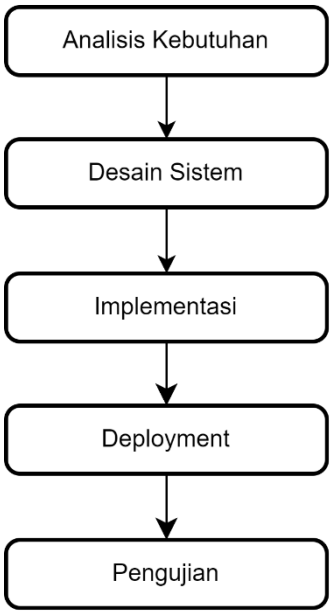
BAB III

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 *microservice* menjadi tujuan utama, sehingga diperlukan sistem dengan struktur *microservice*. Dalam hal ini, *microservice* media sosial akan dikembangkan sebagai media uji. Metode yang dipilih untuk optimasi 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 *microservice*. Proses kompresi dan dekompresi *JSON* akan dipisahkan menjadi modul yang dapat di *install* dan digunakan dalam setiap layanan. Selanjutnya, *frontend* diperlukan

untuk mengakses atau mensimulasi interaksi dengan *microservice* dan juga bertindak sebagai media untuk pengujian dan perhitungan performa dari kompresi *JSON*.

Sebagai panduan lebih lanjut, tabel berikut ini menyajikan detail tentang modul, layanan *microservice*, dan elemen *frontend* yang akan digunakan. Dalam konteks modul, kompresi Gzip, kompresi *HPack*, dan data *JSON* menjadi fokus utama. Layanan *microservice* mencakup autentikasi, pengguna, kiriman, interaksi kiriman, tagar, dan kontainer isasi. Sementara itu, *frontend* akan memanfaatkan antarmuka pengguna, metrik performa, *framework React*, dan *Tailwind* CSS. Detail 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 *microservice*.

Tabel 3.1 Data Analisa Kebutuhan

Modul	***Microservice***	***Frontend***
Kompresi Gzip	Autentikasi	Antarmuka Pengguna
Kompresi *HPack*	Pengguna	Metrik Performa
Data *JSON*	Kiriman	*Framework React*
Perhitungan performa	Interaksi Kiriman	*Framework Tailwind* CSS
Pilihan metode kompresi	Tagar	Kontainer isasi
	Pengguna	
	Feed Beranda	
	Kontainer isasi	

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, kiriman, interaksi kiriman atau komentar, tagar, Analitik Pengguna, Analitik Kiriman, dan *Feed* Beranda. Semua layanan *containerized* 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 *containerized* menggunakan *Docker*.

Kebutuhan fitur *frontend* adalah sebagai berikut:

a. Autentikasi:
    a) Pendaftaran pengguna.
    b) *Login*.
    c) Verifikasi *token* pengguna.
    d) Pemulihan *password*.
    e) Reset *password* pengguna.
    f) *Logout* pengguna.
b. Manajemen Profil Pengguna dan Teman:
    a) Tampilan profil pengguna.
    b) Memperbarui profil pengguna.
    c) Daftar teman pengguna.
    d) Mengirim, menerima, dan membatalkan permintaan pertemanan.
    e) Menghapus teman dari daftar teman pengguna.
    f) Permintaan pertemanan yang tertunda.

c. Kiriman dan Interaksi:
    a) Membuat kiriman baru.
    b) Tampilan detail kiriman.
    c) Memperbarui kiriman.
    d) Menghapus kiriman.
    e) Pencarian kiriman.
    f) Menyukai dan tidak menyukai kiriman.
    g) Tampilan kiriman yang disukai oleh pengguna.
    h) Menambahkan dan menghapus komentar pada kiriman.
    i) Memperbarui komentar.
    j) Tampilan komentar untuk kiriman.
d. Tagar:
    a) Menambahkan dan menghapus tagar pada kiriman.
    b) Tampilan tagar yang terkait dengan kiriman.
    c) Memperbarui tagar.
e. Analitik:
    a) Pencatatan informasi *login* pengguna.
    b) Tampilan catatan *login* pengguna.
    c) Pencatatan analitik pencarian pengguna.
    d) Tampilan analitik pencarian pengguna.
    e) Tampilan jumlah tampilan, suka, dan komentar untuk kiriman.
f. *Feed* Beranda:
    a) Tampilan *feed* beranda pengguna.

3.3 Desain Sistem

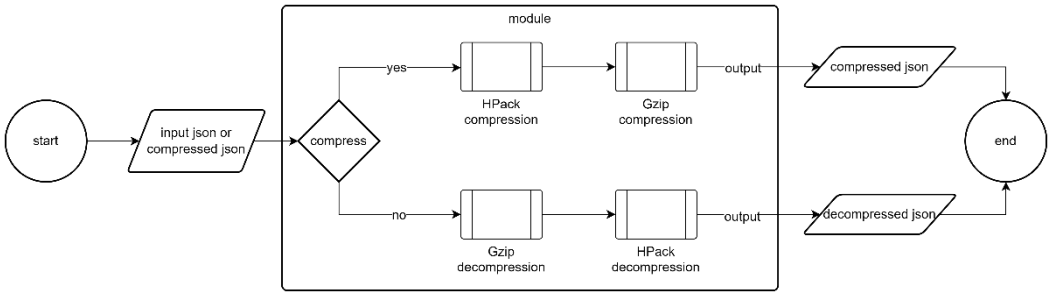
Fase ini merupakan tahap kunci dalam penelitian karena pada tahap ini kerangka kerja aplikasi akan dirancang. Fokus utama dari tahap ini adalah mendiskusikan bagaimana desain sistem dilakukan untuk mencapai optimalisasi aplikasi dengan arsitektur *microservice* menggunakan HPack dan Gzip. Selain itu, 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. kompresi(*json Object*): Melakukan proses kompresi pada objek *JSON*. Fungsi ini mengembalikan sebuah *promise* yang berisi data yang telah dikompres.

b. dekompresi(*compressed Object*): 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. *Invalid Algorithm Error*: Jenis kesalahan ini dilempar jika pengguna mencoba menggunakan algoritma yang tidak didukung.
b. *Compression Error*: Jenis kesalahan ini dilempar jika terjadi masalah selama proses kompresi.
c. *Decompression Error*: 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. Modul Kompresi: Modul ini akan mengandung fungsi kompres yang menerima objek *JSON* dan mengembalikan *promise* dengan data yang telah dikompres.
b. Modul Dekompresi: 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 *microservice* ini, terdapat beberapa layanan yang perlu diperhatikan. layanan tersebut antara lain Layanan Autentikasi, Layanan Pengguna, Layanan Pos, Layanan Interaksi, Layanan Tagar, Layanan Analitik Pengguna, Layanan Analitik Pos, dan Layanan Beranda. Setiap Layanan memiliki peran dan tanggung jawabnya sendiri dalam menjalankan fungsi-fungsi khusus seperti manajemen pengguna, kiriman, interaksi, tagar, dan analitik.

3.3.2.1 Layanan Identifikasi

Layanan Identifikasi adalah sebuah sistem yang merangkum berbagai fungsi penting dalam pengelolaan identitas pengguna. Fungsi-fungsi tersebut meliputi registrasi pengguna baru, proses *login*, verifikasi *token* keamanan, pemulihan kata sandi yang hilang atau lupa, serta proses *logout*. Layanan ini bertujuan untuk menjaga keamanan dan integritas data pengguna, serta memfasilitasi interaksi pengguna dengan sistem atau platform secara aman dan efisien. Di balik semua ini, tujuan utama Layanan Identifikasi adalah untuk melindungi identitas dan privasi pengguna, dan memastikan bahwa setiap transaksi dan interaksi yang terjadi adalah sah dan valid.

3.3.2.2 Layanan Pengguna

Definisi Layanan: Layanan Pengguna adalah suatu sistem yang bertanggung jawab dalam mengelola berbagai aspek dari profil pengguna. Profil ini mencakup sejumlah elemen penting seperti biografi pengguna, nama pengguna, nama lengkap, foto sampul, serta daftar teman. Fungsi ini memungkinkan pengguna untuk melihat dan memodifikasi profil mereka sendiri atau memeriksa profil teman mereka.

Selain itu, Layanan Pengguna juga memiliki tugas dalam mengelola permintaan pertemanan. Ini berarti bahwa layanan ini bertanggung jawab dalam mengatur dan memproses permintaan pertemanan yang dikirimkan oleh pengguna kepada pengguna lainnya. Dalam hal ini, layanan ini berperan penting dalam memfasilitasi interaksi dan jaringan sosial antara pengguna. Tujuan utama Layanan

Pengguna adalah untuk memastikan bahwa setiap pengguna memiliki pengalaman yang baik dan personal dalam berinteraksi dengan sistem atau platform, serta memastikan bahwa pengguna dapat mengelola dan memperbarui profil mereka dengan mudah dan efisien.

3.3.2.3 Layanan Kiriman

Layanan Kiriman adalah sistem yang ditujukan untuk mengelola berbagai aspek dari kiriman pengguna. Hal ini mencakup proses pembuatan baru, pengeditan konten yang sudah ada, penampilan kiriman ke publik, penghapusan kiriman yang tidak diinginkan, serta pencarian kiriman tertentu. Layanan ini dirancang untuk memfasilitasi pengguna dalam berbagi, mencari, dan menginteraksi dengan konten.

Salah satu fitur unik dari Layanan Kiriman adalah kemampuannya untuk menyimpan "tanggal interaksi terakhir" untuk setiap kiriman. "Tanggal interaksi terakhir" ini akan diperbarui setiap kali terjadi interaksi baru dengan kiriman tersebut, baik itu komentar, *like*, atau jenis interaksi lainnya. Dengan demikian, Layanan Kiriman memungkinkan pengguna untuk melacak dan memahami dinamika interaksi dengan kiriman mereka. Tujuan utama dari layanan ini adalah untuk membuat proses berbagi dan menemukan konten menjadi semakin mudah, efisien, dan interaktif bagi semua pengguna.

3.3.2.4 Layanan Interaksi

Layanan Interaksi adalah suatu sistem yang diperuntukkan untuk mengatur dan mengelola berbagai bentuk interaksi pengguna dengan kiriman atau komentar. Ini mencakup berbagai jenis respons, seperti mengekspresikan 'suka' terhadap suatu kiriman atau komentar, dan memberikan komentar sebagai bentuk interaksi lebih lanjut.

Fungsi dari Layanan Interaksi ini penting dalam mendorong partisipasi pengguna dan memfasilitasi dialog atau diskusi yang dinamis antara pengguna. Dengan memanfaatkan layanan ini, pengguna dapat dengan mudah mengekspresikan pendapat atau perasaan mereka terhadap konten yang mereka temui, serta berinteraksi dengan pengguna lain dalam konteks yang lebih luas. Tujuan utama dari Layanan Interaksi ini adalah untuk mendorong komunikasi yang

aktif dan positif antara pengguna, dan untuk membantu dalam pembentukan komunitas yang terlibat dan berinteraksi dalam platform atau sistem tersebut.

3.3.2.5 Layanan Tagar

Definisi Layanan: Layanan Tagar adalah sebuah sistem yang ditujukan untuk mengelola tagar atau '*hashtags*' yang terkait dengan suatu kiriman. Ini mencakup berbagai proses seperti pembuatan tagar baru, pengeditan tagar yang telah ada, penampilan tagar di kiriman, dan penghapusan tagar jika diperlukan.

Layanan Tagar memainkan peran penting dalam membantu pengguna untuk mengategorikan dan menemukan konten yang relevan berdasarkan topik atau tema tertentu. Melalui penggunaan tagar, pengguna dapat dengan mudah mencari dan menjelajahi konten yang terkait dengan minat mereka, serta berpartisipasi dalam diskusi atau tren yang sedang berlangsung.

Tujuan utama dari Layanan Tagar adalah untuk memfasilitasi organisasi dan penemuan konten, serta mendorong interaksi dan partisipasi pengguna dalam diskusi atau tren topik yang sedang berlangsung. Dengan cara ini, layanan ini mendukung pembentukan komunitas yang dinamis dan terlibat, serta memungkinkan pengguna untuk berbagi dan menemukan konten dengan cara yang lebih terstruktur dan relevan.

3.3.2.6 Layanan Analitik Pengguna

Layanan Analitik Pengguna adalah sistem yang dirancang untuk mencatat dan menampilkan analitik yang spesifik untuk pengguna, termasuk namun tidak terbatas pada informasi *login* dan analitik pencarian.

Layanan ini membantu dalam memantau dan menganalisis aktivitas pengguna dalam platform, seperti frekuensi dan waktu *login*, serta pola pencarian mereka. Dengan demikian, pengguna dapat mendapatkan pemahaman yang lebih baik tentang bagaimana mereka menggunakan platform dan cara mereka berinteraksi dengan konten yang tersedia.

Selain itu, Layanan Analitik Pengguna juga penting bagi pemilik platform atau sistem, karena dapat membantu mereka memahami pola dan perilaku

pengguna, memantau performa platform, dan merencanakan peningkatan atau modifikasi yang mungkin diperlukan untuk meningkatkan pengalaman pengguna.

3.3.2.7 Layanan Analitik Kiriman

Layanan Analitik Pos adalah suatu sistem yang dirancang khusus untuk mencatat dan menampilkan analitik yang terkait dengan interaksi pengguna terhadap kiriman. Analitik ini mencakup berbagai metrik, seperti jumlah tampilan suatu kiriman, jumlah 'suka' yang diterima, dan jumlah komentar yang diberikan pengguna.

Layanan ini sangat penting untuk memberikan pemahaman yang mendalam tentang bagaimana konten dipahami dan diterima oleh pengguna. Dengan menganalisis data ini, pengguna dan pemilik platform dapat memahami apa yang paling menarik bagi pengguna, serta bagaimana mereka berinteraksi dengan konten yang ada.

Selain itu, Layanan Analitik Pos juga sangat bermanfaat untuk pemilik platform dalam mengevaluasi efektivitas dan popularitas konten yang dikirim, sehingga mereka dapat merencanakan strategi konten yang lebih baik di masa mendatang.

3.3.2.8 Layanan Beranda

Layanan Beranda adalah sistem yang dirancang untuk mengumpulkan dan menampilkan kiriman dari teman pengguna serta kiriman yang telah diberi interaksi oleh teman mereka. Kiriman-kiriman ini disusun dan ditampilkan berdasarkan "tanggal interaksi terakhir", yang memungkinkan pengguna untuk melihat konten yang paling baru atau paling relevan berdasarkan interaksi terakhir.

Layanan ini memegang peran penting dalam menciptakan pengalaman pengguna yang dinamis dan relevan, dengan memastikan bahwa pengguna melihat konten yang paling relevan dengan jaringan sosial mereka. Dengan kata lain, Layanan Beranda membantu pengguna tetap *up-to-date* dengan aktivitas dan interaksi teman mereka.

Dengan mengikuti metodologi desain tingkat tinggi ini, aplikasi media sosial berbasis *microservice* dapat dikembangkan menggunakan Node.js,

PostgreSQL, Express.js, dan *Sequelize*. *dockerized microservice* memudahkan proses *deployment* dan pengelolaan. Setiap *microservice* 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*, 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 kiriman. Data ini akan diambil melalui API *endpoint* yang disediakan oleh *Microservice* kiriman analitik.

3.3.3.4 Penggunaan *Docker*

Aplikasi *frontend* akan containerized menggunakan *Docker*. Hal ini memudahkan dalam pengelolaan dan pengiriman aplikasi serta memastikan probabilitas 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 Ringkasan 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 Detail 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 kiriman yang mengelola kiriman, termasuk membuat, mengedit, melihat, menghapus, dan mencari kiriman.

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 containerized* 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 *microservice* dan *frontend containerized* 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 *containerized* 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 fondasi 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.