universitas
MALIKUSSALEH

PROPOSAL SKRIPSI

IMPLEMENTASI HPACK DAN GZIP PADA OPTIMASI APLIKASI DENGAN ARSITEKTUR MICROSERVICE

Disusun Sebagai Syarat Memperoleh Gelar Sarjana Teknik
Jurusan Teknik Informatika Fakultas Teknik
Universitas Malikussaleh

DISUSUN OLEH :

NAMA	**:**	**SYIBBRAN MULAESYI**
NIM	**:**	**190170067**
PRODI	**:**	**TEKNIK INFORMATIKA**

JURUSAN INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MALIKUSSALEH
LHOKSEUMAWE
2023

LEMBAR PENGESAHAN PROPOSAL

IMPLEMENTASI HPACK DAN GZIP PADA OPTIMASI APLIKASI DENGAN ARSITEKTUR MICROSERVICE

PROPOSAL

Oleh:

NAMA: SYIBBRAN MULAESYI

NIM: 190170067

PRODI: TEKNIK INFORMATIKA

Telah disetujui oleh:

Pembimbing Utama	Pembimbing Pendamping
Rizal, S.Si., M.IT	**Zara Yunizar, S.Kom., M.Kom**
NIP. 197811052006041013	**NIP. 198310182019032009**

Mengetahui,

Ketua Jurusan Informatika

Munirul Ula, S.T., M.Eng., Ph.D

NIP. 197808082008121001

KATA PENGANTAR

Puji syukur ke hadirat Tuhan Yang Maha Esa, karena berkat rahmat dan hidayah-Nya, saya dapat menyelesaikan penelitian tugas akhir ini dengan baik.

Penyusunan penelitian tugas akhir ini merupakan salah satu syarat untuk menyelesaikan program pendidikan saya di Universitas Malikussaleh. Tugas akhir ini berjudul "Implementasi HPack Dan Gzip Pada Optimasi Aplikasi Dengan Arsitektur microservice".

Dalam tugas akhir ini, saya akan menjelaskan tentang Implementasi HPack Dan Gzip Pada Optimasi Aplikasi Dengan Arsitektur microservice. Implementasi kedua algoritma tersebut diharapkan dapat mempercepat proses pengiriman dan penerimaan data antar aplikasi.

Penulisan tugas akhir ini tidak lepas dari bantuan dan dukungan dari berbagai pihak yang telah memberikan dukungan dan bimbingan selama proses penyusunan tugas akhir ini. Oleh karena itu, saya ingin menyampaikan terima kasih kepada :

1. Bapak Prof. Dr. Ir. Herman Fithra, M.T., IPM., ASEAN Eng., selaku Rektor Universitas Malikussaleh.
2. Bapak Dr. Muhammad, S.T., M.Sc selaku Dekan Fakultas Teknik Universitas Malikussaleh
3. Bapak Munirul Ula, S.T., M.Eng., Ph.D selaku Ketua Jurusan Informatika Universitas Malikussaleh.
4. Ibu Zara Yunizar, S.Kom., M.Kom selaku Ketua Prodi Teknik Informatika
5. Bapak Rizal, S.Si., M.IT selaku Dosen pembimbing Utama
6. Ibu Zara Yunizar, S.Kom., M.Kom selaku Dosen pembimbing Pendamping
7. ... selaku Dosen Penguji utama dan Penguji Pendamping
8. Bapak dan ibu dosen serta staf akademik yang telah membantu penulis selama mengikuti perkuliahan di Program Studi Teknik Informatika Universitas Malikussaleh
9. Kepada Orang tua penulis Terima kasih atas segala doa, dukungan dan kasih sayang yang diberikan

10. kepada penulis, sehingga penulis dapat menyelesaikan tugas akhir ini.
11. Semua pihak yang telah membantu penulis dalam menyelesaikan Penelitian ini.
12. Penulis menyadari kalau ilmu serta pengalaman yang penulis miliki belum sempurna.

Sebagai tugas akhir, penulisan ini masih jauh dari sempurna. Oleh karena itu, saya sangat mengharapkan kritik dan saran dari para pembaca agar tugas akhir ini dapat diperbaiki dan menjadi lebih baik lagi.

Akhir kata, saya berharap tugas akhir ini dapat bermanfaat bagi pembaca dan dapat menjadi bahan referensi yang baik dalam bidang optimasi Aplikasi Dengan Arsitektur microservice.

Lhokseumawe, 11 Februari 2023

Penulis,

Syibbran Mulaesyi

190170067

DAFTAR ISI

LEMBAR PENGESAHAN PROPOSAL ..... **I**
KATA PENGANTAR ..... **II**
DAFTAR ISI ..... **IV**
DAFTAR TABEL ..... **VII**
DAFTAR GAMBAR ..... **VIII**
DAFTAR LAMPIRAN ..... **IX**
BAB I PENDAHULUAN **1**
1.1 Latar Belakang 1
1.2 Rumusan Masalah 4
1.3 Tujuan Penelitian 4
1.4 Manfaat Penelitian 4
1.5 Ruang Lingkup Dan Batasan Penelitian 5
BAB II TINJAUAN KEPUSTAKAAN **7**
2.1 Arsitektur microservice 7
2.1.1 Definisi dan Karakteristik dari Arsitektur microservice 7
2.1.2 Kelebihan dan Kekurangan Arsitektur microservice 7
2.1.3 Implementasi Arsitektur microservice pada Aplikasi 8
2.2 Optimasi Aplikasi 9
2.2.1 Definisi Optimasi Aplikasi 9
2.2.2 Jenis-jenis Optimasi Aplikasi 10
2.2.3 Manfaat Optimasi Aplikasi 10
2.2.4 Strategi Optimasi Aplikasi 11
2.3 HPack 12
2.3.1 Definisi HPack 12
2.3.2 Fungsi dan Kelebihan HPack 12
2.3.3 Cara Kerja HPack 12
2.4 Gzip 14
2.4.1 Cara Kerja Gzip 14

2.4.2 Keuntungan Menggunakan Gzip 15
2.5 JSON(Javascript Object Notation) 15
2.5.1 Definisi dan Struktur JSON 15
2.5.2 Keunggulan JSON 15
2.5.3 Implementasi JSON 16
2.6 Penelitian Terdahulu 16
BAB III METODE PENELITIAN 20
3.1 Tahapan Pengembangan 20
3.2 Analisis kebutuhan 21
3.2.1 Modul kompresi dekompresi JSON 22
3.2.2 Microservice Social Media 22
3.2.3 Frontend 22
3.3 Desain Sistem 24
3.3.1 Modul kompresi dekompresi JSON 25
3.3.2 Microservice Social Media 26
3.3.3 Frontend 35
3.4 Implementasi 36
3.4.1 Overview Implementasi 37
3.4.2 *Detil Implementasi* 37
3.4.3 *Proses Implementasi* 37
3.5 Deployment 38
3.5.1 Pra-deployment 38
3.5.2 Deployment 38
3.5.3 Pasca-deployment 38
3.6 Pengujian 38
3.6.1 Metodologi Pengujian 39
3.6.2 Alat Pengujian 39
3.6.3 Hasil Pengujian 39
3.6.4 Rancangan Analisis *Hasil* 39
DAFTAR PUSTAKA 40

DAFTAR TABEL

Tabel 2.1 Penelitian Terdahulu 18
Tabel 3.1 Data Analisa Kebutuhan 22
Tabel 3.2 Definisi API Autentikasi 27
Tabel 3.3 Definisi Tabel User 27
Tabel 3.4 Definisi API Pengguna 28
Tabel 3.5 Definisi Tabel UserProfile 28
Tabel 3.6 Definisi Tabel UserProfile 29
Tabel 3.7 Definisi Tabel Friend 29
Tabel 3.8 Definisi API Post 29
Tabel 3.9 Definisi Tabel Post 30
Tabel 3.10 Definisi Tabel PostInteraction 31
Tabel 3.11 Definisi Tabel Comment 31
Tabel 3.12 Definisi Tabel Like 31
Tabel 3.13 Definisi API Tag 32
Tabel 3.14 Definisi Tabel Tag 32
Tabel 3.15 Definisi Tabel PostTag 32
Tabel 3.16 Definisi API Analitik Pengguna 33
Tabel 3.17 Definisi Tabel LoginRecord 33
Tabel 3.18 Definisi Tabel SearchRecord 33
Tabel 3.19 Definisi API Analitik Post 34
Tabel 3.20 Definisi Tabel PostView 34
Tabel 3.21 Definisi Tabel PostLike 34
Tabel 3.22 Definisi Tabel PostComment 35
Tabel 3.23 Definisi API Post 35

DAFTAR GAMBAR

Gambar 3.1 Bagan Tahapan Penelitian 20
Gambar 3.2 Proses Kerja Modul 25

DAFTAR LAMPIRAN

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam era digital saat ini, kita tidak bisa lepas dari aplikasi web. Hampir setiap aktivitas kita sehari-hari, baik itu bekerja, belanja, hiburan, atau berkomunikasi dilakukan melalui aplikasi web. Seiring dengan perkembangan teknologi, aplikasi web pun semakin berkembang pesat. Mulai dari aplikasi sederhana seperti media sosial, hingga aplikasi kompleks seperti e-commerce, semuanya menggunakan berbagai jenis arsitektur perangkat lunak yang berbeda-beda.

Salah satu arsitektur perangkat lunak yang sedang populer saat ini adalah arsitektur microservice. Microservice itu sendiri adalah sebuah proses kohesif dan independen yang saling berinteraksi melalui pesan (Dragoni dkk., 2017). Dengan kata lain Microservice adalah pendekatan dalam pengembangan perangkat lunak yang memecah sebuah aplikasi menjadi bagian-bagian yang lebih kecil dan mandiri. Setiap bagian tersebut memiliki tugas dan fungsi yang spesifik, dan dapat berkomunikasi dengan bagian lainnya melalui antarmuka yang terdefinisi dengan jelas (Lewis & Fowler, 2014). Sedangkan arsitektur microservice adalah aplikasi terdistribusi di mana semua modulnya adalah microservice (Dragoni dkk., 2017).

Arsitektur microservice memiliki beberapa kelebihan dibandingkan dengan arsitektur monolitik, di mana seluruh aplikasi dibangun dalam satu kesatuan. Pertama, arsitektur microservice memungkinkan pengembang untuk bekerja secara mandiri pada bagian-bagian tertentu dari aplikasi tanpa harus mempengaruhi bagian lainnya (Richards, 2022). Hal ini memungkinkan pengembangan dan perbaikan aplikasi menjadi lebih cepat dan efisien.

Selain itu, arsitektur microservice juga memungkinkan skalabilitas yang lebih baik. Dalam arsitektur monolitik, jika terdapat peningkatan jumlah pengguna atau permintaan, maka seluruh aplikasi harus ditingkatkan, bahkan jika hanya bagian tertentu yang membutuhkan peningkatan. Dalam arsitektur microservice,

bagian-bagian yang membutuhkan peningkatan dapat ditingkatkan secara mandiri, sehingga aplikasi dapat lebih efisien dan responsif.

Salah satu kekurangan dari arsitektur microservice yang perlu dicermati adalah kompleksitas dalam manajemen infrastruktur dan koordinasi antar bagian aplikasi(Richards, 2022). Sebagai contoh, ketika sebuah aplikasi web menggunakan arsitektur microservice, maka setiap service harus berkomunikasi dengan service lainnya untuk memastikan semua bagian aplikasi dapat berjalan dengan lancar dan saling terintegrasi dengan baik. Ketika data yang ditransfer antar service semakin besar, maka bisa terjadi *bottleneck* dalam komunikasi antar service tersebut. Hal ini dapat memperlambat respons waktu aplikasi, bahkan dapat menyebabkan kegagalan dalam operasional aplikasi.

Namun, kompleksitas dalam manajemen infrastruktur dan koordinasi antar bagian aplikasi dalam arsitektur microservice tidak selalu menghasilkan performa yang optimal (Richardson, 2018). Seperti yang telah dijelaskan sebelumnya, ketika data yang ditransfer antar service semakin besar, maka bisa terjadi bottleneck dalam komunikasi antar service tersebut. Salah satu solusi untuk mengatasi masalah ini adalah dengan mengoptimalkan proses komunikasi antar service dalam arsitektur microservice. Salah satu cara yang dapat dilakukan adalah dengan mengurangi ukuran data yang ditransfer antar service dengan mengkompresinya. Pada umumnya format data yang sering digunakan dalam komunikasi antar service dalam arsitektur microservice adalah JSON (Salah dkk., 2017). JSON merupakan format data yang ringan, mudah dibaca dan dipahami oleh manusia, dan juga mudah diinterpretasikan oleh mesin (Bray, 2017). Namun, meskipun JSON tergolong ringan, ukuran data yang dikirimkan antar service dapat tetap besar ketika jumlah data yang ditransfer semakin banyak. Contoh nya pada json array dimana sebuah array memiliki beberapa JSON object yang semua nya memiliki key yang sama.

Metode kompresi yang dapat mengkopresi json object yang pertama adalah menggunakan metode HPack. HPack adalah protokol kompresi header HTTP yang dirancang khusus untuk digunakan dalam HTTP/2. HPack adalah sebuah motode kompresi yang menghilangkan *header redundant*, membatasi kerentanan terhadap

serangan keamanan, dan memiliki persyaratan memori terbatas untuk digunakan dalam lingkungan yang terbatas (Peon & Ruellan, 2015).

HPack dapat di gunakan untuk mengatasi masalah komunikasi antar service, di mana jumlah request yang dilakukan oleh browser dalam satu waktu bisa sangat besar. Dalam situasi seperti ini, jika response dikirimkan dalam format yang tidak terkompresi, maka ukuran data yang dikirimkan dapat menjadi sangat besar. HPack pada penelitian ini bertujuan untuk mengurangi ukuran data response dengan cara mengeliminasi redundansi dalam JSON Object dan mengurangi ukuran index yang digunakan untuk mereferensikan kembali response yang sama.

Salah satu metode kompresi data yang populer adalah Gzip. Gzip merupakan metode kompresi data yang mengurangi ukuran data dengan cara menghilangkan redundansi dalam data. Dengan menggunakan Gzip, ukuran data yang dikirimkan antar service dapat dikurangi hingga ~95% (OBJELEAN, 2011). Hal ini dapat mengurangi beban komunikasi antar service dalam arsitektur microservice, sehingga performa aplikasi dapat meningkat.

Dalam konteks implementasi HPack dan Gzip pada optimasi aplikasi dengan arsitektur microservice, penggunaan HPack dan Gzip dapat membantu mengoptimalkan proses komunikasi antar service dengan mengurangi ukuran data response yang dikirimkan, sehingga dapat mengurangi beban komunikasi antar service dalam arsitektur microservice dan meningkatkan performa aplikasi.

Dalam penelitian sebelumnya disimpulkan bahwa HPack yang saat di kombinasikan dengan Gzip untuk mengkopresi JSON Object dapat mengurangi ukuran JSON sebanyak ~97% (OBJELEAN, 2011).

Topik ini sangat menarik untuk diteliti karena masih sedikit peneliti yang membahas masalah ini. Selain itu, HPack juga belum banyak digunakan dan belum banyak dibahas tentang metode ini. Selain itu, belum ada peneliti yang mengkombinasikan arsitektur microservice dan metode kompresi untuk tujuan optimasi.

Dari penjelasan di atas, penulis akan membahas topik "Penerapan HPack dan Gzip dalam Optimalisasi Aplikasi dengan Arsitektur microservice" dalam penelitian ini.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, maka yang menjadi rumusan masalah dalam penelitian ini adalah :

1. Bagaimana implementasi dari Hpack dan Gzip dalam mengkompresi ukuran response pada arsitektur microservice?
2. Apakah kompresi response dari microservice dapat mengoptimasi aplikasi berbasis arsitektur microservice?

1.3 Tujuan Penelitian

Berdasarkan latar belakang dan rumusan masalah yang telah dijelaskan sebelumnya, tujuan dari penelitian ini adalah untuk mengoptimalkan proses komunikasi antar *service* dalam arsitektur microservice dengan menggunakan metode kompresi data. Penelitian ini akan memfokuskan pada implementasi dan evaluasi performa metode kompresi data Gzip dan HPack pada komunikasi antar *service* dalam aplikasi web berbasis arsitektur microservice yang menggunakan format data JSON. Tujuan dari penelitian ini adalah untuk memperoleh hasil evaluasi kinerja dari metode kompresi data Gzip dan HPack dalam mengoptimalkan proses komunikasi antar *service* dalam arsitektur microservice, sehingga dapat memberikan rekomendasi terbaik mengenai metode kompresi data yang paling sesuai digunakan dalam lingkungan arsitektur microservice dengan format data JSON.

1.4 Manfaat Penelitian

Beberapa keuntungan yang dapat diperoleh dari penelitian ini adalah:

1. Memberikan pemahaman lebih mendalam tentang penggunaan arsitektur microservice dalam pengembangan aplikasi web, khususnya dalam hal manajemen infrastruktur dan koordinasi antar bagian aplikasi.
2. Memberikan solusi dalam mengoptimalkan proses komunikasi antar service dalam arsitektur microservice dengan mengurangi ukuran data yang ditransfer antar *service* dengan menggunakan metode kompresi data, seperti Gzip dan HPack.

3. Meningkatkan performa aplikasi web yang menggunakan arsitektur microservice dengan mengurangi beban komunikasi antar *service* melalui penggunaan metode kompresi data.
4. Memberikan rekomendasi kepada pengembang aplikasi web dalam pemilihan metode kompresi data yang tepat sesuai dengan kebutuhan dan karakteristik aplikasi yang dikembangkan.
5. Menjadi referensi dan acuan bagi penelitian selanjutnya yang terkait dengan penggunaan arsitektur microservice dan metode kompresi data dalam pengembangan aplikasi web.

1.5 Ruang Lingkup Dan Batasan Penelitian

Penelitian ini akan fokus pada penerapan HPack dan Gzip dalam optimalisasi aplikasi dengan arsitektur microservice. Ruang lingkup penelitian meliputi:

1. Analisis terhadap teknik kompresi data JSON, khususnya HPack dan Gzip.
2. Evaluasi efektivitas HPack dan Gzip dalam mengurangi ukuran data yang ditransfer antar *service* pada arsitektur microservice.
3. Studi kasus pada aplikasi yang menggunakan arsitektur microservice untuk menunjukkan penerapan dan dampak penggunaan HPack dan Gzip.

Sebagai bagian dari ruang lingkup penelitian, penelitian ini akan memfokuskan pada analisis dan evaluasi terhadap teknik kompresi data JSON, khususnya HPack dan Gzip, dalam rangka mengoptimalkan kinerja aplikasi dengan arsitektur microservice. Namun, ada beberapa batasan dalam penelitian ini yang perlu diperhatikan, yaitu:

1. Penelitian terbatas pada metode kompresi HPack dan Gzip; metode kompresi lainnya tidak akan dibahas secara detail.
2. Penelitian ini tidak akan membahas secara mendalam mengenai aspek keamanan yang terkait dengan penggunaan HPack dan Gzip.
3. Penelitian ini hanya membahas optimasi melalui kompresi JSON *response* atau *request*.

4. Penelitian hanya akan mencakup aplikasi yang menggunakan arsitektur microservice; aplikasi dengan arsitektur lain, seperti monolitik, tidak akan dibahas di dalam penelitian ini.

BAB II

TINJAUAN KEPUSTAKAAN

2.1 Arsitektur microservice

2.1.1 Definisi dan Karakteristik dari Arsitektur microservice

Arsitektur microservice adalah pendekatan yang digunakan dalam pengembangan aplikasi, di mana aplikasi dibagi menjadi beberapa komponen yang lebih kecil dan independen yang disebut " microservice" (Newman, 2021). Setiap microservice bertanggung jawab untuk menjalankan fungsionalitas spesifik dan berkomunikasi dengan microservice lain melalui API yang terdefinisi dengan baik (Lewis & Fowler, 2014). Karakteristik utama dari arsitektur microservice meliputi:

Modularitas: microservice memungkinkan pemisahan fungsionalitas menjadi modul-modul kecil yang independen dan mudah dikelola (Newman, 2015).

Skalabilitas: Setiap microservice dapat diskalakan secara individual sesuai dengan kebutuhan, memungkinkan aplikasi untuk mengakomodasi beban yang beragam (Lewis & Fowler, 2014).

Keberagaman teknologi: microservice memungkinkan penggunaan teknologi yang berbeda dalam satu aplikasi, karena setiap microservice dapat dikembangkan dengan bahasa pemrograman, kerangka kerja, dan basis data yang berbeda (Newman, 2021).

2.1.2 Kelebihan dan Kekurangan Arsitektur microservice

Microservice adalah suatu pendekatan dalam pengembangan perangkat lunak dimana aplikasi dibangun sebagai kumpulan dari layanan yang lebih kecil dan dapat beroperasi secara independen. Pendekatan ini memberikan sejumlah keuntungan yang signifikan, namun tentunya tidak luput dari beberapa tantangan dan kelemahan. Seperti dua sisi mata uang, pendekatan microservice memiliki kelebihan dan kekurangan yang harus dipertimbangkan sebelum memutuskan untuk menerapkannya.

Kelebihan arsitektur microservice meliputi (Lewis & Fowler, 2014; Newman, 2021):

a. Kemudahan pengembangan dan pemeliharaan: microservice memudahkan pengembangan dan pemeliharaan aplikasi dengan memisahkan fungsionalitas menjadi modul yang lebih kecil dan independen.
b. Fleksibilitas dalam pemilihan teknologi: microservice memungkinkan tim untuk menggunakan teknologi yang paling sesuai untuk kebutuhan spesifik mereka.
c. Resiliensi: Kegagalan dalam satu microservice umumnya tidak akan mengakibatkan kegagalan seluruh aplikasi, sehingga meningkatkan keandalan.

Kekurangan arsitektur microservice meliputi (Lewis & Fowler, 2014; Newman, 2021):

a. Kompleksitas: Mengelola dan mengkoordinasikan banyak microservice dapat meningkatkan kompleksitas sistem secara keseluruhan.
b. Tantangan komunikasi: Komunikasi antara microservice dapat menyebabkan overhead tambahan dan mempengaruhi kinerja aplikasi.
c. Kesulitan dalam pengujian: Pengujian aplikasi yang dibangun dengan arsitektur microservice mungkin lebih sulit karena ketergantungan antar layanan dan kebutuhan untuk menguji komunikasi antara layanan.

2.1.3 Implementasi Arsitektur microservice pada Aplikasi

Untuk mengimplementasikan arsitektur microservice pada aplikasi, langkah-langkah berikut dapat diikuti (Lewis & Fowler, 2014; Newman, 2021):

a. Identifikasi komponen: Tentukan fungsionalitas utama aplikasi dan pecah menjadi komponen independen yang dapat dikembangkan sebagai microservice.
b. Desain API: Buat API yang terdefinisi dengan baik untuk komunikasi antara microservice.
c. Pilih teknologi: Tentukan teknologi yang paling sesuai untuk setiap *microservice*, seperti bahasa pemrograman, kerangka kerja, dan basis data.

d. Pengembangan dan integrasi: Kembangkan setiap *microservice* secara individual dan integrasikan dengan aplikasi secara keseluruhan melalui API yang telah ditentukan.
e. Monitor dan kelola: Gunakan alat pemantauan dan manajemen yang sesuai untuk mengawasi kinerja, keandalan, dan skalabilitas microservice dalam aplikasi.

Dengan mengimplementasikan arsitektur microservice, aplikasi akan menjadi lebih modular, fleksibel, dan andal. Namun, penting untuk mempertimbangkan kompleksitas tambahan dan tantangan yang mungkin dihadapi selama pengembangan dan pengujian. Dalam banyak kasus, keuntungan dari arsitektur microservice akan melebihi kekurangannya, terutama dalam aplikasi skala besar yang memerlukan tingkat skalabilitas dan keandalan yang tinggi.

2.2 Optimasi Aplikasi

2.2.1 Definisi Optimasi Aplikasi

Dalam dunia teknologi, optimasi aplikasi dipandang sebagai upaya kritis yang memperbaiki kualitas aplikasi dari berbagai dimensi. Berdasarkan definisi yang diajukan oleh Fowler (2019), optimasi aplikasi adalah proses kompleks yang melibatkan peningkatan kinerja, efisiensi, dan keandalan sebuah aplikasi melalui implementasi berbagai teknik dan metode. Proses ini tidak hanya mencakup peningkatan kualitas kode dan struktur program, tetapi juga menyangkut aspek seperti peningkatan kinerja basis data, penyesuaian infrastruktur, dan peningkatan desain antarmuka pengguna.

Fokus utama dari optimasi aplikasi tidak terbatas pada perbaikan teknis semata, tetapi juga mencakup peningkatan kualitas pengalaman pengguna. Hal ini ditunjukkan oleh Souders (2007), yang menyatakan bahwa tujuan utama dari optimasi aplikasi adalah untuk memperbaiki pengalaman pengguna dan mencapai tujuan bisnis dengan memanfaatkan sumber daya yang ada secara efisien. Ini berarti bahwa optimasi aplikasi juga melibatkan pemahaman mendalam tentang kebutuhan dan harapan pengguna, serta penyesuaian aplikasi dengan tujuan bisnis perusahaan.

Secara keseluruhan, optimasi aplikasi bukanlah suatu proses yang statis, tetapi merupakan siklus perbaikan berkelanjutan yang mengharuskan tim pengembangan untuk secara konsisten memantau kinerja aplikasi, menganalisis isu-isu yang ada, dan menerapkan perbaikan yang diperlukan. Optimasi aplikasi adalah elemen penting dalam siklus hidup pengembangan perangkat lunak, yang bertujuan untuk memaksimalkan nilai dari produk akhir baik untuk pengguna akhir maupun untuk perusahaan.

2.2.2 Jenis-jenis Optimasi Aplikasi

Kinerja aplikasi dapat dimaksimalkan melalui proses optimasi, yang menjadi langkah penting dalam pengembangan perangkat lunak. Beberapa aspek aplikasi, mulai dari penulisan kode hingga desain antarmuka pengguna dan konfigurasi infrastruktur, dapat ditingkatkan melalui optimasi. Berikut adalah beberapa jenis optimasi aplikasi yang dijelaskan beserta rujukan literatur yang relevan (Allspaw & Robbins, 2010; Bai, 2011; Krug, 2013; McConnell, 2004).

Optimasi aplikasi dapat dibagi menjadi beberapa jenis, di antaranya:

a. Optimasi kode: Melibatkan penulisan kode yang lebih efisien dan ringkas, meminimalkan kompleksitas, dan mengurangi redundansi.
b. Optimasi basis data: Meningkatkan kinerja akses dan manipulasi data dengan mengoptimalkan *query* dan struktur basis data.
c. Optimasi desain UI/UX: Meningkatkan keterlibatan pengguna dan kepuasan dengan memperbaiki desain antarmuka dan alur aplikasi.
d. Optimasi infrastruktur: Menyesuaikan kapasitas dan konfigurasi sistem untuk memperbaiki kinerja dan skalabilitas aplikasi.

2.2.3 Manfaat Optimasi Aplikasi

Di dalam konteks pengembangan perangkat lunak, optimasi aplikasi memberikan berbagai manfaat signifikan yang membawa dampak positif baik untuk pengguna maupun pengembang. Beberapa manfaat penting ini mencakup peningkatan kinerja, penggunaan sumber daya yang efisien, keandalan, stabilitas, dan peningkatan pengalaman pengguna. Rincian dari manfaat ini akan dijelaskan lebih lanjut.

a. Peningkatan kinerja: Aplikasi yang dioptimalkan menawarkan waktu respons yang lebih cepat dan memproses data dengan lebih efisien (Souders, 2007).
b. Penggunaan sumber daya yang efisien: Optimasi mengurangi konsumsi sumber daya seperti memori, CPU, dan bandwidth, sehingga mengurangi biaya operasional (Allspaw & Robbins, 2010).
c. Keandalan dan stabilitas: Aplikasi yang dioptimalkan cenderung memiliki risiko kegagalan yang lebih rendah dan lebih mudah dikelola (Fowler, 2019).
d. Pengalaman pengguna yang lebih baik: Optimasi aplikasi menghasilkan antarmuka yang lebih responsif, intuitif, dan mudah digunakan, sehingga meningkatkan kepuasan pengguna (Krug, 2013).

2.2.4 Strategi Optimasi Aplikasi

Optimasi aplikasi adalah suatu proses yang melibatkan serangkaian strategi dan teknik yang secara spesifik dirancang untuk memperbaiki kinerja dan efisiensi aplikasi. Berbagai strategi ini mencakup, namun tidak terbatas pada, profiling dan benchmarking, analisis dan perbaikan kode, penyesuaian basis data, optimasi desain dan alur aplikasi, monitoring dan analisis data, serta kompresi pengiriman data.

a. Profiling dan benchmarking: Mengukur kinerja aplikasi untuk mengidentifikasi area yang memerlukan perbaikan (Myers dkk., 2011).
b. Analisis dan perbaikan kode: Mengkaji kode sumber dan mengimplementasikan perubahan yang diperlukan untuk meningkatkan kinerja dan efisiensi (McConnell, 2004).
c. Penyesuaian basis data: Mengoptimalkan query dan struktur basis data untuk mempercepat akses dan manipulasi data (Bai, 2011).
d. Optimasi desain dan alur aplikasi: Memperbaiki desain antarmuka dan alur aplikasi untuk meningkatkan keterlibatan dan kepuasan pengguna (Krug, 2013).

e. Monitoring dan analisis data: Mengumpulkan dan menganalisis data pengguna untuk mengidentifikasi area yang memerlukan perbaikan dan menyesuaikan strategi optimasi (Allspaw & Robbins, 2010).
f. Kompresi pengiriman data: Mengurangi besar ukuran data dengan cara mengompresi sebelum mengirim data.

Dengan menerapkan strategi optimasi ini, diharapkan aplikasi menjadi lebih efisien, responsif, dan ramah pengguna, menciptakan pengalaman yang lebih baik bagi pengguna dan membantu organisasi mencapai tujuan bisnisnya.

2.3 HPack

2.3.1 Definisi HPack

HPack adalah algoritma kompresi yang digunakan untuk mengurangi ukuran data dalam komunikasi HTTP/2 menggunakan huffman encoding. Algoritma ini memanfaatkan struktur tabel dinamis yang memungkinkan pengkodean dan dekoding Header yang lebih efisien, mengurangi overhead, dan meningkatkan kinerja (Peon & Ruellan, 2015). Tujuan utama dari HPack adalah untuk mengurangi latensi dan meningkatkan kecepatan transfer data dalam protokol HTTP/2.

2.3.2 Fungsi dan Kelebihan HPack

Fungsi utama HPack adalah untuk mengompresi Header pada protokol HTTP/2, sehingga mengurangi ukuran data yang perlu ditransmisikan. kelebihan HPack meliputi Efisiensi kompresi, HPack menggunakan struktur tabel dinamis dan stais yang mengurangi redundansi dalam data, sehingga menghasilkan tingkat kompresi yang lebih baik (Peon & Ruellan, 2015).

2.3.3 Cara Kerja HPack

HPack bekerja dengan menggunakan dua tabel, yaitu tabel statis dan tabel dinamis, untuk mengasosiasikan field header ke indeks. Tabel statis adalah tabel yang telah ditentukan sebelumnya dan berisi field header yang umum (sebagian besar dengan nilai kosong). Sedangkan tabel dinamis bersifat dinamis dan dapat digunakan oleh encoder untuk mengindeks field header yang diulangi dalam daftar header yang dikodekan (Peon & Ruellan, 2015).

Proses kompresi di HPack tidak secara spesifik menggambarkan algoritma untuk encoder. Sebaliknya, HPack mendefinisikan dengan tepat bagaimana decoder diharapkan beroperasi, memungkinkan encoder menghasilkan encoding apa pun yang diperbolehkan oleh definisi tersebut.

Untuk mendekompresi blok header, decoder hanya perlu mempertahankan tabel dinamis sebagai konteks decoding. Tidak ada status dinamis lain yang diperlukan. Saat digunakan untuk komunikasi dua arah, seperti dalam HTTP, tabel dinamis encoding dan decoding yang dipertahankan oleh titik akhir sepenuhnya independen, yaitu, tabel dinamis permintaan dan respons adalah terpisah (Peon & Ruellan, 2015).

Decoder memproses blok header secara berurutan untuk merekonstruksi daftar header asli. Blok header adalah penggabungan representasi field header. Setelah field header didekode dan ditambahkan ke daftar header yang direkonstruksi, field header tidak dapat dihapus. Field header yang ditambahkan ke daftar header dapat dengan aman diteruskan ke aplikasi. Dengan meneruskan field header yang dihasilkan ke aplikasi, decoder dapat diimplementasikan dengan komitmen memori transitori minimal selain memori yang diperlukan untuk tabel dinamis (Peon & Ruellan, 2015).

Semua representasi field header yang terkandung dalam blok header diproses dalam urutan munculnya. Representasi yang diindeks melibatkan penambahan field header yang sesuai dengan entri yang dirujuk dalam tabel statis atau dinamis ke daftar header yang didekodekan. Representasi literal yang tidak ditambahkan ke tabel dinamis melibatkan penambahan field header ke daftar header yang didekodekan. Representasi literal yang ditambahkan ke tabel dinamis melibatkan penambahan field header ke daftar header yang didekodekan dan penambahan field header ke awal tabel dinamis. Penambahan ini bisa mengakibatkan pengusiran entri sebelumnya di tabel dinamis (Peon & Ruellan, 2015).

Untuk membatasi kebutuhan memori pada sisi decoder, ukuran tabel dinamis dibatasi. Ukuran tabel dinamis adalah jumlah ukuran entri-entrinya. Ukuran entri adalah jumlah panjang namanya dalam oktet, panjang nilainya dalam

oktet, dan 32. Ukuran entri dihitung menggunakan panjang namanya dan nilai tanpa kode Huffman yang diterapkan.

2.4 Gzip

Gzip adalah program kompresi data yang dikembangkan oleh Jean-loup Gailly. Program ini menggunakan algoritma Lempel-Ziv (LZ77) untuk mengurangi ukuran file. Gzip menggantikan file asli dengan file yang memiliki ekstensi '.gz', sambil mempertahankan kepemilikan, mode, dan waktu akses dan modifikasi yang sama. Gzip hanya akan mencoba mengkompresi file reguler dan akan mengabaikan symbolic links .

2.4.1 Cara Kerja Gzip

Gzip menggunakan metode kompresi yang disebut DEFLATE. DEFLATE merupakan kombinasi dari algoritma kompresi LZ77 (Lempel-Ziv 1977) dan huffman coding. Algoritma LZ77 mencoba untuk mengidentifikasi dan menghapus redundansi dalam data dengan mencari pola kesamaan antara string-string yang berdekatan. Setelah itu, Huffman coding digunakan untuk mengurangi ukuran data dengan memberikan representasi bit yang lebih pendek untuk string-string yang lebih umum.

langkah-langkah utama dalam proses kompresi dengan gzip:

a. Membaca File Terkompresi: Gzip memulai dengan membaca file terkompresi yang akan didekompresi.
b. Mengembalikan Huffman Coding: Gzip menggunakan informasi tambahan dalam file terkompresi untuk mengembalikan tabel Huffman coding yang digunakan dalam proses kompresi.
c. Menggabungkan String: Gzip menggabungkan string-string yang ditemukan dengan menggunakan kamus kompresi yang disertakan dalam file terkompresi.
d. Pengembalian Data Asli: Setelah penggabungan string, gzip mengembalikan data asli dengan mengganti penunjuk kamus dengan string-string yang sesuai.

e. Rekonstruksi Data: Terakhir, gzip merekonstruksi data asli dengan menggabungkan string-string yang telah dikembalikan sesuai dengan urutan aslinya.

2.4.2 Keuntungan Menggunakan Gzip

Gzip biasanya dapat mengurangi ukuran teks seperti kode sumber atau bahasa Inggris sebesar 60-70%. Kompresi umumnya jauh lebih baik daripada yang dicapai oleh LZW (seperti yang digunakan dalam kompres), Huffman coding (seperti yang digunakan dalam pack), atau adaptive Huffman coding (compact) (Gailly, 2022).

2.5 JSON(Javascript Object Notation)

JSON (JavaScript Object Notation) adalah format pertukaran data yang ringan dan mudah dibaca oleh manusia serta mudah diinterpretasikan oleh komputer. Dalam konteks pengembangan web dan aplikasi, JSON digunakan secara luas untuk mentransmisikan dan menyimpan data struktural.

2.5.1 Definisi dan Struktur JSON

JSON merupakan representasi struktural data yang terdiri dari pasangan "kunci-nilai" (key-value pairs). Data dalam format JSON disusun dalam objek (object), larik (array), dan tipe data primitif seperti string, angka, boolean, dan nilai null. Format JSON sering kali mengadopsi sintaks mirip dengan objek dan larik dalam JavaScript, sehingga mudah dipahami oleh para pengembang (Tiwary dkk., 2021).

2.5.2 Keunggulan JSON

Salah satu keunggulan utama JSON adalah keterbacaannya (Tiwary dkk., 2021). Dengan menggunakan sintak yang intuitif dan mudah dipahami, JSON memungkinkan pengembang dan pengguna manusia untuk membaca dan mengerti data dengan mudah. Selain itu, JSON juga mendukung interoperabilitas, yang berarti format ini dapat digunakan secara luas dalam berbagai bahasa pemrograman dan dapat diimplementasikan di berbagai platform. JSON juga sering digunakan dalam API (*Application Programming Interface*) untuk mentransmisikan data

antara server dan klien. Format JSON juga memungkinkan pengolahan data struktural dengan mudah, termasuk parsing, manipulasi, dan serialisasi.

2.5.3 Implementasi JSON

Pemrosesan JSON dapat dilakukan baik pada sisi server maupun pada sisi klien. Pada sisi server, banyak bahasa pemrograman menyediakan pustaka atau modul untuk membaca dan menulis data JSON. Sebagai contoh, PHP menyediakan fungsi json_encode() dan json_decode(), sedangkan Python menyediakan modul json. Pada sisi klien, JavaScript memiliki metode built-in untuk parsing dan mengonversi data JSON menjadi objek yang dapat diakses. Dengan menggunakan metode JSON.parse(), data JSON dapat diubah menjadi objek JavaScript yang dapat dimanipulasi dan digunakan dalam pengembangan aplikasi web (Tiwary dkk., 2021).

JSON digunakan dalam berbagai kasus penggunaan di dalam pengembangan aplikasi dan pertukaran data. Salah satu contoh penggunaan umum adalah dalam komunikasi data antara server dan klien melalui API REST menggunakan format JSON. Dalam arsitektur ini, server akan menyediakan endpoint API yang mengirimkan data dalam format JSON kepada klien, yang kemudian dapat mengonsumsi data tersebut. Selain itu, JSON juga sering digunakan untuk menyimpan konfigurasi dan pengaturan aplikasi dalam file JSON yang dapat dengan mudah diakses dan dikelola. Selain itu, JSON juga memungkinkan pertukaran data antara aplikasi yang berbeda dengan menggunakan format yang seragam dan mudah dipahami.

2.6 Penelitian Terdahulu

Subbab ini memberikan gambaran tentang penelitian-penelitian terdahulu yang relevan dengan topik skripsi ini, yaitu implementasi HPACK dan GZIP pada optimasi aplikasi dengan arsitektur microservice. Beberapa penelitian sebelumnya telah dipelajari dan dijadikan sebagai referensi dalam penulisan skripsi ini.

Sebuah penelitian yang dilakukan oleh Sergiu Jecan dan Nicolae Goga, berjudul "The Influence of the Compression Algorithms on the Information Transmitted Through the HTTP Protocol", telah menunjukkan pengaruh signifikan

dari algoritma kompresi terhadap kualitas informasi yang ditransmisikan melalui protokol HTTP. Hasil penelitian mereka menunjukkan bahwa algoritma GZIP memberikan hasil terbaik dalam hal efisiensi kompresi dan kecepatan transmisi data.Penelitian lain yang relevan adalah "Multi-Objective Optimization of Container-Based Microservice Scheduling in Edge Computing" oleh Guisheng Fan, Liang Chen, Huiqun Yu, dan Wei Qi. Penelitian ini berfokus pada optimasi multi-objektif dari penjadwalan microservice berbasis kontainer dalam edge computing.

Selanjutnya, penelitian berjudul "Ant Colony Algorithm for Multi-Objective Optimization of Container-Based Microservice Scheduling in Cloud" oleh Miao Lin, Jianqing Xi, Weihua Bai, dan Jiayin Wu membahas tentang optimasi multi-objektif dari penjadwalan microservice berbasis kontainer dalam cloud computing. Penelitian ini menawarkan algoritma Ant Colony sebagai solusi untuk permasalahan penjadwalan tersebut.

Meskipun fokus dan konteks dari penelitian-penelitian ini berbeda dengan skripsi ini, mereka memberikan wawasan penting tentang cara kerja algoritma kompresi dan arsitektur microservice, yang menjadi bagian penting dari skripsi ini. Pengaruh algoritma kompresi dan arsitektur microservice pada kualitas dan efisiensi aplikasi menjadi pertimbangan penting dalam penelitian ini.

Tabel 2.1 Penelitian Terdahulu

No.	Judul	Peneliti
1	"The Influence of the Compression Algorithms on the Information Transmitted Through the HTTP Protocol"	Sergiu Jecan, Nicolae Goga
	Hasil Penelitian: Dalam penelitian ini, ditunjukkan bahwa kualitas informasi yang ditransmisikan melalui protokol HTTP dapat dipengaruhi oleh algoritma kompresi. Beberapa algoritma kompresi yang berbeda dibandingkan oleh peneliti, dan ditemukan bahwa algoritma GZIP memberikan hasil terbaik dalam hal efisiensi kompresi dan kecepatan transmisi data.	
	Perbedaan: Antara dua penelitian ini terletak pada fokus dan konteks mereka. Pengaruh algoritma kompresi pada kualitas informasi yang ditransmisikan melalui protokol HTTP secara umum menjadi fokus dalam penelitian pertama, sementara penelitian kedua lebih spesifik, dengan berfokus pada implementasi HPACK dan GZIP dalam konteks aplikasi dengan arsitektur microservice. Meskipun keduanya membahas tentang algoritma kompresi, konteks dan tujuan mereka berbeda.	
2	"Multi-Objective Optimization of Container-Based Microservice Scheduling in Edge Computing"	Guisheng Fan, Liang Chen, Huiqun Yu, dan Wei Qi
	Hasil Penelitian: Dalam penelitian ini, dibahas tentang optimasi multi-objektif dari penjadwalan microservice berbasis kontainer dalam edge computing. Algoritma LRLBAS (Latency, Reliability, and Load Balancing Aware Scheduling) yang didasarkan pada optimasi swarm partikel (PSO) diusulkan oleh peneliti. Tujuan dari algoritma ini adalah mengoptimalkan latensi jaringan antar microservice, keandalan aplikasi microservice, dan keseimbangan beban kluster.	

Tabel 2.1 Penelitian Terdahulu

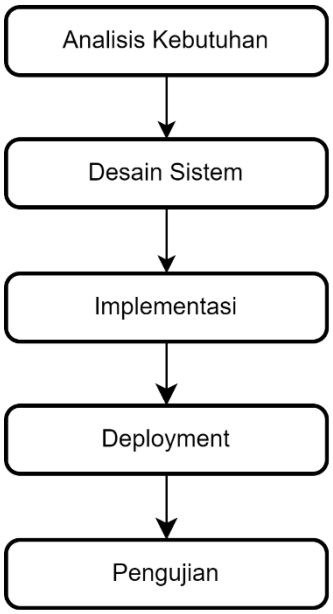
No.	Judul	Peneliti
	Perbedaan: Perbedaan antara dua penelitian ini terletak pada fokus dan konteks mereka. Optimasi multi-objektif dari penjadwalan microservice berbasis kontainer dalam edge computing menjadi fokus dalam penelitian pertama, sementara penelitian kedua lebih spesifik, dengan berfokus pada implementasi HPACK dan GZIP dalam konteks aplikasi dengan arsitektur microservice. Meskipun keduanya membahas tentang arsitektur microservice, konteks dan tujuan mereka berbeda.	
3	"Ant Colony Algorithm for Multi-Objective Optimization of Container-Based Microservice Scheduling in Cloud"	Miao Lin, Jianqing Xi, Weihua Bai, Jiayin Wu
	Hasil Penelitian: Dalam penelitian ini, dibahas tentang optimasi multi-objektif dari penjadwalan microservice berbasis kontainer dalam cloud computing. Algoritma Ant Colony diusulkan oleh peneliti untuk menyelesaikan masalah penjadwalan. Algoritma ini mempertimbangkan bukan hanya pemanfaatan sumber daya komputasi dan penyimpanan dari node fisik, tetapi juga jumlah permintaan microservice dan tingkat kegagalan dari node fisik. Hasil eksperimental menunjukkan bahwa algoritma optimasi yang diusulkan mencapai hasil yang lebih baik dalam optimasi keandalan layanan kluster, keseimbangan beban kluster, dan overhead transmisi jaringan.	
	Perbedaan: Diantara dua penelitian ini terletak pada fokus dan konteks mereka. Optimasi multi-objektif dari penjadwalan microservice berbasis kontainer dalam cloud computing menjadi fokus dalam penelitian pertama, sementara penelitian kedua lebih spesifik, dengan berfokus pada implementasi HPACK dan GZIP dalam konteks aplikasi dengan arsitektur microservice. Meskipun keduanya membahas tentang arsitektur microservice, konteks dan tujuan mereka berbeda.	

BAB III

METODE PENELITIAN

3.1 Tahapan Pengembangan

Pengembangan dalam penelitian ini menggunakan metodologi *waterfall* yang dibagi menjadi lima bagian utama yaitu analisis kebutuhan, desain sistem, implementasi, *deployment* dan pengujian Pada Gambar 3.1 dijelaskan secara visual.



Gambar 3.1 Bagan Tahapan Penelitian

3.2 Analisis Kebutuhan

Optimasi aplikasi menggunakan arsitektur mikroservis menjadi tujuan utama, sehingga diperlukan sistem dengan struktur mikroservis. Dalam hal ini, mikroservis media sosial akan dikembangkan sebagai media uji. Metode yang dipilih untuk mengoptimasi aplikasi mencakup kompresi setiap *input* atau *output* dari sebuah, seperti dengan mengompresi data JSON. Algoritma seperti HPack dan GZIP diimplementasikan untuk memudahkan proses optimasi dan mikroservis. Proses kompresi dan dekompresi JSON akan dipisahkan menjadi modul yang dapat diinstal dan digunakan dalam setiap *service*. Selanjutnya, *frontend* diperlukan untuk mengakses atau mensimulasi interaksi dengan mikroservis dan juga bertindak sebagai media untuk pengujian dan perhitungan performa dari pengompresian JSON.

Sebagai panduan lebih lanjut, tabel berikut ini menyajikan detail tentang modul, layanan mikroservis, dan elemen frontend yang akan digunakan. Dalam konteks modul, kompresi gzip, kompresi hpack, dan data JSON menjadi fokus utama. Layanan mikroservis mencakup autentikasi, pengguna, postingan, interaksi postingan, tag, dan kontainerisasi. Sementara itu, frontend akan memanfaatkan antarmuka pengguna, metrik performa, framework React, dan Tailwind CSS. Rincian ini bertujuan untuk memberikan pemahaman yang lebih jelas tentang struktur dan mekanisme yang akan diterapkan dalam pengembangan aplikasi ini. Seluruh elemen tersebut akan saling berinteraksi dan bekerja sama untuk mencapai tujuan optimasi aplikasi melalui arsitektur mikroservis.

Tabel 3.1 Data Analisa Kebutuhan

Modul	Microservice	Frontend
Kompresi gzip	Autentikasi	Antarmuka Pengguna
Kompresi hpack	Pengguna	Metrik Performa
Data JSON	Postingan	Framework React
Perhitungan performa	Interaksi Postingan	*Framework* Tailwind CSS
Pilihan metode kompresi	Tag	Kontainerisasi
	Pengguna	
	Postingan	
	Feed Beranda	
	Kontainerisasi	

3.2.1 Modul Kompresi Dekompresi JSON

Dalam kebutuhan modul, diidentifikasi beberapa elemen penting seperti dukungan terhadap kompresi gzip dengan level kompresi default, dukungan terhadap kompresi hpack menggunakan JSONH, dapat menerima *input* JSON, dan fitur untuk melakukan penghitungan performa(waktu proses, ukuran sebelum dan sesudah kompresi). Selain itu, fitur untuk memilih tipe kompresi seperti auto, Hpack, Gzip ataupun keduanya.

3.2.2 Microservice Social Media

Untuk microservice sebagai contoh akan menggunakan aplikasi sosial media. Kebutuhan *microservice* mencakup sebagai berikut: proses pendaftaran pengguna, login, verifikasi token, pemulihan kata sandi, logout, pengelola profil pengguna, postingan, interaksi postingan atau komentar, tag, Analitik Pengguna, Analitik Postingan, dan Feed Beranda. Semua *service* dikontainerkan menggunakan Docker.

3.2.3 Frontend

Kebutuhan *frontend* mencakup antarmuka pengguna untuk berinteraksi dengan *microservice*, fitur untuk menampilkan metrik performa, dan aplikasi ini di

dikembangkan menggunakan *framework* React dan Tailwind CSS. Selain itu, frontend juga dikontainerkan menggunakan Docker.

Kebutuhan fitur frontend adalah sebagai berikut:

a. Autentikasi:
   - Pendaftaran pengguna.
   - Login.
   - Verifikasi token pengguna.
   - Pemulihan password.
   - Reset password pengguna.
   - Logout pengguna.

b. Manajemen Profil Pengguna dan Teman:
   - Tampilan profil pengguna.
   - Memperbarui profil pengguna.
   - Daftar teman pengguna.
   - Mengirim, menerima, dan membatalkan permintaan pertemanan.
   - Menghapus teman dari daftar teman pengguna.
   - Permintaan pertemanan yang tertunda.

c. Postingan dan Interaksi:
   - Membuat postingan baru.
   - Tampilan detail postingan.
   - Memperbarui postingan.
   - Menghapus postingan.
   - Pencarian postingan.
   - Menyukai dan tidak menyukai postingan.
   - Tampilan postingan yang disukai oleh pengguna.
   - Menambahkan dan menghapus komentar pada postingan.
   - Memperbarui komentar.
   - Tampilan komentar untuk postingan.

d. Tag:
   - Menambahkan dan menghapus tag pada postingan.
   - Tampilan tag yang terkait dengan postingan.

- Memperbarui tag.

e. Analitik:
   - Pencatatan informasi login pengguna.
   - Tampilan catatan login pengguna.
   - Pencatatan analitik pencarian pengguna.
   - Tampilan analitik pencarian pengguna.
   - Tampilan jumlah tampilan, suka, dan komentar untuk postingan.

f. Feed Beranda:
   - Tampilan feed beranda pengguna.

3.3 Desain Sistem

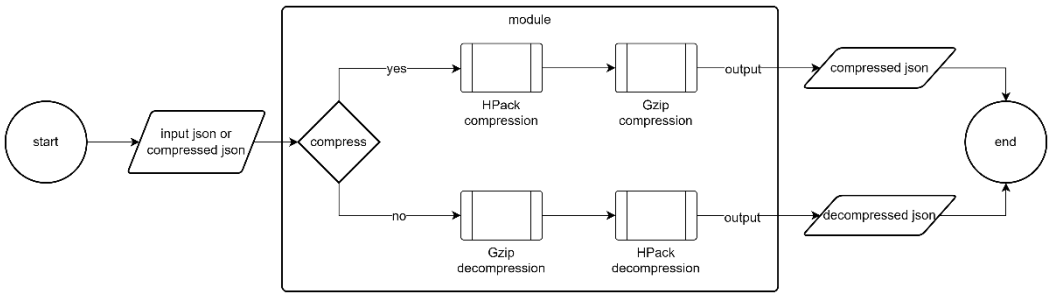
Fase ini merupakan tahap kunci dalam penelitian karena pada tahap ini kerangka kerja aplikasi akan dirancang. Fokus utama dari tahap ini adalah mendiskusikan bagaimana desain sistem dilakukan untuk mencapai optimalisasi aplikasi dengan arsitektur microservice menggunakan HPack dan Gzip. Selain itu, bagaimana perancangan sistem tersebut berkontribusi terhadap efisiensi dan efektivitas aplikasi juga akan dibahas.

Perancangan sistem adalah tahap penting dalam penelitian, yang mencakup identifikasi dan pemahaman berbagai komponen sistem, bagaimana komponen tersebut berinteraksi, dan bagaimana mereka berkontribusi terhadap tujuan umum dari sistem. Dalam penelitian ini, perancangan sistem akan melibatkan tiga komponen utama yaitu Modul Kompresi Dekompresi JSON, Microservice Media Sosial, dan Frontend.

Setelah ini, akan diikuti dengan pembahasan mendalam mengenai desain sistem untuk tiga komponen yang telah disebutkan di atas. Bab ini akan membantu dalam memahami bagaimana komponen tersebut dirancang, serta bagaimana setiap komponen berkontribusi terhadap optimalisasi aplikasi secara keseluruhan dalam konteks arsitektur microservice dengan menggunakan HPack dan Gzip. Setiap bagian akan berisi rincian tentang desain dan fungsi dari komponen tersebut, serta penjelasan tentang bagaimana mereka berinteraksi dengan komponen lain dalam sistem.

3.3.1 Modul kompresi dekompresi JSON

Dua aspek penting perlu dipertimbangkan dalam perancangan modul ini yaitu kemudahan penggunaan dan fleksibilitas. Modul harus dirancang sedemikian rupa sehingga mudah digunakan, tetapi tetap memungkinkan penyesuaian sesuai kebutuhan pengguna.



Gambar 3.2 Proses Kerja Modul

Fungsi Modul idealnya memaparkan dua fungsi utama:

a. kompres(jsonObject): Melakukan proses kompresi pada objek JSON. Fungsi ini mengembalikan sebuah *promise* yang berisi data yang telah dikompres.
b. dekompres(objekTerompres): Melakukan proses dekompresi data. Fungsi ini mengembalikan sebuah *promise* yang berisi objek JSON yang telah didekompresi.

Kesalahan Modul seharusnya juga dapat mendefinisikan dan melemparkan beberapa jenis kesalahan khusus, selain kesalahan JavaScript standar:

a. InvalidAlgorithmError: Jenis kesalahan ini dilempar jika pengguna mencoba menggunakan algoritma yang tidak didukung.
b. CompressionError: Jenis kesalahan ini dilempar jika terjadi masalah selama proses kompresi.
c. DecompressionError: Jenis kesalahan ini dilempar jika terjadi masalah selama proses dekompresi. Kesalahan khusus ini akan membantu pengguna untuk lebih mudah memahami dan menangani masalah yang muncul.

Struktur Internal Struktur internal dari modul ini terdapat dua sub modul.

a. modulKompres: Modul ini akan mengandung fungsi kompres yang menerima objek JSON dan mengembalikan promise dengan data yang telah dikompres.
b. modulDekompres: Modul ini akan mengandung fungsi dikompres yang menerima data yang telah dikompres dan mengembalikan promise dengan objek JSON yang telah didekompresi.

Modul-modul ini akan digunakan oleh fungsi kompresi dan dekompresi pada modul utama.

3.3.2 Microservice Social Media

Pengembangan arsitektur berbasis microservice untuk aplikasi media sosial merupakan proses yang membutuhkan perencanaan dan desain yang teliti. Dalam pendekatan ini, pendekatan utamanya adalah dengan memecah fungsi-fungsi aplikasi menjadi komponen-komponen kecil yang independen, yang disebut sebagai microservice. Pendekatan ini memberikan keuntungan dalam hal skalabilitas, keberlanjutan, dan fleksibilitas. Dalam metodologi desain ini, kami akan menjelaskan langkah-langkah yang terlibat dalam pengembangan microservice dan penggunaan Docker, dengan memanfaatkan teknologi Node.js, PostgreSQL, Express.js, dan Sequelize.

Pada pengembangan aplikasi media sosial berbasis mikroservice ini, terdapat beberapa service yang perlu diperhatikan. service tersebut antara lain Service Autentikasi, Service Pengguna, Service Pos, Service Interaksi, Service Tag, Service Analitik Pengguna, Service Analitik Pos, dan Service Beranda. Setiap Service memiliki peran dan tanggung jawabnya sendiri dalam menjalankan fungsi-fungsi khusus seperti manajemen pengguna, postingan, interaksi, tag, dan analitik.

3.3.2.1 Service Autentikasi

a. Definisi Service: Service Identifikasi mengurus registrasi pengguna, login, verifikasi token, pemulihan kata sandi, dan logout.

b. Desain API:

Tabel 3.2 Definisi API Autentikasi

Endpoint	Method	Deskripsi
/auth/register	POST	Membuat akun pengguna baru
/auth/login	POST	Autentikasi pengguna dan hasilkan token akses
/auth/verify-token	POST	Verifikasi keberlakuan token akses
/auth/recover-password	POST	Memulai proses pemulihan kata sandi
/auth/reset-password	POST	Atur ulang kata sandi pengguna
/auth/logout	POST	Mencabut token akses

c. Desain Database:

Tabel 3.3 Definisi Tabel User

Column Name	Data Type
id	UUID (primary key)
username	VARCHAR(255) (unique)
email	VARCHAR(255) (unique)
password_hash	VARCHAR(255)
refresh_token	VARCHAR(255)

3.3.2.2 Service Pengguna

a. Definisi Service: Service Pengguna mengelola profil pengguna, termasuk bio, nama pengguna, nama lengkap, foto sampul, dan daftar teman. Pengguna dapat melihat profil sendiri atau profil teman mereka. Service ini juga mengurus permintaan pertemanan.

b. Desain API:

Tabel 3.4 Definisi API Pengguna

Endpoint	Method	Deskripsi
/users/{userId}	GET	Mengambil profil pengguna berdasarkan ID
/users/{userId}/profile	GET	Mengambil profil pengguna sendiri
/users/{userId}/friends	GET	Mengambil daftar teman pengguna
/users/{userId}/friends/{friendId}	POST	Kirim permintaan pertemanan
/users/{userId}/friends/{friendId}	DELETE	Batalkan permintaan pertemanan
/users/{userId}/friends/{friendId}	PUT	Terima permintaan pertemanan
/users/{userId}/friends/{friendId}	DELETE	Hapus teman
/users/{userId}/friends/requests	GET	Mengambil permintaan pertemanan yang tertunda untuk seorang pengguna
/users/{userId}/profile	PUT	Perbarui profil pengguna

c. Desain Database:

Tabel 3.5 Definisi Tabel UserProfile

Column Name	Data Type
user_id	UUID (primary key, foreign key dari Users.id)
bio	TEXT
username	VARCHAR(255) (unique)
full_name	VARCHAR(255)
cover_image	VARCHAR(255)

Tabel 3.6 Definisi Tabel UserProfile

Column Name	Data Type
user_id	UUID (primary key, foreign key dari UserProfile.user_id)
friend_id	UUID (primary key, foreign key dari UserProfile.user_id)
status	ENUM('pending', 'accepted')

Tabel 3.7 Definisi Tabel Friend

Column Name	Data Type
sender_user_id	UUID (primary key, foreign key dari UserProfile.user_id)
receiver_user_id	UUID (primary key, foreign key dari UserProfile.user_id)
request_status	ENUM('pending', 'accepted', 'rejected')

3.3.2.3 Service Post

a. Definisi Service: Service Post mengelola postingan, termasuk pembuatan, pengeditan, penampilan, penghapusan, dan pencarian postingan. Setiap postingan akan menyimpan "tanggal interaksi terakhir" yang diperbarui ketika ada interaksi dengan postingan tersebut.

b. Desain API:

Tabel 3.8 Definisi API Post

Endpoint	Method	Deskripsi
/posts	POST	Buat kiriman baru
/posts/{postId}	GET	Mengambil detail kiriman berdasarkan ID
/posts/{postId}	PUT	Perbarui kiriman berdasarkan ID
/posts/{postId}	DELETE	Hapus kiriman berdasarkan ID
/posts/search	GET	Cari kiriman berdasarkan kriteria tertentu

c. Desain Database:

Tabel 3.9 Definisi Tabel Post

Column Name	Data Type
id	UUID (primary key)
user_id	UUID (foreign key dari UserProfile.user_id)
text_content	TEXT
created_date	TIMESTAMP
edited_date	TIMESTAMP
last_interaction_date	TIMESTAMP

3.3.2.4 Service Interaksi

a. Definisi Service: Service Interaksi mengelola interaksi dengan postingan atau komentar, termasuk suka dan komentar.

b. Desain API:

Endpoint	Method	Deskripsi
/posts/{postId}/likes	POST	Sukai sebuah kiriman
/posts/{postId}/likes	DELETE	Batalkan suka sebuah kiriman
/users/{userId}/likes	GET	Mengambil kiriman yang disukai oleh seorang pengguna
/posts/{postId}/comments	POST	Tambahkan komentar ke sebuah kiriman
/posts/{postId}/comments/{commentId}	DELETE	Hapus komentar
/posts/{postId}/comments/{commentId}	PUT	Perbarui komentar
/posts/{postId}/comments	GET	Mengambil komentar untuk sebuah kiriman

c. Desain Database:

Tabel 3.10 Definisi Tabel PostInteraction

Column Name	Data Type
user_id	UUID (foreign key dari UserProfile.user_id)
post_id	UUID (foreign key dari Post.id)
interaction_type	ENUM('like', 'comment')
created_date	TIMESTAMP

Tabel 3.11 Definisi Tabel Comment

Column Name	Data Type
id	UUID (primary key)
user_id	UUID (foreign key dari UserProfile.user_id)
post_id	UUID (foreign key dari Post.id)
text_content	TEXT
created_date	TIMESTAMP

Tabel 3.12 Definisi Tabel Like

Column Name	Data Type
id	UUID (primary key)
user_id	UUID (foreign key dari UserProfile.user_id)
post_id	UUID (foreign key dari Post.id)
created_date	TIMESTAMP

3.3.2.5 Service Tag

a. Definisi Service: Service Tag mengurus tag yang terkait dengan postingan, termasuk pembuatan, pengeditan, penampilan, dan penghapusan.

b. Desain API:

Tabel 3.13 Definisi API Tag

Endpoint	Method	Deskripsi
/tags/	POST	Membuat sebuah tag
/tags/	DELETE	Menghapus sebuah tag
/tags/	GET	Mengambil semua tag
/tags/{tagId}	PUT	Perbarui sebuah tag
/tags/posts/{postId}	POST	Tambahkan tag ke sebuah kiriman
/tags/posts/{postId}	DELETE	Hapus tag dari sebuah kiriman
/tags/posts	GET	Mengambil post yang terkait dengan sebuah tag
/tags/posts/{postId}	GET	Mengambil tag yang terkait dengan sebuah kiriman
/tags/posts/{postId}	PUT	Perbarui sebuah tag

c. Desain Database:

Tabel 3.14 Definisi Tabel Tag

Column Name	Data Type
id	UUID (primary key)
name	VARCHAR(255) (unique)

Tabel 3.15 Definisi Tabel PostTag

Column Name	Data Type
post_id	UUID (primary key, foreign key dari Post.id)
tag_id	UUID (primary key, foreign key dari Tag.id)

3.3.2.6 Service Analitik Pengguna

a. Definisi Service: Service Analitik Pengguna mencatat dan menampilkan analitik khusus pengguna, termasuk informasi login dan analitik pencarian.

b. Desain API:

Tabel 3.16 Definisi API Analitik Pengguna

Endpoint	**Method**	**Deskripsi**
/users/{userId}/login-logs	POST	Menatat informasi login pengguna
/users/{userId}/login-logs	GET	Mengambil log login pengguna
/users/{userId}/search-analytics	POST	Mencatat analitik pencarian pengguna
/users/{userId}/search-analytics	GET	Mengambil analitik pencarian pengguna

c. Desain Database:

Tabel 3.17 Definisi Tabel LoginRecord

Column Name	**Data Type**
user_id	UUID (primary key, foreign key dari UserProfile.user_id)
login_date	TIMESTAMP
IP_address	VARCHAR(255)
device_information	VARCHAR(255)

Tabel 3.18 Definisi Tabel SearchRecord

Column Name	**Data Type**
user_id	UUID (primary key, foreign key dari UserProfile.user_id)
search_term	VARCHAR(255)
search_date	TIMESTAMP
results_count	INT

3.3.2.7 Service Analitik Post

a. Definisi Service: Service Analitik Pos mencatat dan menampilkan analitik terkait interaksi dengan postingan, seperti tampilan, suka, dan komentar.
b. Desain API:

Tabel 3.19 Definisi API Analitik Post

Endpoint	Method	Deskripsi
/posts/{postId}/views	GET	Mengambil jumlah tampilan untuk sebuah kiriman
/posts/{postId}/likes	GET	Mengambil jumlah suka untuk sebuah kiriman
/posts/{postId}/comments	GET	Mengambil jumlah komentar untuk sebuah kiriman
/posts/{postId}/likes-detailed	GET	Mengambil catatan detail suka kiriman
/posts/{postId}/comments-detailed	GET	Mengambil catatan detail komentar kiriman

c. Desain Database:

Tabel 3.20 Definisi Tabel PostView

Column Name	Data Type
post_id	UUID (primary key, foreign key dari Post.id)
user_id	UUID (foreign key dari UserProfile.user_id)
view_date	TIMESTAMP

Tabel 3.21 Definisi Tabel PostLike

Column Name	Data Type
post_id	UUID (primary key, foreign key dari Post.id)
user_id	UUID (foreign key dari UserProfile.user_id)
like_date	TIMESTAMP

Tabel 3.22 Definisi Tabel PostComment

Column Name	Data Type
post_id	UUID (primary key, foreign key dari Post.id)
user_id	UUID (foreign key dari UserProfile.user_id)
comment_date	TIMESTAMP

3.3.2.8 Service Beranda

a. Definisi Service: Service Beranda mengumpulkan postingan dari teman pengguna dan postingan di mana teman mereka berinteraksi. Postingan disusun berdasarkan "tanggal interaksi terakhir."
b. Desain API:

Tabel 3.23 Definisi API Post

Endpoint	Method	Deskripsi
/feeds/	GET	Mengambil umpan beranda untuk seorang pengguna

c. Desain Database: Pada service ini tidak di butuhkan database dikarenakan semua data berasal dari service lainnya.

Dengan mengikuti metodologi desain tingkat tinggi ini, aplikasi media sosial berbasis mikroservice dapat dikembangkan menggunakan Node.js, PostgreSQL, Express.js, dan Sequelize. Dockerisasi mikroservice memudahkan proses deployment dan pengelolaan. Setiap mikroservice harus diimplementasikan secara independen dengan mengikuti best practice dan design pattern yang sesuai.

3.3.3 Frontend

Pada bagian desain sistem frontend, fokus utama adalah mengembangkan antarmuka pengguna interaktif yang berinteraksi dengan microservice. Aplikasi frontend akan dikembangkan menggunakan React.js, sebuah framework JavaScript yang populer untuk pengembangan antarmuka pengguna. Selain itu, juga akan digunakan Tailwind CSS untuk mengatur tata letak dan gaya tampilan aplikasi.

3.3.3.1 Antarmuka Pengguna (User Interface)

Desain antarmuka pengguna akan berfokus pada menciptakan pengalaman pengguna yang baik dan intuitif. Hal ini meliputi pemilihan warna, tata letak, dan elemen-elemen antarmuka seperti tombol, formulir, dan navigasi.

3.3.3.2 Komunikasi Dengan Microservice Backend

Aplikasi frontend akan berkomunikasi dengan microservice backend melalui API yang disediakan. Setiap endpoint API yang diperlukan untuk fitur-fitur aplikasi akan diimplementasikan dalam komponen-komponen frontend menggunakan *library* axios.

3.3.3.3 Fitur Performa Metrik

Fitur-fitur yang terkait dengan metrik performa diimplementasikan dalam aplikasi frontend. Pengguna dapat melihat metrik seperti jumlah views, likes, dan komentar pada suatu post. Data ini akan diambil melalui API endpoint yang disediakan oleh Microservice Post Analytics.

3.3.3.4 Penggunaan Docker

Aplikasi frontend akan dikontainerkan menggunakan Docker. Hal ini memudahkan dalam pengelolaan dan pengiriman aplikasi serta memastikan portabilitas aplikasi antara lingkungan pengembangan dan produksi.

3.3.3.5 Responsif dan Kompatibilitas

Desain sistem frontend akan memastikan bahwa aplikasi dapat merespons secara baik terhadap perubahan ukuran layar dan perangkat yang berbeda. Selain itu, kompatibilitas dengan berbagai browser yang umum digunakan juga akan diperhatikan.

3.4 Implementasi

Pada tahap ini, berbagai teknologi dan metode telah diterapkan untuk mencapai tujuan penelitian. Sistem yang dikembangkan pada penelitian ini menggunakan pendekatan microservice dengan bantuan bahasa pemrograman JavaScript dan kerangka kerja Express.js. PostgreSQL digunakan sebagai basis data karena skalabilitas dan fitur-fiturnya yang kuat. Selain itu, teknologi HPACK dan

GZIP juga digunakan untuk melakukan kompresi data dan header HTTP/2, masing-masing.

3.4.1 Overview Implementasi

Dalam proyek ini, kami menggunakan JavaScript sebagai bahasa pemrograman utama dan Express.js sebagai kerangka kerja untuk mempermudah pembuatan aplikasi berbasis microservice. Kami memilih PostgreSQL sebagai basis data karena skalabilitas dan fitur-fiturnya yang kuat.

3.4.2 Detil Implementasi

Kami telah mengembangkan berbagai layanan microservice untuk menangani berbagai fungsi di dalam sistem. Misalnya, layanan autentikasi yang menangani registrasi pengguna, login, verifikasi token, pemulihan kata sandi, dan logout. Layanan pengguna yang mengelola profil pengguna termasuk bio, username, nama lengkap, gambar sampul, dan daftar teman. Layanan posting yang mengelola posting, termasuk membuat, mengedit, melihat, menghapus, dan mencari postingan.

Selain itu, kami juga telah mengimplementasikan teknologi HPACK dan GZIP. Kami membuat modul JavaScript khusus untuk mengompresi JSON menggunakan HPACK dan GZIP, yang kemudian digunakan di seluruh layanan kami.

Semua layanan dan frontend dikontainerkan menggunakan Docker, yang memudahkan pengiriman dan penyebaran aplikasi kami.

3.4.3 Proses Implementasi

Proses implementasi dimulai dengan pembuatan modul JavaScript untuk mengompresi JSON dengan HPACK dan GZIP. Setelah itu, kami mulai membangun layanan microservice sesuai dengan desain sistem yang telah kami tentukan. Selanjutnya, kami membuat frontend untuk berinteraksi dengan layanan yang telah kami buat. Terakhir, kami mengontainerkan microservice dan frontend dengan Docker untuk memudahkan pengiriman dan penyebaran aplikasi.

3.5 Deployment

Pada tahap ini, aplikasi yang telah selesai diimplementasi dan diuji dipindahkan ke lingkungan produksi. Deployment aplikasi ini melibatkan beberapa langkah penting.

3.5.1 Pra-deployment

Pada tahap ini, persiapan dilakukan sebelum deployment aplikasi. Meskipun tidak ada spesifikasi khusus untuk infrastruktur Google Cloud, disarankan untuk menggunakan instance yang memberikan keseimbangan optimal antara biaya dan performa. Dalam hal ini, penyesuaian dapat dilakukan berdasarkan kebutuhan spesifik aplikasi.

3.5.2 Deployment

Dalam tahap ini, aplikasi dipindahkan dari lingkungan pengembangan ke lingkungan produksi. Aplikasi ini di-containerize menggunakan Docker, yang memungkinkan peningkatan kontrol, efisiensi, dan portabilitas. Proses deployment aplikasi ke Google Cloud dilakukan dengan Google Cloud Run, yang memungkinkan deployment yang mudah dan cepat.

3.5.3 Pasca-deployment

Setelah aplikasi berhasil dipindahkan ke lingkungan produksi, beberapa tes pengecekan akhir dilakukan untuk memastikan semua fungsi aplikasi bekerja dengan baik dalam lingkungan baru. Untuk pengecekan ini, disarankan melakukan pengujian fungsi dasar dari aplikasi dan pengecekan konektivitas antar komponen microservice.

Proses deployment ini memberikan pondasi yang kuat untuk tahap pengujian selanjutnya, yang akan lebih mengevaluasi performa dan efisiensi dari aplikasi yang telah di-deploy.

3.6 Pengujian

Dalam tahap ini, pengujian dilakukan untuk mengukur efisiensi kompresi dan dampaknya terhadap performa sistem. Waktu load halaman dan ukuran data JSON dibandingkan saat menggunakan kompresi (HPACK dan GZIP) dan saat tidak menggunakan kompresi.

3.6.1 Metodologi Pengujian

Setiap pengujian dijalankan sebanyak lima kali dan rata-rata hasilnya diambil untuk memastikan konsistensi dan akurasi data. Pengujian ini dilakukan pada berbagai halaman dan berbagai jenis data untuk mendapatkan hasil yang representatif.

3.6.2 Alat Pengujian

Google Lighthouse digunakan untuk mengukur waktu load halaman. Alat ini memberikan data waktu load halaman yang akurat dan dapat dipercaya. Untuk mengukur ukuran data JSON, microservice diprogram untuk menghasilkan log yang mencakup ukuran data sebelum dan sesudah kompresi, serta waktu yang dibutuhkan untuk proses kompresi.

3.6.3 Hasil Pengujian

Hasil rata-rata dari pengujian memberikan data tentang waktu memuat halaman, ukuran data JSON, dan waktu kompresi.

3.6.4 Rancangan Analisis *Hasil*

Untuk menganalisis hasil pengujian, beberapa langkah akan diambil. Pertama, peningkatan atau penurunan waktu memuat halaman saat menggunakan kompresi dibandingkan saat tidak menggunakan kompresi akan dihitung. Selanjutnya, penurunan ukuran data JSON saat menggunakan kompresi juga akan dihitung. Data ini kemudian akan digunakan untuk mengevaluasi bagaimana kompresi dapat mempengaruhi performa sistem, termasuk dampaknya terhadap waktu memuat halaman dan ukuran data yang dikirimkan.

DAFTAR PUSTAKA

Allspaw, J., & Robbins, J. (2010). Web Operations: Keeping the Data on Time. O'Reilly Media.

Bai, Y. (2011). *Practical database programming with java*. Wiley-Blackwell.

Bray, T. (2017). *The JavaScript Object Notation (JSON) Data Interchange Format* (T. Bray, Ed.; Request for Comments, Nomor 8259). RFC Editor. https://doi.org/10.17487/RFC8259

Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., & Safina, L. (2017). microservice: Yesterday, today, and tomorrow. Dalam *Present and Ulterior Software Engineering* (hlm. 195–216). Springer International Publishing. https://doi.org/10.1007/978-3-319-67425-4_12

Fowler, M. (2019). *Refactoring: Improving the Design of Existing Code* (2 ed.). Addison Wesley.

Gailly, J.-L. (2022). *GNU gzip*.

Krug, S. (2013). *Don't make me think, revisited: A Common Sense Approach to Web Usability* (3 ed.). New Riders Publishing.

Lewis, J., & Fowler, M. (2014, Maret 12). *microservice*. https://martinfowler.com/articles/ microservice.html

McConnell, S. (2004). *Code Complete* (2 ed.). Microsoft Press.

Myers, G. J., Sandler, C., & Badgett, T. (2011). *The art of software testing* (G. J. Myers, T. Badgett, & C. Sandler, Ed.; 3 ed.). John Wiley & Sons.

Newman, S. (2021). *Building microservice: Designing Fine-Grained Systems* (2 ed.). O'Reilly Media.

OBJELEAN, A. (2011). *JSON Compression Algorithms*. http://repository.utm.md/handle/5014/6418

Peon, R., & Ruellan, H. (2015). *HPACK: Header Compression for HTTP/2*. https://doi.org/10.17487/RFC7541

Richards, M. (2022). *Software architecture patterns* (M. Duffield, S. Evans, & K. Brown, Ed.; Second Edition). O'Reilly Media, Inc.

Richardson, C. (2018). *microservice patterns: with examples in Java*. Simon and Schuster.

Salah, T., Zemerly, M. J., Yeun, C. Y., Al-Qutayri, M., & Al-Hammadi, Y. (2017). The evolution of distributed systems towards microservice architecture. *2016 11th International Conference for Internet Technology and Secured Transactions, ICITST 2016*, 318–325. https://doi.org/10.1109/ICITST.2016.7856721

Souders, S. (2007). *High performance web sites: Essential Knowledge for Front-End* (1 ed.). O'Reilly Media.

Tiwary, G. P., Stroulia, E., & Srivastava, A. (2021). Compression of XML and JSON API Responses. *IEEE Access*, *9*, 57426–57439. https://doi.org/10.1109/ACCESS.2021.3073041