan benchmark industri yang membahas performa API pada berbagai skenario beban.

Data yang terkumpul kemudian diolah dengan pendekatan kuantitatif dan kualitatif secara sederhana namun sistematis. Analisis kuantitatif mencakup statistik deskriptif terhadap metrik seperti waktu respons, *throughput*, error rate, dan metrik lain serta pengamatan hubungan antara tingkat beban dan perubahan performa pada masing-masing arsitektur. Seluruh data mentah, hasil pengolahan, dan skrip analisis disimpan dalam format terstruktur (misalnya CSV, spreadsheet, dan berkas skrip) agar dapat ditelusuri kembali, direplikasi, dan digunakan sebagai dasar pembahasan hasil maupun penelitian lanjutan..

3.5. *Setting* Penelitian

Penelitian ini dilaksanakan menggunakan laptop pribadi dengan spesifikasi yang cukup untuk menunjang kebutuhan pengembangan dan pengujian aplikasi web *e-commerce*. Laptop yang digunakan bermodel Victus by HP, dilengkapi prosesor AMD Ryzen 7 6800H dengan 8 core dan 16 thread, kecepatan clock mendekati 3,1 GHz, memori sebesar 16 GB DDR5 yang berjalan pada kecepatan efektif sekitar 4.8 GHz, serta kartu grafis NVIDIA GeForce RTX 3050 Laptop GPU berkapasitas 4 GB. Sistem operasi yang terpasang adalah Microsoft Windows 11 Home Single Language versi 10.0.26100.

Penggunaan laptop dengan spesifikasi cukup, penting untuk memastikan lingkungan pengembangan backend Node.js, frontend React, dan basis data PostgreSQL berjalan lancar tanpa hambatan sumber daya. Selain itu, spesifikasi ini memungkinkan pelaksanaan pengujian beban secara realistis menggunakan Apache JMeter, di mana pengujian memerlukan kapasitas CPU dan memori yang memadai untuk mensimulasikan ratusan pengguna virtual sekaligus tanpa mengorbankan akurasi hasil.

Penelitian berlangsung selama periode yang fleksibel selama waktu semester pengerjaan TA, dengan pengembangan dan pengujian dilakukan di lingkungan kerja pribadi maupun kampus, dengan koneksi internet menyediakan akses untuk dokumentasi, pencatatan melalui GitHub, serta pemanfaatan layanan cloud jika diperlukan. Penelitian ini dilaksanakan dengan arsitektur terdistribusi untuk memastikan validitas dan akurasi hasil pengujian performa. Server aplikasi, yang mencakup backend Node.js (untuk REST dan tRPC), frontend React, dan basis data PostgreSQL, di-deploy pada sebuah virtual private server (VPS) di penyedia layanan cloud (misalnya DigitalOcean Droplet atau sejenisnya) dengan spesifikasi standar industri (misalnya, 2 vCPU, 4 GB RAM, dan SSD).

Sementara itu, mesin pengujian (klien) yang menjalankan perangkat lunak Apache JMeter berlokasi di mesin terpisah, yaitu laptop pribadi peneliti (Victus by HP). Pemisahan ini bertujuan untuk mengisolasi sumber daya, sehingga beban yang dihasilkan oleh JMeter tidak memengaruhi performa server aplikasi yang sedang diukur. Seluruh pengujian dilakukan melalui jaringan internet publik untuk

mensimulasikan latensi dan kondisi jaringan dunia nyata yang akan dihadapi oleh pengguna akhir.

Subjek penelitian ini adalah analisis komparatif performa arsitektur API, yaitu REST dan tRPC, dalam konteks aplikasi *e-commerce*. Objek yang diukur adalah metrik-metrik performa (seperti response time, throughput, dan error rate dan metrik lain) yang dihasilkan oleh kedua implementasi API ketika diberi beban kerja yang identik.

3.6. Alat Bantu atau Tools yang Digunakan

- Next.js 14.x (dengan React 18.x): Digunakan sebagai framework frontend untuk membangun antarmuka, menyediakan routing bawaan, rendering sisi server (SSR), dan optimasi performa.
- TypeScript: dimanfaatkan untuk static type checking di sisi frontend dan backend, sehingga kontrak data antara REST maupun tRPC lebih konsisten dan mudah dipelihara.
- PostgreSQL 15.x: berperan sebagai basis data relasional untuk menyimpan data produk, keranjang, dan pesanan, dengan optimasi query melalui indeks pada kolom-kolom yang sering diakses.
- Express.js 4.x: digunakan sebagai web framework untuk implementasi REST API, menyediakan routing dan middleware yang fleksibel untuk menangani request dan response.
- tRPC: digunakan untuk implementasi varian API kedua dengan pendekatan RPC bertipe kuat, dilengkapi validasi input menggunakan Zod dan integrasi yang erat dengan TypeScript.
- Apache JMeter 5.6: dimanfaatkan sebagai tool utama untuk load testing, stress testing, dan skenario performa lainnya terhadap kedua API, dengan output berupa file CSV berisi metrik respons dan error.
- Postman Collection: digunakan untuk pengujian manual dan verifikasi fungsional awal endpoint REST dan tRPC sebelum skenario otomatis dijalankan di JMeter.
- Prometheus dan Grafana : dipakai untuk mengumpulkan metrik

server seperti CPU, memori, jaringan, lalu menampilkannya dalam dasbor real-time selama pengujian performa.

- Python 3.11+ dan R: dimanfaatkan untuk analisis data hasil pengujian menggunakan pustaka seperti pandas, numpy, scipy, atau ggplot2, serta untuk membuat visualisasi statistik yang lebih lanjut.
- Git dan GitHub: digunakan sebagai sistem version control dan kolaborasi, untuk REST, tRPC, frontend.
- Visual Studio Code: menjadi IDE utama dengan ekstensi seperti ESLint, Prettier, REST Client, dan integrasi database untuk mendukung pengembangan dan debugging.
- Docker : dimanfaatkan untuk membuat lingkungan terkontainer baik untuk backend maupun database, sehingga pengujian dapat dilakukan pada environment yang konsisten.
- Spreadsheet dan tools diagram (Google Sheets/Excel, Draw.io): dipakai untuk pengelolaan awal data hasil pengujian, pembuatan pivot sederhana, serta pembuatan diagram arsitektur dan sequence diagram pendukung dokumentasi.

Seluruh tools tersebut dipilih untuk mendukung alur kerja pengembangan modern yang mencakup implementasi dua varian API, pengujian fungsional dan performa, pemantauan sumber daya, hingga analisis dan dokumentasi hasil secara sistematis. Kombinasi stack pengembangan (Node.js, React, TypeScript), tool pengujian (JMeter, Postman), monitoring (Prometheus, Grafana), serta analisis data (Python, R, spreadsheet) diharapkan mampu menyediakan landasan teknis yang kuat untuk membandingkan REST dan tRPC secara objektif pada konteks sistem e-commerce berbasis web.

3.7. Evaluasi

Evaluasi sistem dilakukan melalui dua aspek utama, yaitu evaluasi fungsional dan evaluasi performa, yang kemudian dilengkapi dengan analisis statistik untuk membandingkan perilaku REST dan tRPC secara kuantitatif.

Evaluasi Fungsional:

Evaluasi fungsional dalam penelitian ini dilaksanakan menggunakan pendekatan black-box testing yang berfokus sepenuhnya pada validasi perilaku eksternal sistem tanpa meninjau struktur kode internal. Pengujian ini dilakukan secara manual terhadap kurang lebih lima belas skenario inti yang merepresentasikan alur kerja utama dalam aplikasi e-commerce, mencakup operasi CRUD pada katalog produk, manajemen keranjang belanja, serta pemrosesan pesanan akhir. Tujuan utama dari fase ini adalah memastikan bahwa logika bisnis yang diimplementasikan pada kedua arsitektur, baik REST API maupun tRPC, telah berjalan sesuai dengan spesifikasi kebutuhan yang dirancang sebelumnya.

Integritas dan konsistensi data menjadi prioritas utama dalam evaluasi ini, di mana setiap skenario diuji untuk memastikan persistensi informasi yang akurat. Pengujian mencakup verifikasi bahwa data dalam keranjang belanja tetap tersimpan dengan benar setelah halaman diperbarui, status pesanan tercatat secara presisi di basis data, serta kalkulasi nilai transaksi seperti total harga dan pajak dihitung secara real-time tanpa kesalahan. Selain itu, validasi respons sistem dilakukan dengan memeriksa kesesuaian struktur keluaran terhadap skema yang telah didefinisikan untuk REST atau definisi tipe data yang berlaku pada tRPC, serta memastikan mekanisme penanganan kesalahan mampu memberikan pesan yang informatif untuk membantu proses debug jika terjadi kegagalan.

Tahapan pengujian fungsional ini ditetapkan sebagai prasyarat mutlak yang harus diselesaikan sebelum penelitian dapat melangkah ke tahap pengujian performa. Target keberhasilan ditetapkan pada angka 100% lulus tanpa error untuk seluruh skenario kritis, dengan toleransi kesalahan 0% pada fitur-fitur wajib. Hanya setelah sistem dinyatakan stabil dan seluruh fungsi logika bisnis terbukti berjalan tanpa cacat, pengujian beban menggunakan alat bantu seperti Apache JMeter dapat dilaksanakan. Hal ini dilakukan untuk menjamin bahwa data performa yang diambil nantinya valid dan tidak bias oleh kegagalan fungsi dasar sistem.

Evaluasi Performa:

Evaluasi performa dalam penelitian ini dilakukan secara komprehensif menggunakan k6 sebagai perangkat pengujian beban utama. Pemilihan k6 didasari oleh kompatibilitasnya yang tinggi dengan ekosistem berbasis JavaScript dan

kemampuannya dalam menangani pengujian protokol tRPC secara efisien dibandingkan alat konvensional lainnya. Untuk pemantauan penggunaan sumber daya perangkat keras, sistem akan diintegrasikan dengan Prometheus untuk pengumpulan metrik server secara berkala dan Grafana untuk visualisasi data penggunaan CPU serta memori secara real-time. Fokus evaluasi mencakup metrik vital seperti rata-rata waktu respons, latensi persentil ke-95 dan ke-99 atau p95 dan p99, throughput transaksi per detik, serta tingkat kesalahan yang terjadi selama pengujian berlangsung.

Metodologi pengujian dirancang mencakup empat pendekatan strategi beban yang berbeda, yaitu Load Testing, Stress Testing, Spike Testing, dan Soak Testing. Load Testing bertujuan mengukur performa sistem di bawah beban kerja normal yang diharapkan untuk memastikan stabilitas operasional sehari-hari. Stress Testing dilakukan dengan menaikkan beban pengguna secara bertahap hingga melampaui batas kapasitas sistem untuk mengidentifikasi titik henti atau breaking point. Spike Testing mensimulasikan lonjakan lalu lintas pengguna yang tiba-tiba dan ekstrem dalam waktu singkat guna menguji elastisitas sistem dalam menangani anomali trafik. Terakhir, Soak Testing dijalankan dalam durasi yang panjang dengan beban konstan untuk mendeteksi potensi kebocoran memori atau memory leaks dan penurunan performa seiring berjalannya waktu.

Pelaksanaan pengujian akan menerapkan keempat metode tersebut pada kelima belas skenario fungsional e-commerce yang telah didefinisikan sebelumnya, mulai dari penelusuran katalog hingga penyelesaian transaksi. Setiap skenario dalam masing-masing metode pengujian akan dijalankan dengan variasi tingkat konkurensi pengguna virtual yang berbeda-beda untuk mendapatkan gambaran performa yang granular pada berbagai tingkat intensitas akses. Pendekatan matriks ini memastikan bahwa setiap fitur tidak hanya diuji fungsinya, tetapi juga ketahanannya terhadap pola akses yang beragam dan dinamis.

Seluruh data mentah yang dihasilkan dari rangkaian pengujian ini tidak akan langsung dianalisis, melainkan dikumpulkan terlebih dahulu ke dalam format file terstruktur seperti CSV atau JSON. Proses pra-pemrosesan data kemudian dilakukan untuk merapikan, memvalidasi, dan menyaring anomali pencatatan yang

mungkin terjadi selama eksekusi skrip pengujian. Tahap konsolidasi data ini sangat krusial untuk memastikan bahwa analisis statistik yang dilakukan selanjutnya didasarkan pada dataset yang bersih, valid, dan representatif terhadap kondisi performa sistem yang sesungguhnya.Untuk menilai apakah perbedaan performa antara REST dan tRPC bermakna atau tidak, digunakan uji signifikansi statistik dengan hipotesis nol bahwa tidak ada perbedaan yang berarti pada tingkat signifikansi 0,05. Jika nilai p-value lebih kecil dari ambang tersebut, hipotesis nol ditolak dan disimpulkan bahwa perbedaan performa kedua pendekatan tersebut signifikan secara statistik.

Berikut adalah contoh konkret skenario pengujian untuk setiap metode yang berfokus pada satu alur pengguna tunggal, yaitu "menelusuri katalog, menambah produk ke keranjang, dan menyelesaikan pesanan" setiap pengujian ini dilakukan pada kedua protokol API dengan jumlah dan skenario sama dan dilakukan oleh simulasi sehingga perbedaan murni pada protokol API.

- **Load Testing:** Skenario ini dijalankan untuk mengukur performa pada beban yang diharapkan. Pengujian dilakukan dengan tiga tingkatan pengguna virtual (VU) yang konstan dan berkelanjutan: 50 VU, 100 VU, dan 200 VU. Setiap tingkatan dijalankan selama 15 menit untuk mengukur metrik seperti waktu respons rata-rata dan throughput pada kondisi beban normal.
- **Stress Testing:** Tujuannya adalah menemukan titik batas kapasitas sistem. Pengujian dimulai dengan 200 VU, kemudian jumlah pengguna ditingkatkan secara bertahap (misalnya, ditambah 50 VU setiap 5 menit) hingga sistem menunjukkan tingkat kesalahan yang signifikan (di atas 5%) atau waktu respons melampaui ambang batas yang dapat diterima.
- **Spike Testing:** Metode ini mensimulasikan lonjakan trafik secara tiba-tiba. Pengujian berjalan pada beban dasar rendah sebesar 50 VU, kemudian secara instan dinaikkan hingga 500 VU selama periode singkat (misalnya 2 menit) sebelum kembali ke beban dasar. Skenario ini menguji kemampuan sistem untuk pulih setelah lonjakan trafik yang ekstrem.
- **Soak Testing:** Bertujuan untuk mengidentifikasi masalah stabilitas jangka

panjang. Skenario alur pengguna dijalankan secara terus-menerus pada beban kerja konstan sebesar 150 VU selama periode waktu yang diperpanjang, misalnya 4 hingga 8 jam, untuk memantau potensi kebocoran memori atau penurunan performa dari waktu ke waktu.

Analisis Statistik:

Evaluasi statistik dalam penelitian ini dirancang untuk menangani volume data yang masif yang dihasilkan dari matriks pengujian performa. Mengingat kompleksitas variasi pengujian—yang mencakup empat metode (load, stress, spike, soak), lima belas skenario fungsional, dan berbagai tingkatan konkurensi pengguna (VU) akan terbentuk puluhan hingga ratusan grup data perbandingan antara REST dan tRPC. Setiap pasangan grup data ini, misalnya "Skenario Checkout pada Load Test 100 VU", akan diperlakukan sebagai entitas analisis independen yang memerlukan validasi statistik tersendiri sebelum ditarik kesimpulan global.

Proses analisis dimulai dengan tabulasi data secara granular, di mana setiap grup data akan diuji menggunakan uji signifikansi statistik untuk mendapatkan nilai p-value. Nilai ini berfungsi sebagai filter awal untuk menentukan apakah perbedaan performa yang teramati antara kedua arsitektur bersifat nyata secara statistik atau sekadar kebetulan. Namun, untuk menghindari kesalahan interpretasi seperti false positive akibat ukuran sampel yang besar, analisis diperdalam dengan perhitungan Cohen’s d untuk mengukur effect size. Metrik ini krusial untuk memetakan besaran dampak praktis dari perbedaan tersebut, mengategorikannya dari sangat kecil hingga besar, sehingga perbedaan yang statistik-signifikan tetapi remeh secara praktis dapat diidentifikasi.

Analisis akhir tidak akan bergantung pada satu nilai universal yang menyamaratakan seluruh kondisi, melainkan dibangun berdasarkan sintesis tren dari seluruh grup data yang telah diolah. Hasil perhitungan statistik (p-value dan Cohen’s d) akan disandingkan dengan analisis deskriptif terhadap metrik non-inferensial seperti tingkat kesalahan (error rate) dan efisiensi penggunaan sumber daya (CPU/Memori). Kesimpulan penelitian akan ditarik secara induktif dengan melihat pola dominan yang muncul di seluruh skenario apakah tRPC konsisten mengungguli REST pada beban tinggi, atau apakah REST lebih stabil pada skenario

spesifik sehingga menghasilkan rekomendasi arsitektur yang berlandaskan bukti empiris yang komprehensif dan berkonteks, bukan sekadar generalisasi angka rata-rata.

Contoh Penerapan Analisis per Skenario :

- Load Testing pada Skenario Checkout 100 VU
  Data menunjukkan REST memiliki rata-rata respons 200 ms dengan error 0,5%, sedangkan tRPC 150 ms dengan error 0,2%. Analisis dilakukan menggunakan t-test dan Cohen's d pada dataset waktu respons untuk memvalidasi apakah percepatan 50 ms tersebut signifikan secara statistik dan seberapa besar dampak praktisnya. Data penggunaan CPU dan memori kemudian disajikan secara deskriptif untuk melengkapi konteks efisiensi.
- Stress Testing pada Skenario Search 380 VU
  Pada titik beban kritis ini, REST mencatat respons 450 ms dengan error 3%, sementara tRPC bertahan di 320 ms dengan error 1%. Analisis statistik difokuskan untuk membuktikan bahwa tRPC secara signifikan lebih tahan banting saat sistem mendekati batas kapasitas. Tingkat error rate yang meningkat pada REST dijelaskan sebagai indikator ketidakstabilan yang tidak dialami tRPC.
- Spike Testing pada Skenario Add-to-Cart Lonjakan 50 ke 300 VU
  Analisis hanya mengambil data pada jendela waktu 30 detik saat lonjakan terjadi. Dengan data REST di 600 ms (5% error) dan tRPC di 380 ms (1,5% error), uji statistik dilakukan untuk membuktikan keunggulan elastisitas tRPC. Data deskriptif digunakan untuk menyoroti kemampuan tRPC meredam lonjakan trafik tanpa kegagalan sistem yang fatal.

Contoh Penarikan Kesimpulan Induktif dari Tren

Pada beban rendah 50 VU, uji statistik menunjukkan perbedaan signifikan tetapi nilai Cohen's d kecil, yang berarti perbedaan performa ada namun tidak terasa bagi pengguna. Pada beban sedang 150 VU, nilai Cohen's d naik menjadi sedang, menandakan selisih performa mulai terlihat nyata. Pada beban tinggi 300 VU, nilai

Cohen’s d melonjak menjadi besar, diiringi data deskriptif di mana REST mengalami bottleneck sumber daya. Kesimpulan akhirnya adalah bahwa arsitektur tRPC memiliki skalabilitas yang superior dibandingkan REST. Tren data menunjukkan bahwa keunggulan tRPC bersifat progresif, di mana manfaat performanya semakin kritis dan signifikan justru ketika sistem menghadapi beban kerja yang berat, bukan pada kondisi trafik sepi.

Kesimpulan :

Penelitian ini dinyatakan berhasil apabila memenuhi serangkaian indikator kelayakan teknis yang mencakup aspek fungsional, validitas data performa, dan kedalaman analisis statistik. Pada tahap awal, evaluasi fungsional harus mencapai tingkat kelulusan mutlak 100% untuk seluruh skenario inti, memastikan bahwa kedua sistem memiliki perilaku logika bisnis yang identik dan bebas dari kesalahan kritis sebelum dilakukan pengujian lebih lanjut.

Dalam aspek performa, indikator keberhasilan ditentukan oleh terkumpulnya dataset yang valid dan lengkap dari keempat metode pengujian beban, yaitu Load, Stress, Spike, dan Soak Testing. Data ini harus mencakup pengukuran metrik kunci pada berbagai variasi jumlah pengguna virtual, sehingga dapat memberikan gambaran komprehensif mengenai perilaku sistem di bawah kondisi tekanan yang berbeda-beda.

Secara analitis, penelitian dianggap sukses jika data yang dihasilkan memadai untuk diuji menggunakan metode statistik inferensial, mencakup perhitungan nilai p-value untuk signifikansi dan Cohen’s d untuk mengukur effect size. Analisis ini bertujuan untuk memvalidasi perbedaan performa secara kuantitatif dan menyusun kesimpulan induktif berdasarkan tren data, sehingga rekomendasi arsitektur yang dihasilkan memiliki landasan empiris yang kuat.

Adapun rincian final mengenai matriks skenario pengujian akan dibakukan setelah tahap analisis kebutuhan sistem diselesaikan. Hal ini dilakukan untuk menjamin bahwa setiap skenario yang dipilih benar-benar merepresentasikan alur kerja kritis yang relevan, sehingga pengujian yang dilakukan tepat sasaran dan hasil analisisnya dapat diaplikasikan pada kasus penggunaan nyata.

3.8. Jadwal Kegiatan

Rancangan tahapan kegiatan

- Analisis kebutuhan fungsional e-commerce, skenario beban (browse, shopping, checkout), serta pemetaan metrik performa dan skenario pengujian.
- Perancangan sistem meliputi arsitektur multi-tier, rancangan endpoint REST dan prosedur tRPC, desain skema basis data.
- Implementasi Frontend Web menggunakan React dalam framework Next.js.
- Implementasi REST API dan implementasi tRPC dilakukan paralel dan bersamaan di akhir penyelesaian frontend.
- Pengujian fungsional (black-box) dilaksanakan setelah fitur inti stabil untuk memastikan kedua sistem berperilaku identik sebelum pengujian performa.
- Pengujian performa dengan JMeter (load, stress, soak, spike) serta pengumpulan metrik server lalu pengumpulan data dalam CSV dan tipe file lain.
- Analisis data kuantitatif dan kualitatif, uji signifikansi statistik, perhitungan effect size, dan penyusunan rekomendasi berdasarkan hasil yang ditemukan.
- Penyusunan laporan tugas akhir (Bab IV–V), revisi dengan pembimbing, dan finalisasi dokumen akhir TA.

Tabel 3. 1. Jadwal Kegiatan Penelitian

No	Kegiatan	Maret				April				Mei				Juni			
	Minggu ke-	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
1	Analisis kebutuhan sistem & skenario pengujian																
2	Perancangan arsitektur sistem, API REST & tRPC, skema basis data																
3	Pembuatan Frontend Web E-commerce																
4	Implementasi tRPC API (router, prosedur, integrasi frontend)																
5	Implementasi REST API (backend, integrasi DB, endpoint e-commerce)																
6	Pengujian fungsional kedua sistem																
7	Pengujian performa dengan JMeter & pengumpulan																

No	Kegiatan	Maret				April				Mei				Juni			
	Minggu ke-	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
	data(REST & tRPC)																
8	Analisis data, evaluasi performa, uji statistik & penyusunan rekomendasi																
9	Penyusunan laporan tugas akhir & revisi																

DAFTAR PUSTAKA

[1] M. M. Mel, "PERAN E-COMMERCE DALAM MENGEMBANGKAN BISNIS DI ERA DIGITAL," *JURNAL DIALOKA: Jurnal Ilmiah Mahasiswa Dakwah dan Komunikasi Islam*, vol. 3, no. 1, pp. 69–84, Jul. 2024, doi: 10.32923/dla.v3i1.4589.

[2] Grand View Research, "E-commerce Market Size, Share And Growth Report, 2030," https://www.grandviewresearch.com/industry-analysis/e-commerce-market. Accessed: Oct. 10, 2025. [Online]. Available: https://www.grandviewresearch.com/industry-analysis/e-commerce-market

[3] R. Prayoga *et al.*, "Analysis Effectiveness Sale Using E-Commerce in the Digital Age: A Case Study on Generation Millennials," *International Journal of Artificial Intelegence Research*, vol. 8, no. 1, 2024, doi: 10.29099/ijair.v8i1.1.1344.

[4] M. Ain, R. Ardiansyah, S. Pratama, M. Akbar, and N. Lapatta, "Comparative Performance Analysis of GRPC and Rest API Under Various Traffic Conditions and Data Sizes Using a Quantitative ApproachComparative Performance Analysis of GRPC and Rest API Under Various Traffic Conditions and Data Sizes Using a Quantitative Approach," *Journal of Applied Informatics and Computing*, vol. 9, pp. 450–457, Oct. 2025, doi: 10.30871/jaic.v9i2.9276.

[5] "The state of gRPC in the browser | gRPC." Accessed: Oct. 10, 2025. [Online]. Available: https://grpc.io/blog/state-of-grpc-web/

[6] "Compare gRPC services with HTTP APIs | Microsoft Learn." Accessed: Oct. 10, 2025. [Online]. Available: https://learn.microsoft.com/en-us/aspnet/core/grpc/comparison?view=aspnetcore-9.0

[7] "2025 State of the API Report," 2025. Accessed: Oct. 11, 2025. [Online]. Available: https://www.postman.com/state-of-api/2025/#ai-native-developers

[8] M. Niswar, R. A. Safruddin, A. Bustamin, and I. Aswad, "Performance evaluation of microservices communication with REST, GraphQL, and

gRPC," *International Journal of Electronics and Telecommunications*, vol. 70, no. 2, pp. 429–436, Jun. 2024, doi: 10.24425/ijet.2024.149562.

[9] R. N. Muzaki and A. Salam, "REDUCING UNDER-FETCHING AND OVER-FETCHING IN REST API WITH GRAPHQL FOR WEB-BASED SOFTWARE DEVELOPMENT," *Jurnal Teknik Informatika (Jutif)*, vol. 5, no. 2, pp. 447–453, Apr. 2024, doi: 10.52436/1.jutif.2024.5.2.1725.

[10] G. Brito and M. T. Valente, "REST vs GraphQL: A Controlled Experiment," *2020 IEEE International Conference on Software Architecture (ICSA)*, pp. 81–91, 2020, [Online]. Available: https://api.semanticscholar.org/CorpusID:212644958

[11] S. Puzhavakath Narayanan, Y. S. Nam, A. Sivakumar, B. Chandrasekaran, B. Maggs, and S. Rao, "Reducing Latency through Page-aware Management of Web Objects by Content Delivery Networks," *Performance Evaluation Review*, vol. 44, no. 1, pp. 89–100, Jun. 2016, doi: 10.1145/2896377.2901472.

[12] F. Quin, D. Weyns, M. Galster, and C. C. Silva, "A/B testing: A systematic literature review," *Journal of Systems and Software*, vol. 211, p. 112011, 2024, doi: https://doi.org/10.1016/j.jss.2024.112011.

[13] S. Addanki, "Architectural Shifts in API Design: The Progression from SOAP to REST and GraphQL."

[14] S. Chandra and A. Farisi, "Comparative Analysis of RESTful, GraphQL, and gRPC APIs: Perfomance Insight from Load and Stress Testing," *Jurnal Sisfokom (Sistem Informasi dan Komputer)*, vol. 14, no. 1, pp. 81–85, Jan. 2025, doi: 10.32736/sisfokom.v14i1.2315.

[15] W. Zhao, F. Hu, J. Wang, T. Shu, and Y. Xu, "A systematic literature review on social commerce: Assessing the past and guiding the future," *Electron Commer Res Appl*, vol. 57, p. 101219, 2023, doi: https://doi.org/10.1016/j.elerap.2022.101219.

[16] F. Sai, Y. Lu, and R. Lin, "Association for Information Systems Association for Information Systems A systematic literature review on impulse buying in e-commerce A systematic literature review on impulse buying in e-

commerce live streaming with co-occurrence map live streaming with co-occurrence map." [Online]. Available: https://aisel.aisnet.org/iceb2024/70

[17] A. Golmohammadi, M. Zhang, and A. Arcuri, "Testing RESTful APIs: A Survey," Dec. 2022, [Online]. Available: http://arxiv.org/abs/2212.14604

[18] Oyekunle Claudius Oyeniran, Adebunmi Okechukwu Adewusi, Adams Gbolahan Adeleke, Lucy Anthony Akwawa, and Chidimma Francisca Azubuko, "Microservices architecture in cloud-native applications: Design patterns and scalability," *Computer Science & IT Research Journal*, vol. 5, no. 9, pp. 2107–2124, Sep. 2024, doi: 10.51594/csitrj.v5i9.1554.

[19] S. Khlamov, M. Mendielieva, O. Vovk, and Z. Deineko, "Comparative Analysis of Jmeter and Postman for API-Based Performance Testing," 2025.

[20] "tRPC | tRPC." Accessed: Nov. 03, 2025. [Online]. Available: https://trpc.io/docs

[21] Y. Rayhan, M. Suhayati, and B. Sutara, "COMPARISON OF REST API AND GRAPHQL IN THE STRAPI CONTENT MANAGEMENT SYSTEM KOMPARASI REST API DAN GRAPHQL DI CONTENT MANAGEMENT SYSTEM STRAPI," vol. 1, no. 2, pp. 120–124, 2024.

[22] "FHIR | Cloud Healthcare API | Google Cloud Documentation." Accessed: Nov. 03, 2025. [Online]. Available: https://docs.cloud.google.com/healthcare-api/docs/concepts/fhir

[23] "SATUSEHAT Dikembangkan Langsung Oleh Kemenkes." Accessed: Nov. 03, 2025. [Online]. Available: https://kemkes.go.id/id/satusehat-dikembangkan-langsung-oleh-kemenkes

[24] H. C. Monga, T. Kinnunen, A. Malhi, A. Javed, and K. Främling, "An OAuth-based Authentication Mechanism for Open Messaging Interface Standard," in *ICAART 2020 - Proceedings of the 12th International Conference on Agents and Artificial Intelligence*, SciTePress, 2020, pp. 216–225. doi: 10.5220/0008970902160225.

[25] P. Rujichaikul and I. Rassameeroj, "Token-Based Authentication Monitoring System," *Journal of Cyber Security and Mobility*, Oct. 2025, doi: 10.13052/jcsm2245-1439.1441.

[26] H. Bagci and A. Kara, “A lightweight and high performance remote procedure call framework for cross platform communication,” in *ICSOFT 2016 - Proceedings of the 11th International Joint Conference on Software Technologies*, SciTePress, 2016, pp. 117–124. doi: 10.5220/0005931201170124.

[27] A. D. Birrell and B. J. Nelson, “Implementing remote procedure calls,” *ACM Trans. Comput. Syst.*, vol. 2, no. 1, pp. 39–59, Feb. 1984, doi: 10.1145/2080.357392.

[28] “Concepts | tRPC.” Accessed: Nov. 02, 2025. [Online]. Available: https://trpc.io/docs/concepts

[29] “Remote Procedure Call - IBM Documentation.” Accessed: Nov. 03, 2025. [Online]. Available: https://www.ibm.com/docs/en/aix/7.2.0?topic=concepts-remote-procedure-call

[30] M. Zemskov, “Reliability of the Type System in TypeScript in Software Development,” *Asian Journal of Research in Computer Science*, vol. 18, no. 2, pp. 50–59, Jan. 2025, doi: 10.9734/ajrcos/2025/v18i2561.

[31] Z. Ghinafikar, M. M. Mu’thy, and M. A. Yaqin, “Perbandingan Metode Agile dan Waterfall Berdasarkan Analisis Waktu Pengembangan Sistem,” *Jurnal Manajemen Teknologi Informatika*, vol. 3, no. 1, pp. 26–44, Apr. 2025, doi: 10.70038/jentik.v3i1.149.

[32] J. Prayoga, B. Solihin Hasugian, and A. Yasir, “Analisis Efektivitas Penerapan Metode Waterfall dan Agile dalam Pengembangan Perangkat Lunak,” 2025. [Online]. Available: https://journals.raskhamedia.or.id/index.php/juiktiDOI:https://doi.org/99.9999/juikti.vxix.xxxx

[33] D. J. Tettey *et al.*, “Agile hybrid methodologies for complex project execution: Balancing flexibility, control, and stakeholder value maximization.,” *International Journal of Research Publication and Reviews*, vol. 6, no. 3, pp. 6190–6204, Mar. 2025, doi: 10.55248/gengpi.6.0325.1295.

[34] N. Koceska and S. Koceski, “Hybrid project management as a new form of

project management."

[35] R. T. Amanda and R. A. Putri, "Application of User-Centered Design Method in E-Commerce Sales System," *SISTEMASI*, vol. 13, no. 3, p. 1295, May 2024, doi: 10.32520/stmsi.v13i3.4145.

[36] Dwi Nurul Huda, L. Sirait, R. Fandhica, and M. R. Romdoni, "Implementation Of User-Centered Design (UCD) Method On The Design Of Real- Time Coffee Shop Reservation Application," *Jurnal Bangkit Indonesia*, vol. 14, no. 1, pp. 41–46, Mar. 2025, doi: 10.52771/bangkitindonesia.v14i1.432.

[37] W. Purbaratri and N. Krisnawaty, "Proceeding of the Perbanas International Seminar on Economics, Business, Management, Accounting and IT (PROFICIENT) 2023 SOFTWARE QUALITY MODEL ISO/IEC 25010 PRIORITIZATION SMART GOVERNMENT APPLICATION."

[38] Elizabeth Atieno Otieno, "Data protection and privacy in e-commerce environment: Systematic review," *GSC Advanced Research and Reviews*, vol. 22, no. 1, pp. 238–271, Jan. 2025, doi: 10.30574/gscarr.2025.22.1.0024.

[39] "React." Accessed: Nov. 16, 2025. [Online]. Available: https://react.dev/

[40] "Next.js Docs | Next.js." Accessed: Nov. 16, 2025. [Online]. Available: https://nextjs.org/docs

[41] "TypeScript: Documentation - The Basics." Accessed: Nov. 16, 2025. [Online]. Available: https://www.typescriptlang.org/docs/handbook/2/basic-types.html

[42] "Static type checking with TypeScript · openedx/paragon · Discussion #1186 · GitHub." Accessed: Nov. 16, 2025. [Online]. Available: https://github.com/openedx/paragon/discussions/1186

[43] F. Pratama and A. Farisi, "Analisis Perbandingan Kinerja Backend API Menggunakan PHP, Golang, dan JavaScript Performance Analysis of Backend API Using PHP, Golang, and Node JS."

[44] R. Irman Hermadi Yani Nurhadryani, "Analisis Uji Performa Aplikasi Dari Hasil Implementasi Refactoring Arsitektur Monolitik Ke Mikroservis dengan Decomposition dan Strangler Pattern," 2023.

[45] M. Neelapu and A. Pub, “Enhancement of Software reliability using Automatic API Testing Model,” *International Journal of Multidisciplinary Research and Growth Evaluation*, vol. 04, pp. 1113–1117, May 2023.

LAMPIRAN

Lampiran 1. Turnitin Originality Report Halaman 1

NonMagang_220711833_SDE_Andi.docx

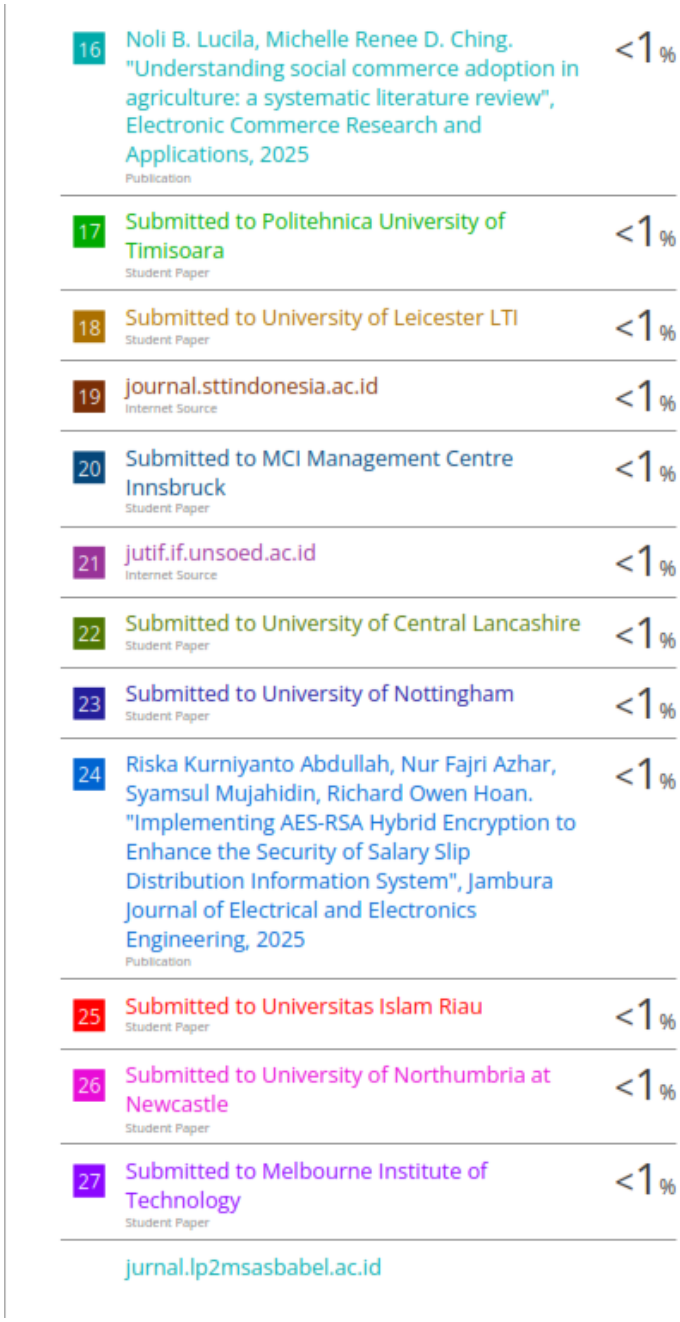
ORIGINALITY REPORT

9%	8%	5%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

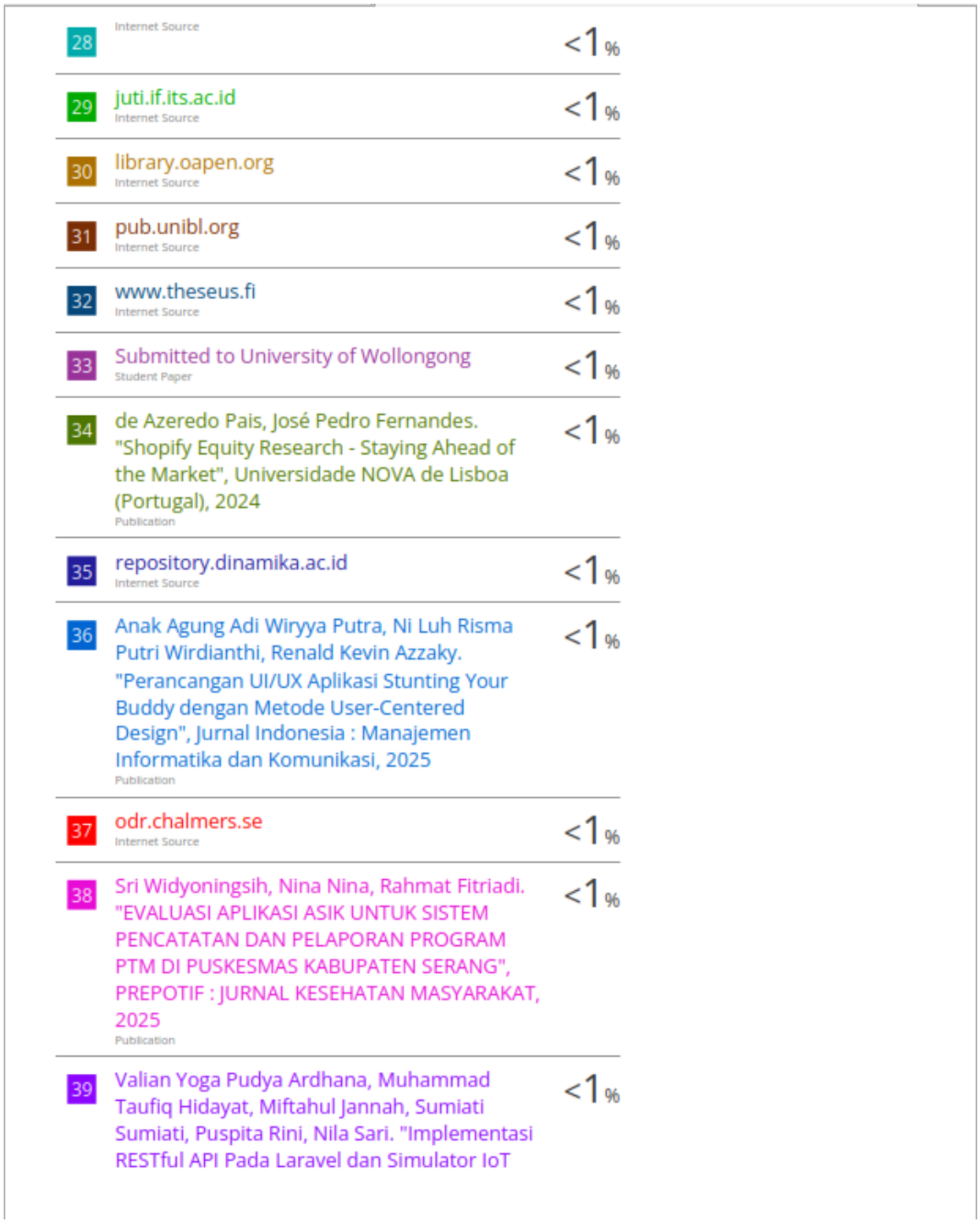
1	e-journal.uajy.ac.id Internet Source	1%
2	jurnal.polibatam.ac.id Internet Source	1%
3	Submitted to Universitas Putera Batam Student Paper	<1%
4	jurnal.kdi.or.id Internet Source	<1%
5	Submitted to Cardiff University Student Paper	<1%
6	artemis.cslab.ece.ntua.gr:8080 Internet Source	<1%
7	Submitted to Universitas Brawijaya Student Paper	<1%
8	risaboviaddesitaviani.blogspot.com Internet Source	<1%
9	www.ijsr.net Internet Source	<1%
10	text-id.123dok.com Internet Source	<1%
11	uis.brage.unit.no Internet Source	<1%
12	ojs.trigunadharma.ac.id Internet Source	<1%
13	tel.archives-ouvertes.fr Internet Source	<1%
14	Submitted to Victorian Institute of Technology Student Paper	<1%
15	www.diva-portal.org Internet Source	<1%

Lampiran 2. Turnitin Originality Report Halaman 2



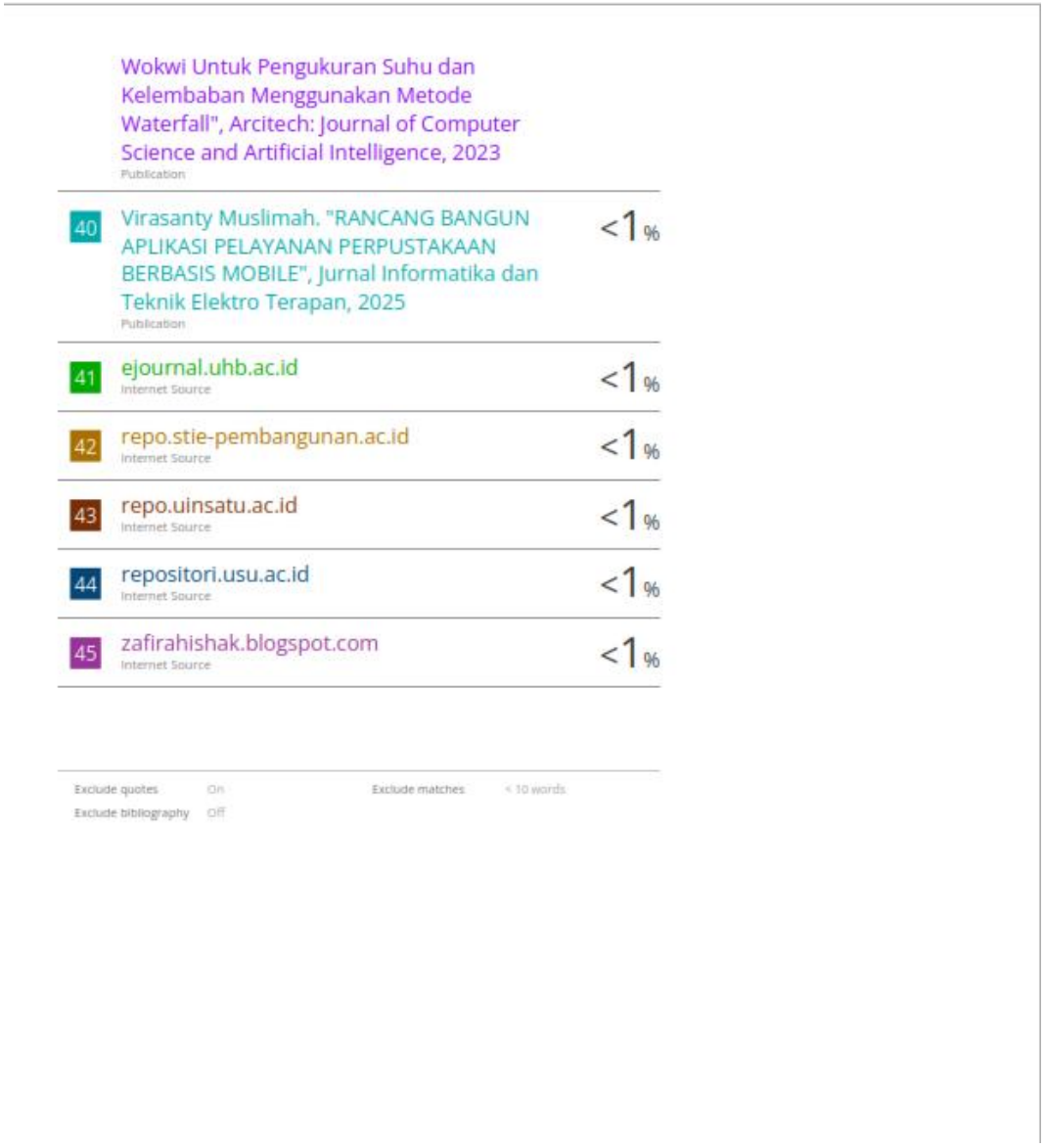
16	Noli B. Lucila, Michelle Renee D. Ching. "Understanding social commerce adoption in agriculture: a systematic literature review", Electronic Commerce Research and Applications, 2025 Publication	<1%
17	Submitted to Politehnica University of Timisoara Student Paper	<1%
18	Submitted to University of Leicester LTI Student Paper	<1%
19	journal.sttindonesia.ac.id Internet Source	<1%
20	Submitted to MCI Management Centre Innsbruck Student Paper	<1%
21	jutif.if.unsoed.ac.id Internet Source	<1%
22	Submitted to University of Central Lancashire Student Paper	<1%
23	Submitted to University of Nottingham Student Paper	<1%
24	Riska Kurniyanto Abdullah, Nur Fajri Azhar, Syamsul Mujahidin, Richard Owen Hoan. "Implementing AES-RSA Hybrid Encryption to Enhance the Security of Salary Slip Distribution Information System", Jambura Journal of Electrical and Electronics Engineering, 2025 Publication	<1%
25	Submitted to Universitas Islam Riau Student Paper	<1%
26	Submitted to University of Northumbria at Newcastle Student Paper	<1%
27	Submitted to Melbourne Institute of Technology Student Paper	<1%
	jurnal.lp2msasbabel.ac.id	

Lampiran 3. Turnitin Originality Report Halaman 3



28	Internet Source	<1%
29	juti.if.its.ac.id Internet Source	<1%
30	library.oapen.org Internet Source	<1%
31	pub.unibl.org Internet Source	<1%
32	www.theseus.fi Internet Source	<1%
33	Submitted to University of Wollongong Student Paper	<1%
34	de Azeredo Pais, José Pedro Fernandes. "Shopify Equity Research - Staying Ahead of the Market", Universidade NOVA de Lisboa (Portugal), 2024 Publication	<1%
35	repository.dinamika.ac.id Internet Source	<1%
36	Anak Agung Adi Wiryya Putra, Ni Luh Risma Putri Wirdianthi, Renald Kevin Azzaky. "Perancangan UI/UX Aplikasi Stunting Your Buddy dengan Metode User-Centered Design", Jurnal Indonesia : Manajemen Informatika dan Komunikasi, 2025 Publication	<1%
37	odr.chalmers.se Internet Source	<1%
38	Sri Widyoningsih, Nina Nina, Rahmat Fitriadi. "EVALUASI APLIKASI ASIK UNTUK SISTEM PENCATATAN DAN PELAPORAN PROGRAM PTM DI PUSKESMAS KABUPATEN SERANG", PREPOTIF : JURNAL KESEHATAN MASYARAKAT, 2025 Publication	<1%
39	Valian Yoga Pudya Ardhana, Muhammad Taufiq Hidayat, Miftahul Jannah, Sumiati Sumiati, Puspita Rini, Nila Sari. "Implementasi RESTful API Pada Laravel dan Simulator IoT	<1%

Lampiran 4. Turnitin Originality Report Halaman 4



Wokwi Untuk Pengukuran Suhu dan Kelembaban Menggunakan Metode Waterfall", Arcitech: Journal of Computer Science and Artificial Intelligence, 2023
Publication

40	Virasanty Muslimah. "RANCANG BANGUN APLIKASI PELAYANAN PERPUSTAKAAN BERBASIS MOBILE", Jurnal Informatika dan Teknik Elektro Terapan, 2025 Publication	<1 %
41	ejournal.uhb.ac.id Internet Source	<1 %
42	repo.stie-pembangunan.ac.id Internet Source	<1 %
43	repo.uinsatu.ac.id Internet Source	<1 %
44	repositori.usu.ac.id Internet Source	<1 %
45	zafirahishak.blogspot.com Internet Source	<1 %

Exclude quotes On
Exclude bibliography Off
Exclude matches < 10 words

Lampiran 5. Turnitin Digital Receipt

turnitin

Digital Receipt

This receipt acknowledges that Turnitin received your paper. Below you will find the receipt information regarding your submission.

The first page of your submissions is displayed below.

Submission author:	Angello Sitanggang
Assignment title:	UAJY Student
Submission title:	NonMagang_220711833_SDE_Andi.docx
File name:	NonMagang_220711833_SDE_Andi.docx
File size:	1.89M
Page count:	59
Word count:	10,667
Character count:	71,556
Submission date:	20-Nov-2025 09:08PM (UTC+0700)
Submission ID:	2821849994

PROPOSAL TUGAS AKHIR

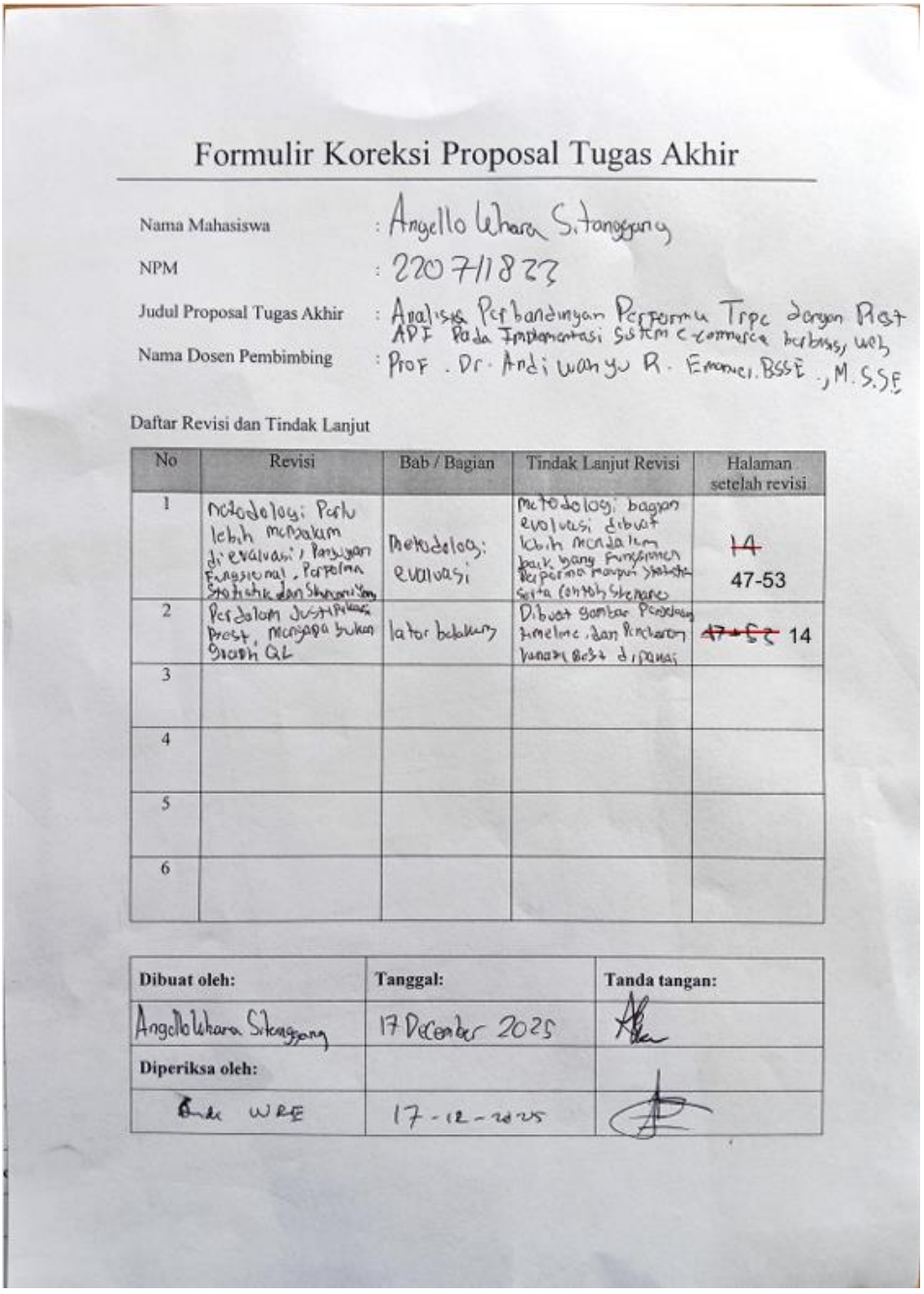
Software Development

Analisis Perbandingan Performa TRPC dengan REST API pada Implementasi Sistem E-Commerce Berbasis Web

Diajukan untuk Memenuhi Salah Satu Persyaratan Mencapai Derajat Sarjana Komputer

PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ATMA JAYA YOGYAKARTA
2025

Lampiran 6. Form Persetujuan Revisi



Formulir Koreksi Proposal Tugas Akhir

Nama Mahasiswa : Angello Wahara S. Tanggang

NPM : 2207711827

Judul Proposal Tugas Akhir : Analisis Perbandingan Performa Tipe Layan Rest API Pada Implementasi Sistem E-commerce berbasis web

Nama Dosen Pembimbing : Prof. Dr. Andi Wahyu R. Emanuel, BSSE., M.S.E

Daftar Revisi dan Tindak Lanjut

No	Revisi	Bab / Bagian	Tindak Lanjut Revisi	Halaman setelah revisi
1	Metodelogi: Perlu lebih mendalam di evaluasi / Penjelasan fungsional, Performa Statistik dan Skenarionya	Metodelogi: evaluasi	Metodelogi: bagian evaluasi dibuat lebih mendalam baik yang fungsional performa maupun statistik serta contoh skenario	~~14~~ 47-53
2	Perjelas dalam justifikasi Rest, mengapa bukan Graph QL	latar belakang	Dibuat gambar Perbedaan timeline, dan Penjelasan kenapa Rest di pakai	~~17 → 52~~ 14
3				
4				
5				
6				

Dibuat oleh:	Tanggal:	Tanda tangan:
Angello Wahara Sitanggang	17 Desember 2025	
Diperiksa oleh:		
WRE	17-12-2025	