BAB I
PENDAHULUAN

1.1 Latar Belakang

Pengembangan sistem informasi mencakup cara distribusi data yang diperlukan untuk mendukung pembuatan keputusan. Setiap sistem informasi memiliki metode distribusi data yang berbeda, mulai dari menggabungkan bagian yang menampilkan informasi kepada pengguna (*frontend*) dan bagian yang mengelola pemrosesan data (*backend*), dengan menggunakan layanan distribusi data seperti REST API untuk sistem dengan struktur *frontend* dan *backend* terpisah. Selama dua dekade terakhir, *Representational State Transfer* (REST) telah menjadi standar arsitektur dalam pengembangan *Application Programming Interfaces* (APIs) untuk aplikasi *backend* dan layanan web *server-side*. [1]. Dengan pesatnya perkembangan teknologi di era ini, tidak mengherankan jika terjadi kemajuan dalam teknologi pendistribusian data. Facebook, dalam hal ini, telah mengembangkan teknologi baru untuk pendistribusian data dengan menerapkan standar arsitektur terbaru dalam pembuatan API, yaitu GraphQL.

REST API memiliki berbagai aturan dan batasan yang menyeluruh, yang memungkinkan layanan web untuk mengakses sumber data secara terstruktur. Namun, aturan ini bisa menjadi kurang fleksibel dan kompleks karena beberapa alasan, termasuk: (1) Peningkatan kompleksitas sistem yang dikembangkan, (2) Kebutuhan akan kualitas layanan yang tinggi dari pengguna, (3) Pengembangan sistem secara real-time, dan (4) Permintaan data dinamis oleh klien dari *frontend* maupun perangkat *mobile* [2]. GraphQL dapat didefinisikan sebagai bahasa *query* yang memungkinkan permintaan data dari database, baik dari sisi klien (*frontend*) maupun server (*backend*). Di sisi *backend*, GraphQL memungkinkan penentuan data apa saja yang akan dikembalikan kepada klien saat klien meminta data. Selain itu, GraphQL dirancang untuk membuat API lebih cepat, fleksibel, dan memudahkan pengembang dalam proses pengembangan. [3]. Sejak GraphQL diluncurkan pada tahun 2015, GraphQL telah mendapatkan popularitas di kalangan pengembang, menjadikannya salah satu arsitektur yang semakin banyak digunakan. Keuntungan yang ditawarkan oleh GraphQL, terutama dalam mengatasi masalah yang tidak dapat dipecahkan oleh arsitektur REST, turut berkontribusi pada pesatnya peningkatan popularitasnya.

Dalam pengembangan sistem, pengembang sering menghadapi berbagai masalah yang perlu diselesaikan dengan cepat untuk memberikan layanan terbaik kepada pengguna. Untuk itu,

pengembang membandingkan dua arsitektur API, yaitu REST dan GraphQL, dengan tujuan untuk mengevaluasi efisiensi masing-masing arsitektur. Analisis perbandingan ini melibatkan beberapa aspek, termasuk kecepatan waktu respons dan ukuran respons dari kedua arsitektur. Peneliti juga akan membandingkan kompleksitas bahasa *query* yang digunakan dalam proses pengambilan data serta skalabilitas kedua arsitektur dalam menangani aplikasi dengan cakupan yang lebih luas. Sebagai contoh, perusahaan *hospitality service* di Bali memanfaatkan REST dan GraphQL untuk mendukung pengembangan aplikasi keuangan perusahaan tersebut.

Seperti hal nya penelitian yang telah dilakukan oleh [1] dua gagasan utama digunakan untuk membandingkan kinerja kedua arsitektur. Pertama, evaluasi dilakukan pada layanan yang beroperasi dengan transaksi nyata dalam sistem informasi manajemen, di mana transaksi besar dan intensif mempengaruhi database kompleks dengan banyak hubungan. Kedua, kinerja diuji secara adil dan independen dengan mendistribusikan permintaan klien dan menyinkronkan respons layanan melalui dua jalur eksekusi paralel untuk setiap API. Evaluasi kinerja dilakukan menggunakan ukuran QoS (*Quality of Services*) dasar, seperti waktu respons, *throughput*, beban CPU, dan penggunaan memori. Hasilnya menunjukkan bahwa REST lebih cepat hingga 50,50% dalam waktu respons dan 37,16% dalam *throughput*, sementara GraphQL lebih efisien dalam penggunaan sumber daya dengan 37,26% untuk beban CPU dan 39,74% untuk penggunaan memori. Dengan demikian, GraphQL lebih tepat digunakan ketika kebutuhan data sering berubah dan efisiensi sumber daya menjadi faktor utama.

Penelitian ini bertujuan untuk membantu pengembang dalam memilih arsitektur yang tepat untuk sistem informasi keuangan, khususnya di perusahaan layanan perhotelan di Bali. Selain itu, penelitian ini berfokus pada penentuan arsitektur yang paling efisien berdasarkan kegunaan, desain arsitektur, kebutuhan sumber daya, waktu, biaya, dan skalabilitas perangkat lunak di masa depan. Penelitian ini juga memberikan analisis terhadap kedua arsitektur untuk membantu pengembang dan peneliti di masa depan dalam memaksimalkan performa *web service*. Hasil yang diharapkan adalah identifikasi metode dengan arsitektur yang optimal, menilai efisiensi dalam hal kegunaan, sumber daya, waktu, biaya, dan penggunaan memori. Dengan temuan ini, diharapkan dapat memberikan solusi bagi pengembang dalam menentukan arsitektur yang tepat untuk membangun sistem informasi di masa mendatang.

1.2 Rumusan Masalah

Berdasarkan latar belakang dan permasalahan yang telah dipaparkan peneliti akan melakukan perbandingan kedua arsitektur antara REST dan GraphQL berdasarkan masalah berikut ini:

1. Bagaimana perbandingan waktu respon dan ukuran respon dari arsitektur REST dan GraphQL saat melakukan *fetching* data, menggunakan pengujian dengan K6.
2. Bagaimana perbandingan skalabilitas dari arsitektur REST dan GraphQL saat aplikasi bertambah menjadi semakin besar, berdasarkan pengujian beban yang dilakukan dengan K6.

1.3 Tujuan Penelitian

Berdasarkan rumusan masalah yang telah dipaparkan penelitian ini memiliki tujuan sebagai berikut:

1. Menganalisa perbandingan waktu respon dan ukuran respon dari arsitektur REST dan GraphQL saat melakukan *fetching* data. Pengujian dilakukan menggunakan K6, sebuah alat uji beban yang memungkinkan simulasi permintaan dan pengukuran waktu respons serta ukuran respons secara efisien.
2. Menganalisa perbandingan skalabilitas dari arsitektur REST dan GraphQL saat aplikasi bertambah semakin besar. Pengujian ini juga dilakukan dengan K6, untuk mengevaluasi bagaimana kedua arsitektur menangani peningkatan jumlah permintaan dan kompleksitas aplikasi yang berkembang.

1.4 Batasan Masalah

Batasan masalah ini bertujuan agar penelitian yang dilakukan tidak melebar pada fokusnya, maka batas penelitian sebagai berikut:

1. Bahasa pemrograman yang akan digunakan adalah Typescript.
2. Runtime yang digunakan adalah Node.JS.
3. Web Framework yang digunakan adalah Express.JS.
4. ORM (*Object Relational Mapping*) yang digunakan adalah Prisma ORM.
5. Aplikasi yang dikembangan adalah membangun sebuah aplikasi perusahaan *hospitality service*. Aplikasi aplikasi perusahaan *hospitality service* yang akan dibangun menggunakan arsitektur REST dan arsitektur GraphQL.