Analisis Performa *Rendering* Antara *Microservice* dan *Serverless Firebase Architecture* pada Aplikasi Berbasis *Website* menggunakan *Google Lighthouse*

Yusril Adriansyah Putra[1], Dwi Fatrianto Suyatno[2].

[1,2] Sistem Informasi, Fakultas Teknik, Universitas Negeri Surabaya
[1]yusril.19022@mhs.unesa.ac.id
[2]dwifatrianto@unesa.ac.id

***Abstrak*— Pengguna *internet* akan terus bertambah seiring berjalannya waktu, semakin banyaknya pengguna *internet* akan juga berdampak pada performa *website*. Arsitektur *Microservice* dan arsitektur *Serverless* merupakan dua arsitektur baru untuk pengembangan aplikasi berbasis *website*. Tujuannya penelitian ini adalah melakukan analisa terhadap performa *website* yang menggunakan kedua arsitektur tersebut dan juga mengetahui kekurangan dan kelebihan dari sisi performa *rendering* diantara arsitektur *Microservice* dan *Serverless* pada aplikasi berbasis *website*. Penelitian ini akan menggunakan *tools Google Lighthouse* yang menggunakan model pengujian *Web Vitals*. Terdapat 6 aspek yang akan di uji pada penelitian ini yaitu *First Contentful Paint* (FCP), *First Input Delay* (FID), *Speed index*, *Total Blocking Time* (TBT), *Largest Contentful Paint* (LCP), dan juga *Cumulative Layout Shift* (CLS). Hasil penelitian ini menunjukkan bahwa Arsitektur *Microservice* lebih unggul karena memiliki nilai yang lebih tinggi dalam 4 aspek yaitu *First Input Delay* (FID), *Total Blocking Time* (TBT), *Largest Contentful Paint* (LCP), dan juga *Cumulative Layout Shift* (CLS) dibandingkan dengan arsitektur *Serverless* yang lebih unggul hanya dalam 1 aspek yaitu *Speed Index*. Akan tetapi pada aspek *First Contentful Paint* (FCP) pada kedua arsitektur memiliki nilai yang sama. Selain aspek tersebut, penilaian keseluruhan dari *Google Lighthouse* juga menunjukkan arsitektur *Microservice* memiliki nilai lebih banyak dibandingkan dengan arsitektur *Serverless*.**

***Kata Kunci*— *Microservice, Serverless, Firebase, Web Vitals, Google Lighthouse*.**

I. PENDAHULUAN

Perkembangan teknologi yang begitu pesat memiliki banyak manfaat bagi masyarakat dalam berbagai bidang. Pada bidang industri, kemajuan teknologi dapat membantu perusahaan dalam meningkatkan produksi [1]. Pengguna internet di Indonesia kurang lebih mencapai 73.7% dari total populasi, angka ini merupakan hasil survey yang dilakukan oleh APJII (Asosiasi Penyelenggara Jasa Internet Indonesia) periode 2019-2020 [2].

Website sudah berkembang sangat pesat. Sekarang, website digunakan tidak hanya untuk media berbagi informasi atau berita [3]. Layaknya sebuah aplikasi pada komputer dan handphone kita, website dapat digunakan untuk komunikasi secara real-time, mendengarkan sebuah lagu, hingga photo editing kini juga dapat dilakukan pada sebuah website [4]. Dari hal tersebut membuktikan bahwa perkembangan teknologi tidak pernah menurun dan akan terus meningkat. Namun seiring meningkatnya jumlah pengguna internet menyebabkan dampak performa pada website [5]. Semakin tinggi kualitas website, semakin baik persepsi pelanggan terhadap perusahaan sehingga meningkatkan rasa kepuasan pelanggan, kepuasan pelanggan akan membangun loyalitas pelanggan yang menjadi modal utama untuk menang dan bertahan dalam persaingan bisnis [6].

Sebagian aplikasi yang telah dirancang dan dibangun saat ini masih menggunakan arsitektur Monolithic, yaitu arsitektur yang menggambarkan sebuah aplikasi yang menjalankan semua logika dalam satu server aplikasi [7]. Seiring berkembangnya waktu dan masalah yang ada pada monolithic, dibuat lah arsitektur baru yaitu arsitektur Microservice. Arsitektur Microservice adalah gaya arsitektur perangkat lunak yang memerlukan pemecahan aplikasi secara bisnis [8]. Microservice pertama kali dikenalkan pada tahun 2011 [9].

Selain Microservice, terdapat juga arsitektur modern lain yaitu Serverless. Arsitektur Serverless merupakan arsitektur yang digunakan untuk mengembangkan berbagai aplikasi dan layanan tanpa perlu mengelola infrastruktur. Server masih menjalankan suatu aplikasi, tetapi semua manajemen server dilakukan oleh penyedia layanan sehingga pengembang hanya perlu fokus pada penulisan kode yang melayani pelanggan [10]. Arsitektur Serverless pertama kali dipopulerkan oleh AWS pada tahun 2014, lalu perusahaan besar seperti Microsoft dan Google mulai mengimplementasikan metode yang sama pada tahun 2016 [11].

Dalam pengujian performa terdapat sebuah tools dari Google bernama Lighthouse. Dalam tools tersebut pengujian menerapkan metode Web Vitals sebagai acuan penilaiannya. Dalam metode Web Vitals, terdapat parameter-parameter yaitu : First Contentful Paint, Time to Interactive, Total Blocking Time, Largest Contentful Paint, dan juga Cumulative Layout Shift [12]. Selain Web Vitals, Lighthouse juga menambahkan parameter tambahan yaitu Speed Index [13].

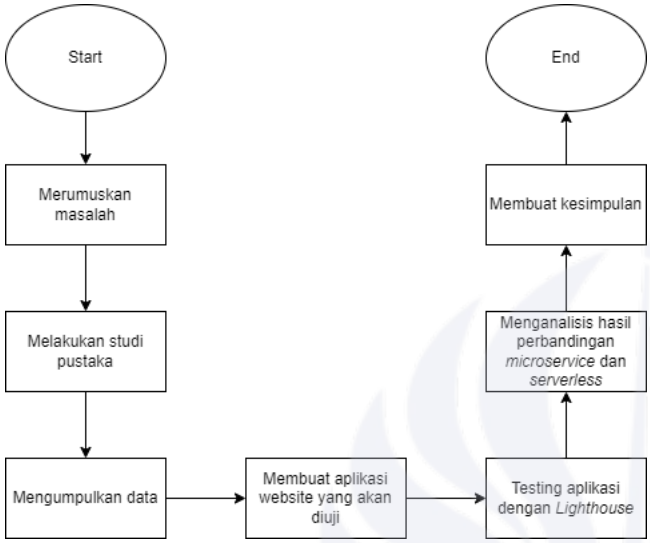
Microservice dan Serverless architecture merupakan arsitektur yang modern yang memiliki konsep pengembangan aplikasi yang berbeda sehingga diperlukannya sebuah analisa performa terhadap kedua arsitektur tersebut. Maka dari itu penulis membuat penelitian dengan judul "Analisis Performa Rendering antara Microservice dan Serverless Firebase Architecture pada Aplikasi Berbasis Website menggunakan Google Lighthouse".

II. METODE PENELITIAN

Cara paling mudah untuk memenuhi persyaratan format penulisan adalah dengan menggunakan dokumen ini sebagai template. Kemudian ketikkan teks Anda ke dalamnya

A. Alur penelitian

Berikut adalah alur penelitian yang akan dilakukan oleh peneliti untuk melakukan pengujian performa *Rendering*:



Gbr. 1 *Flowchart* Alur Penelitian.

Berikut adalah penjelasan mengenai alur diatas:

1) Pertama merumuskan masalah sesuai dengan latar belakang penelitian.
2) Kedua, penelitian melakukan studi pustaka guna mencari referensi teori yang akan diuji.
3) Setelah itu mengumpulkan data yang akan di *render* untuk aplikasi yang diuji nantinya.
4) Selanjutnya membuat aplikasi yang akan diuji dengan menggunakan arsitektur *Microservice* dan *Serverless*.
5) Kemudian melakukan testing dengan *Google Lighthouse* pada aplikasi yang dirancang. Testing nantinya akan dilakukan 5 kali untuk mendapatkan hasil yang maksimal.
6) Setelah hasil dari *testing* atau audit keluar, selanjutnya adalah menganalisis dan membandingkan hasil dari 5 kali pengujian sebelumnya.
7) Terakhir, akan dilakukan rangkuman ke dalam kesimpulan sebagai kajian akhir penulis.

B. Pengumpulan Data

Teknik pengumpulan data yang dilakukan pada penelitian ini adalah dengan mengambil salah satu *dataset* dari sebuah platform *open source* penyedia *dataset* bernama *Kaggle*. *Dataset* yang digunakan berisi mengenai *review* kepuasan konsumen terhadap produk bernama *Amazon Alexa*. Data tersebut berjumlah 1000 dimana setiap data berisikan tanggal, rating, varian produk, dan juga *feedback* yang diberikan.

Berikut adalah contoh data *review* dari produk *Amazon Alexa*:

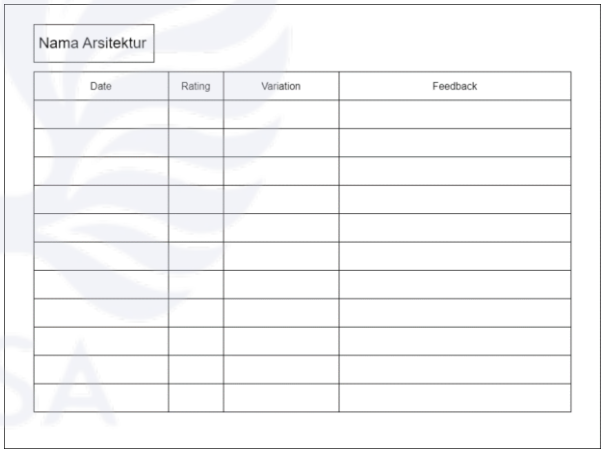
TABEL I
CONTOH DATA *CUSTOMER REVIEW* PRODUK *AMAZON ALEXA*

Date	Rating	Variation	Feedback
31-Jul-18	5	Charcoal Fabric	Love my Echo!
31-Jul-18	5	Charcoal Fabric	Loved it!
31-Jul-18	4	Heather Gray Fabric	Really happy with this purchase. Great speaker and easy to set up.
31-Jul-18	5	Heather Gray Fabric	looks great
31-Jul-18	5	Charcoal Fabric	Music

Data yang didapat nantinya akan dimasukkan ke dalam *database Firestore* untuk percobaan arsitektur *Serverless* dan *database PostgreSQL* yang di *deploy* ke *ElephantSQL* untuk percobaan arsitektur *Microservice*.

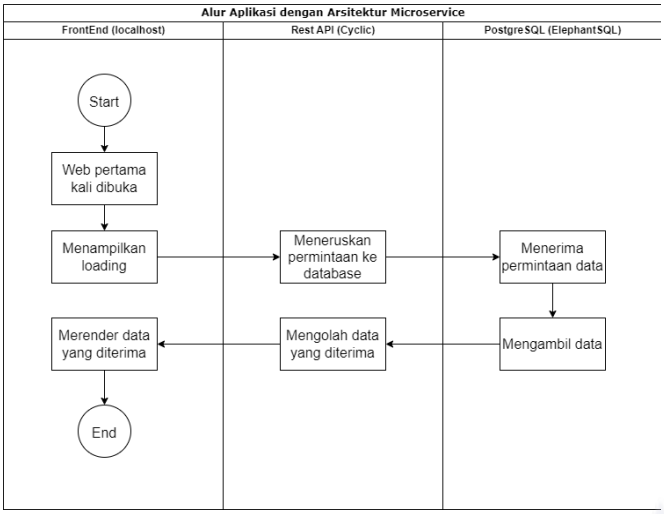
C. Aplikasi yang Akan di Uji

Akan terdapat 2 aplikasi yang sama dengan nama *Amazon Alexa Reviews*, namun arsitektur yang digunakan berbeda. Aplikasi yang diuji akan melakukan *render* sekaligus semua data yang sudah dikumpulkan sebelumnya.



Gbr. 2 Tampilan Aplikasi yang Akan Diuji.

Berikut merupakan alur pada aplikasi yang menggunakan arsitektur *Microservice*:



Gbr. 3 Alur Aplikasi Arsitektur *Microservice.*

Berikut adalah keterangan pada aplikasi yang menggunakan arsitektur *Microservice*:

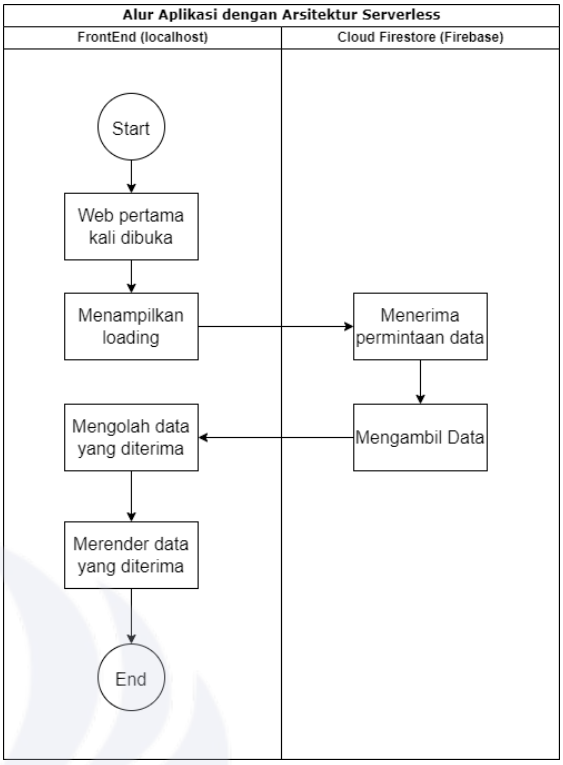
1) Pada sisi *frontend*, aplikasi dikembangkan dengan *framework ReactJs*.
2) Pada sisi *backend*, aplikasi dikembangkan menggunakan *framework ExpressJS* sebagai *API (Application Program Interface)*, dan *NodeJS* sebagai *runtime environment*.
3) Pada penyimpanan *database*, dikembangkan menggunakan *PostgreSQL*.
4) Pada tahap *deployment*, API di *deploy* menggunakan layanan *Cyclic*, sedangkan *database* di *deploy* dengan layanan *ElephantSQL*.

Berikut merupakan diagram *Physical Data Model* (PDM) untuk *database* pada arsitektur *Microservice:*

Reviews		
PK	id	Integer
	rating	VARCHAR(255)
	date	VARCHAR(255)
	variation	VARCHAR(255)
	verified_reviews	VARCHAR(255)

Gbr. 4 Diagram PDM pada *Microservice*.

Berikut merupakan alur pada aplikasi yang menggunakan arsitektur *Serverless*:



Gbr. 5 Alur Aplikasi Arsitektur *Serverless*.

Berikut adalah keterangan pada aplikasi yang menggunakan arsitektur *Serverless*:

1) Pada sisi *frontend*, aplikasi dikembangkan dengan *framework ReactJs*.
2) Pada penyimpanan *database*, dikembangkan menggunakan salah satu layanan *Firebase*, yaitu *Cloud Firestore*.
3) Karena tidak membutuhkan *backend API*, maka tidak ada *deployment* pada arsitektur *Serverless*.

Berikut merupakan diagram *Physical Data Model* (PDM) untuk database pada arsitektur *Serverless:*

Reviews		
PK	id	STRING
	rating	STRING
	date	STRING
	variation	STRING
	verified_reviews	STRING

Gbr. 5 Diagram PDM pada *Serverless.*

D. Testing Aplikasi dengan Google Lighthouse

Setelah kedua aplikasi sudah dikembangkan, langkah selanjutnya adalah melakukan pengujian performa *rendering* dengan bantuan *tools Google Lighthouse*. Pada proses

pengujiannya, *Google Lighthouse* menggunakan parameter yang ada pada model *Web Vitals* dengan 1 parameter tambahan yaitu *Speed Index*. Dilansir dari halaman dokumentasi resmi *Google Lighthouse* [13], Berikut adalah parameter yang diukur beserta pengkategorian waktu rendering pada *Google Lighthouse*:

TABEL II
PARAMETER DAN PENGKATEGORIAN NILAI PADA *GOOGLE LIGHTHOUSE*.

Parameter	Kategori	Waktu *Rendering*
First Contentful Paint	Cepat (Hijau)	0 - 1.8 *seconds*
	Sedang (Orange)	1.9 - 3 *seconds*
	Lambat (Merah)	> 3 *seconds*
Time to Interactive	Cepat (Hijau)	0 - 3.8 *seconds*
	Sedang (Orange)	3.9 - 7.3 seconds
	Lambat (Merah)	> 7.3 *seconds*
Cumulative Layout Shift	Cepat (Hijau)	0 - 0.1 %
	Sedang (Orange)	0.2 - 0.25 %
	Lambat (Merah)	> 0.25 %
Total Blocking Time	Cepat (Hijau)	0 - 200 *milliseconds*
	Sedang (Orange)	201 - 600 *milliseconds*
	Lambat (Merah)	> 600 *milliseconds*
Largest Contentful Paint	Cepat (Hijau)	0 - 2.5 *seconds*
	Sedang (Orange)	2.6 - 4 *seconds*
	Lambat (Merah)	> 4 *seconds*
Speed Index	Cepat (Hijau)	0 - 3.4 *seconds*
	Sedang (Orange)	3.5 - 5.8 *seconds*
	Lambat (Merah)	> 5.8 *seconds*
Final Score	Cepat (Hijau)	90 - 100
	Sedang (Orange)	50 - 89
	Lambat (Merah)	0 - 49

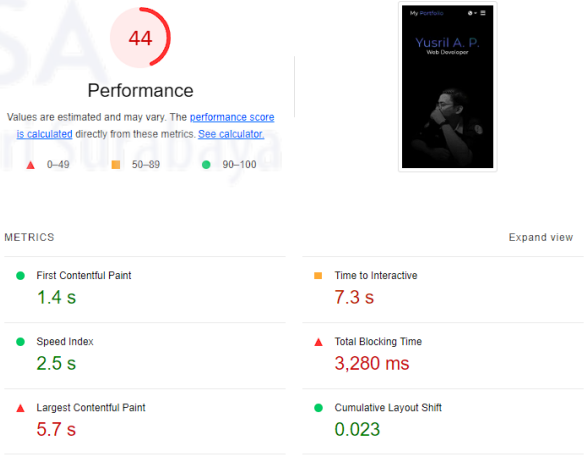
Dilansir dari sumber yang sama, setelah nilai data uji didapatkan, langkah selanjutnya *Google Lighthouse* menghitung nilai keseluruhan performa berdasarkan pembobotan seperti berikut:

TABEL III
BOBOT PARAMETER

Parameter	Bobot
First Contentful Paint	10 %
Time to Interactive	10 %
Cumulative Layout Shift	15 %
Total Blocking Time	10 %
Largest Contentful Paint	30 %
Speed Index	25 %
Total	**100%**

Setelah semua parameter dinilai dan dikategorikan, selanjutnya *Google Lighthouse* akan menampilkan keseluruhan output setiap parameter dan hasil akhir penilaian dari bobot parameter.

Berikut adalah contoh output dari *Google Lighthouse*:



Gbr. 6 Contoh output dari *Google Lighthouse*.

E. Menganalisis Hasil Perbandingan

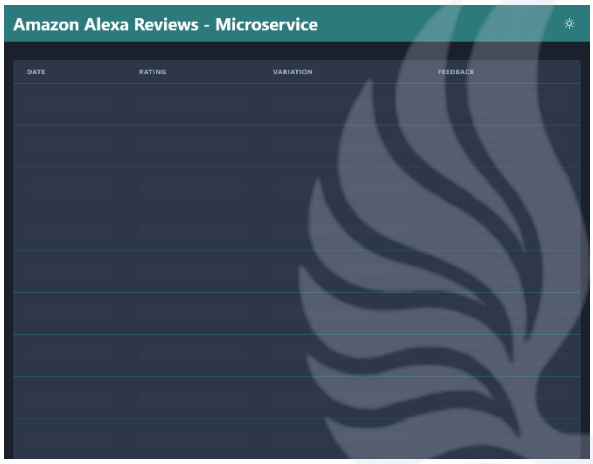
Setelah semua data sudah diperoleh dari hasil kedua arsitektur tersebut, peneliti dapat menarik kesimpulan dari hasil analisa berdasarkan pengukuran menggunakan parameter yang ditetapkan sebagai nilai pada masing-masing sisi *render*.

III. HASIL PENELITIAN DAN PEMBAHASAN

A. Pengembangan Aplikasi yang Akan Diuji

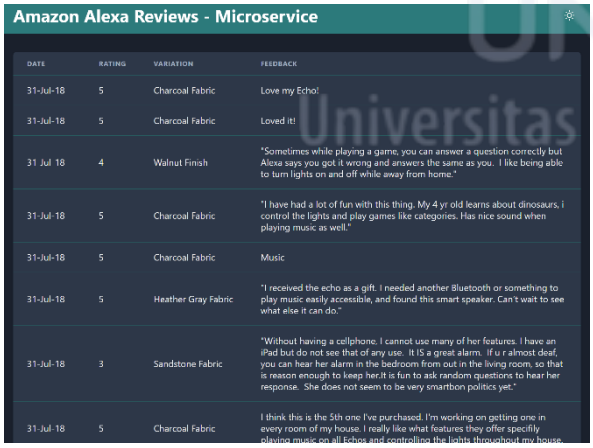
Peneliti akan mengambil data *consumer review* dari produk *Amazon Alexa*, data diambil dari platform Kaggle. Data tersebut memiliki format tsv (Tab-separated values). Setelah data didapatkan, selanjutnya peneliti akan membuat *script* dengan *javascript* untuk melakukan *data injection* ke *ElephantSQL* dan *Cloud Firestore* sebagai *database* yang akan digunakan nantinya untuk skenario pengujian masing-masing arsitektur.

Berikut merupakan tampilan aplikasi dari masing-masing arsitektur:



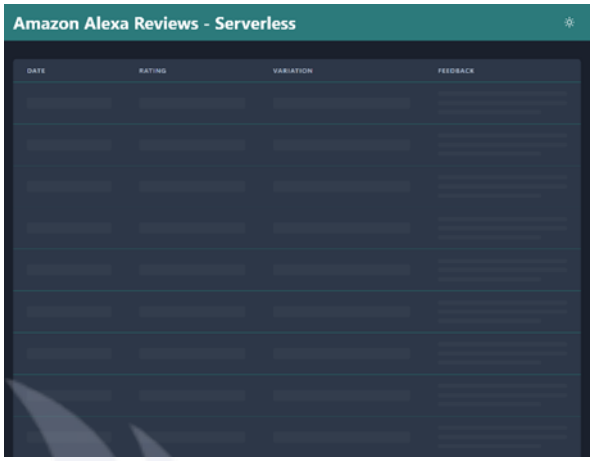
Gbr. 7 Tampilan *Loading* Aplikasi *Microservice*.

Pada Gambar 7 menampilkan aplikasi pada arsitektur *Microservice* dalam keadaan *loading*.



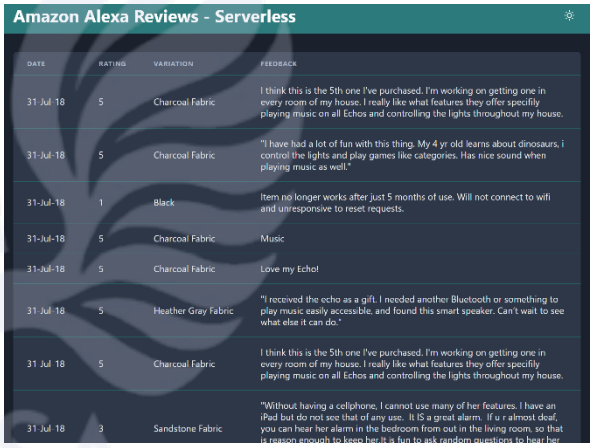
Gbr. 8 Tampilan Aplikasi *Microservice*.

Pada Gambar 8 menunjukkan aplikasi dengan arsitektur *Microservice* yang dikembangkan menggunakan *ReactJS* untuk bagian *FrontEnd*, *ExpressJS* untuk *REST API backend Microservice*, *PostgreSQL* yang sudah *dideploy* ke *ElephantSQL* sebagai *database* yang sudah terdapat data pengujian dari *data injection* sebelumnya.



Gbr. 9 Tampilan *Loading* Aplikasi *Serverless*.

Pada Gambar 9 menampilkan aplikasi pada arsitektur *Serverless* dalam keadaan *loading*.

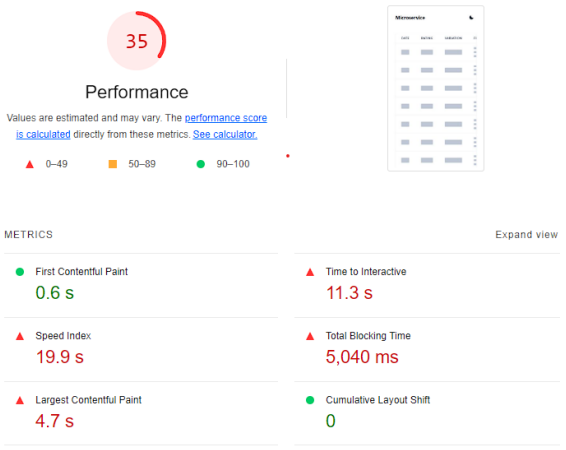


Gbr. 10 Tampilan Aplikasi *Serverless*.

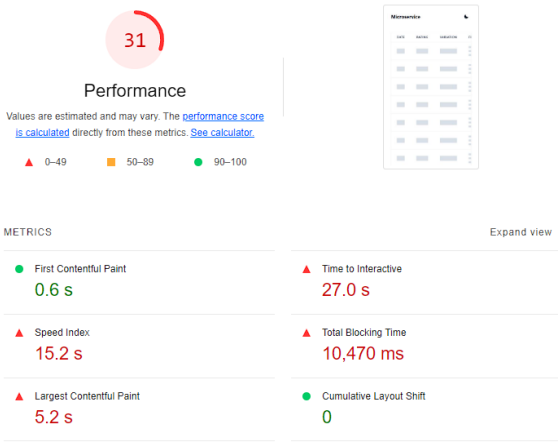
Pada Gambar 10 menunjukkan aplikasi dengan arsitektur *Serverless* dikembangkan menggunakan *ReactJS* untuk bagian *FrontEnd*, dan juga *Cloud Firestore* sebagai *database* yang juga sudah terdapat data pengujian dari *data injection* sebelumnya.

B. Hasil Penelitian Arsitektur Microservice

Berikut adalah hasil pengujian ke-1 dari performa arsitektur *Microservice*:



Gbr. 11 Hasil Pengujian ke-1 Arsitektur *Microservice*.



Gbr. 12 Hasil Pengujian ke-2 Arsitektur *Microservice*.

Pada Gambar 11 menunjukkan hasil pengujian ke-1, Arsitektur *Microservice* mendapatkan nilai 35 pada keseluruhan performa yang masuk ke dalam kategori lambat. Pada aspek *First Contentful Paint* (FCP) menunjukkan 0,6 *seconds*, untuk aspek *Time to Interactive* atau *First Input Delay* (FID) menunjukkan 11,3 *seconds*, untuk aspek *Speed Index* menunjukkan 19,9 *seconds*, untuk aspek *Total Blocking Time* (TBT) menunjukkan 5.040 *miliseconds*, untuk aspek *Large Contentful Paint* (LCP) menunjukkan 4,7 *seconds*, sedangkan untuk *Cumulative Layout Shift* (CLS) menunjukkan 0 %.

Untuk pengkategorian dari hasil pengujian ke-1 dapat dilihat pada Tabel IV berikut:

TABEL IV
PENGKATEGORIAN HASIL PENGUJIAN KE-1 *MICROSERVICE*

Aspek	Nilai	Kategori
FCP	0,6	Cepat (Hijau)
FID	11,3	Lambat (Merah)
Speed Index	19,9	Lambat (Merah)
TBT	5.040	Lambat (Merah)
LCP	4,7	Lambat (Merah)
CLS	0	Cepat (Hijau)
Final Score	35	Lambat (Merah)

Berikut adalah hasil pengujian ke-2 dari performa arsitektur *Microservice*:

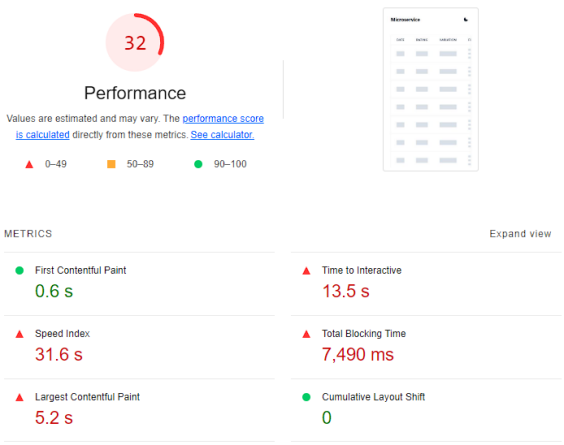
Pada Gambar 12 menunjukkan hasil pengujian ke-2, Arsitektur *Microservice* mendapatkan nilai 31 pada keseluruhan performa yang masuk ke dalam kategori lambat. Pada aspek *First Contentful Paint* (FCP) menunjukkan 0,6 *seconds*, untuk aspek *Time to Interactive* atau *First Input Delay* (FID) menunjukkan 27,0 *seconds*, untuk aspek *Speed Index* menunjukkan 15,2 *seconds*, untuk aspek *Total Blocking Time* (TBT) menunjukkan 10.470 *miliseconds*, untuk aspek *Large Contentful Paint* (LCP) menunjukkan 5,2 *seconds*, sedangkan untuk *Cumulative Layout Shift* (CLS) menunjukkan 0 %.

Untuk pengkategorian dari hasil pengujian ke-2 dapat dilihat pada Tabel V berikut:

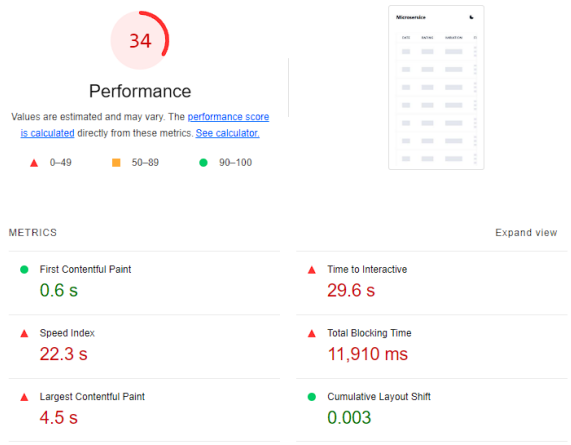
TABEL V
PENGKATEGORIAN HASIL PENGUJIAN KE-2 *MICROSERVICE*

Aspek	Nilai	Kategori
FCP	0,6	Cepat (Hijau)
FID	27,0	Lambat (Merah)
Speed Index	15,2	Lambat (Merah)
TBT	10.470	Lambat (Merah)
LCP	5,2	Lambat (Merah)
CLS	0	Cepat (Hijau)
Final Score	31	Lambat (Merah)

Berikut adalah hasil pengujian ke-3 dari performa arsitektur *Microservice*:



Gbr. 13 Hasil Pengujian ke-3 Arsitektur *Microservice*.



Gbr. 14 Hasil Pengujian ke-4 Arsitektur *Microservice*.

Pada Gambar 13 menunjukkan hasil pengujian ke-3, Arsitektur *Microservice* mendapatkan nilai 32 pada keseluruhan performa yang masuk ke dalam kategori lambat. Pada aspek *First Contentful Paint* (FCP) menunjukkan 0,6 *seconds*, untuk aspek *Time to Interactive* atau *First Input Delay* (FID) menunjukkan 13,5 *seconds*, untuk aspek *Speed Index* menunjukkan 31,6 *seconds*, untuk aspek *Total Blocking Time* (TBT) menunjukkan 7.490 *miliseconds*, untuk aspek *Large Contentful Paint* (LCP) menunjukkan 5,2 *seconds*, sedangkan untuk *Cumulative Layout Shift (CLS)* menunjukkan 0 %.

Untuk pengkategorian dari hasil pengujian ke-3 dapat dilihat pada Tabel VI berikut:

TABEL VI
PENGKATEGORIAN HASIL PENGUJIAN KE-3 *MICROSERVICE*

Aspek	Nilai	Kategori
FCP	0,6	Cepat (Hijau)
FID	13,5	Lambat (Merah)
Speed Index	31,6	Lambat (Merah)
TBT	7.490	Lambat (Merah)
LCP	5,2	Lambat (Merah)
CLS	0	Cepat (Hijau)
Final Score	32	Lambat (Merah)

Berikut adalah hasil pengujian ke-4 dari performa arsitektur *Microservice*:

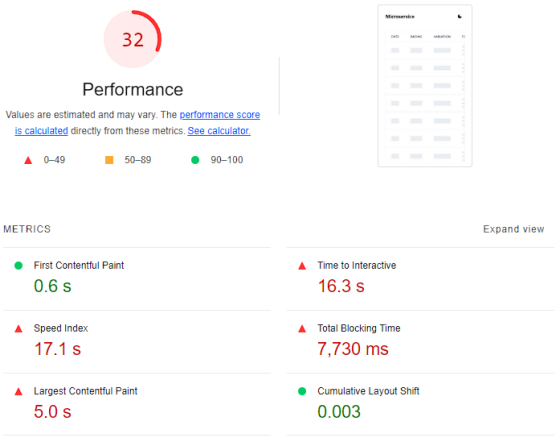
Pada Gambar 14 menunjukkan hasil pengujian ke-4, Arsitektur *Microservice* mendapatkan nilai 34 pada keseluruhan performa yang masuk ke dalam kategori lambat. Pada aspek *First Contentful Paint* (FCP) menunjukkan 0,6 *seconds*, untuk aspek *Time to Interactive* atau *First Input Delay* (FID) menunjukkan 29,6 *seconds*, untuk aspek *Speed Index* menunjukkan 22,3 *seconds*, untuk aspek *Total Blocking Time* (TBT) menunjukkan 11.910 *miliseconds*, untuk aspek *Large Contentful Paint* (LCP) menunjukkan 4,5 *seconds*, sedangkan untuk *Cumulative Layout Shift* (CLS) menunjukkan 0.003 %.

Untuk pengkategorian dari hasil pengujian ke-4 dapat dilihat pada Tabel VII berikut:

TABEL VII
PENGKATEGORIAN HASIL PENGUJIAN KE-4 *MICROSERVICE*

Aspek	Nilai	Kategori
FCP	0,6	Cepat (Hijau)
FID	29,6	Lambat (Merah)
Speed Index	22,3	Lambat (Merah)
TBT	11.910	Lambat (Merah)
LCP	4,5	Lambat (Merah)
CLS	0,003	Cepat (Hijau)
Final Score	34	Lambat (Merah)

Berikut adalah hasil pengujian ke-5 dari performa arsitektur *Microservice*:



Gbr. 15 Hasil Pengujian ke-5 Arsitektur *Microservice*.

Pada Gambar 15 menunjukkan hasil pengujian ke-5, Arsitektur *Microservice* mendapatkan nilai 32 pada keseluruhan performa yang masuk ke dalam kategori lambat. Pada aspek *First Contentful Paint* (FCP) menunjukkan 0,6 *seconds*, untuk aspek *Time to Interactive* atau *First Input Delay* (FID) menunjukkan 16,3 *seconds*, untuk aspek *Speed Index* menunjukkan 17,1 *seconds*, untuk aspek *Total Blocking Time* (TBT) menunjukkan 7.730 *miliseconds*, untuk aspek *Large Contentful Paint* (LCP) menunjukkan 5,0 *seconds*, sedangkan untuk *Cumulative Layout Shift* (CLS) menunjukkan 0.003 %.

Untuk pengkategorian dari hasil pengujian ke-5 dapat dilihat pada Tabel VIII berikut:

TABEL VIII
PENGKATEGORIAN HASIL PENGUJIAN KE-5 *MICROSERVICE*

Aspek	Nilai	Kategori
FCP	0,6	Cepat (Hijau)
FID	16,3	Lambat (Merah)
Speed Index	17,1	Lambat (Merah)
TBT	7.730	Lambat (Merah)
LCP	5,0	Lambat (Merah)
CLS	0,003	Cepat (Hijau)
Final Score	32	Lambat (Merah)

Setelah mendapatkan semua hasil dari proses pengujian, selanjutnya akan dilakukan rata-rata setiap parameter dan hasil akhir dari ke-5 proses pengujian sebelumnya. Berikut adalah hasil rata-rata pada arsitektur *Microservice*:

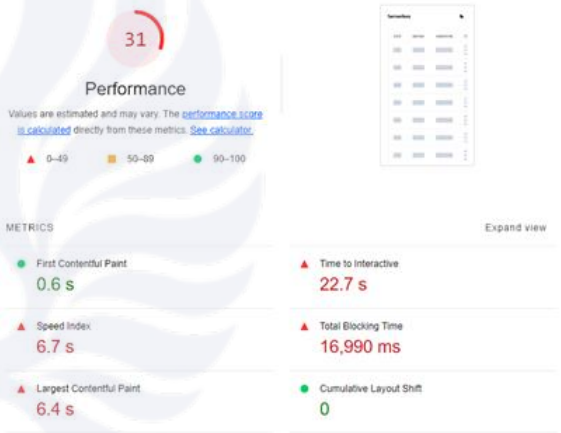
TABEL IX
HASIL RATA-RATA PADA ARSITEKTUR *MICROSERVICE*

Aspek	**Uji ke-1**	**Uji ke-2**	**Uji ke-3**	**Uji ke-4**	**Uji ke-5**	**Rata-Rata**
FCP	0,6	0,6	0,6	0,6	0,6	0,6
FID	11,3	27,0	13,5	29,6	16,3	19,54
Speed Index	19,9	15,2	31,6	22,3	17,1	21,22
TBT	5.040	10.470	7.490	11.910	7,730	8.528
LCP	4,7	5,2	5,2	4,5	5,0	4,92
CLS	0	0	0	0,003	0,003	0,0012
Final Score	35	31	32	34	32	32,8

Dari Tabel IX, menunjukkan arsitektur *Microservice* pada aspek *First Contentful Paint* (FCP) rata-rata memerlukan 0,6 *seconds* yang masuk ke dalam kategori cepat, untuk aspek *First Input Delay* (FID) rata-rata memerlukan 19,54 *seconds* yang masuk ke dalam kategori lambat, pada aspek *Speed Index* rata-rata memerlukan 21,22 *seconds* yang masuk ke dalam kategori lambat, pada aspek *Total Blocking Time* (TBT) rata-rata memerlukan 8.528 *miliseconds* yang masuk kedalam kategori lambat, pada aspek *Large Contentful Paint* (LCP) rata-rata memerlukan 4,92 *seconds* yang masuk ke dalam kategori lambat, untuk aspek *Cumulative Layout Shift* (CLS) rata-rata menunjukkan 0,0012 % yang masuk ke dalam kategori cepat, sehingga keseluruhan *final score* menunjukkan 32.8 yang masuk ke dalam kategori lambat.

C. *Hasil Penelitian Arsitektur Serverless*

Berikut adalah hasil pengujian ke-1 dari performa arsitektur *Serverless*:



Gbr. 16 Hasil Pengujian ke-1 Arsitektur *Serverless*.

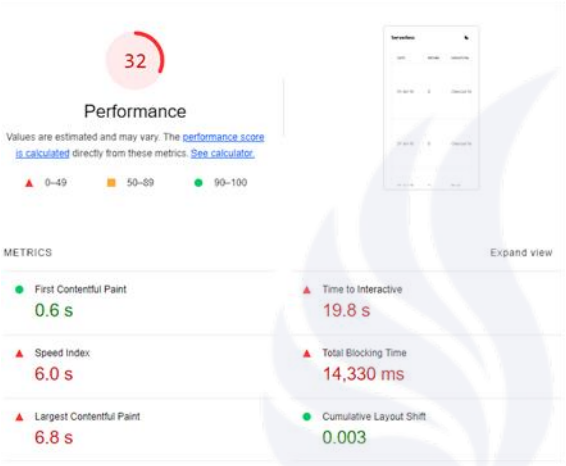
Pada Gambar 16 menunjukkan hasil pengujian ke-1, Arsitektur *Serverless* mendapatkan nilai 32 pada keseluruhan performa yang masuk ke dalam kategori lambat. Pada aspek *First Contentful Paint* (FCP) menunjukkan 0,6 *seconds*, untuk aspek *Time to Interactive* atau *First Input Delay* (FID) menunjukkan 16,3 *seconds*, untuk aspek *Speed Index* menunjukkan 17,1 *seconds*, untuk aspek *Total Blocking Time* (TBT) menunjukkan 7.730 *miliseconds*, untuk aspek *Large Contentful Paint* (LCP) menunjukkan 5,0 *seconds*, sedangkan untuk *Cumulative Layout Shift* (CLS) menunjukkan 0.003 %.

Untuk pengkategorian dari hasil pengujian ke-1 dapat dilihat pada Tabel X berikut:

TABEL X
PENGKATEGORIAN HASIL PENGUJIAN KE-1 *SERVERLESS*

Aspek	Nilai	Kategori
FCP	0,6	Cepat (Hijau)
FID	16,3	Lambat (Merah)
Speed Index	17,1	Lambat (Merah)
TBT	7.730	Lambat (Merah)
LCP	5,0	Lambat (Merah)
CLS	0,003	Cepat (Hijau)
Final Score	32	Lambat (Merah)

Berikut adalah hasil pengujian ke-2 dari performa arsitektur *Serverless*:



Gbr. 17 Hasil Pengujian ke-2 Arsitektur *Serverless*.

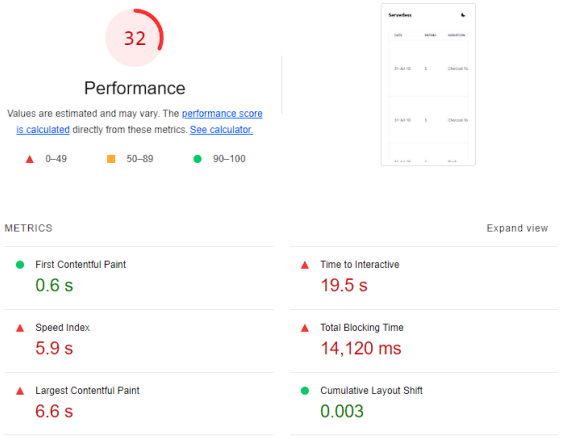
Pada Gambar 17 menunjukkan hasil pengujian ke-2, Arsitektur *Serverless* mendapatkan nilai 32 pada keseluruhan performa yang masuk ke dalam kategori lambat. Pada aspek *First Contentful Paint* (FCP) menunjukkan 0,6 *seconds*, untuk aspek *Time to Interactive* atau *First Input Delay* (FID) menunjukkan 19,8 *seconds*, untuk aspek *Speed Index* menunjukkan 6,0 *seconds*, untuk aspek *Total Blocking Time* (TBT) menunjukkan 14.330 *miliseconds*, untuk aspek *Large Contentful Paint* (LCP) menunjukkan 6,8 *seconds*, sedangkan untuk *Cumulative Layout Shift* (CLS) menunjukkan 0.003 %.

Untuk pengkategorian dari hasil pengujian ke-2 dapat dilihat pada Tabel XI berikut:

TABEL XI
PENGKATEGORIAN HASIL PENGUJIAN KE-2 *SERVERLESS*

Aspek	Nilai	Kategori
FCP	0,6	Cepat (Hijau)
FID	19,8	Lambat (Merah)
Speed Index	6,0	Lambat (Merah)
TBT	14.330	Lambat (Merah)
LCP	6,8	Lambat (Merah)
CLS	0,003	Cepat (Hijau)
Final Score	32	Lambat (Merah)

Berikut adalah hasil pengujian ke-3 dari performa arsitektur *Serverless*:



Gbr. 18 Hasil Pengujian ke-3 Arsitektur *Serverless*.

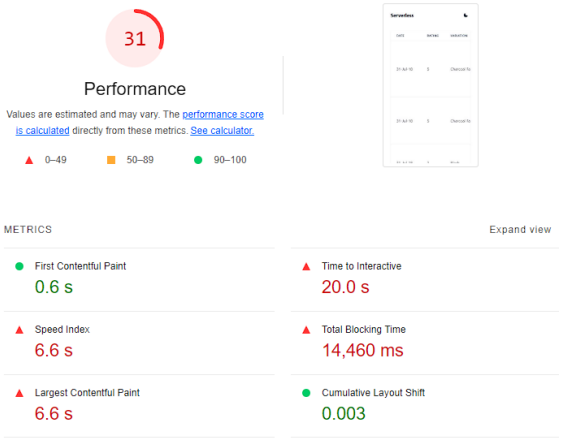
Pada Gambar 18 menunjukkan hasil pengujian ke-3, Arsitektur *Serverless* mendapatkan nilai 32 pada keseluruhan performa yang masuk ke dalam kategori lambat. Pada aspek *First Contentful Paint* (FCP) menunjukkan 0,6 *seconds*, untuk aspek *Time to Interactive* atau *First Input Delay* (FID) menunjukkan 19,5 *seconds*, untuk aspek *Speed Index* menunjukkan 5,9 *seconds*, untuk aspek *Total Blocking Time* (TBT) menunjukkan 14.120 *miliseconds*, untuk aspek *Large Contentful Paint* (LCP) menunjukkan 6,6 *seconds*, sedangkan untuk *Cumulative Layout Shift* (CLS) menunjukkan 0.003 %.

Untuk pengkategorian dari hasil pengujian ke-3 dapat dilihat pada Tabel XII berikut:

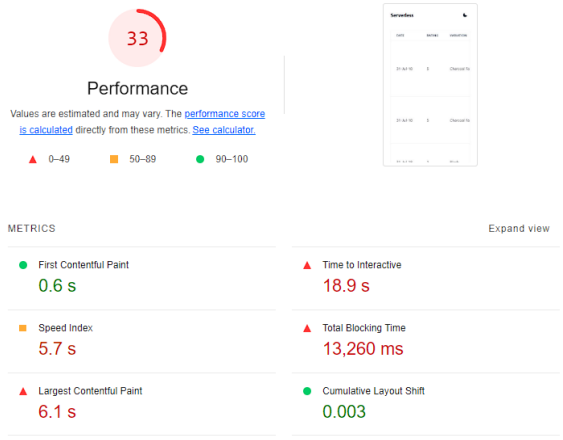
TABEL XII
PENGKATEGORIAN HASIL PENGUJIAN KE-3 *SERVERLESS*

Aspek	Nilai	Kategori
FCP	0,6	Cepat (Hijau)
FID	19,5	Lambat (Merah)
Speed Index	5,9	Lambat (Merah)
TBT	14.120	Lambat (Merah)
LCP	6,6	Lambat (Merah)
CLS	0,003	Cepat (Hijau)
Final Score	32	Lambat (Merah)

Berikut adalah hasil pengujian ke-4 dari performa arsitektur *Serverless*:



Gbr. 19 Hasil Pengujian ke-4 Arsitektur *Serverless*.



Gbr. 20 Hasil Pengujian ke-5 Arsitektur *Serverless*.

Pada Gambar 19 menunjukkan hasil pengujian ke-4, Arsitektur *Serverless* mendapatkan nilai 31 pada keseluruhan performa yang masuk ke dalam kategori lambat. Pada aspek *First Contentful Paint* (FCP) menunjukkan 0,6 *seconds*, untuk aspek *Time to Interactive* atau *First Input Delay* (FID) menunjukkan 20,0 *seconds*, untuk aspek *Speed Index* menunjukkan 6,6 *seconds*, untuk aspek *Total Blocking Time* (TBT) menunjukkan 14.460 *miliseconds*, untuk aspek *Large Contentful Paint* (LCP) menunjukkan 6,6 *seconds*, sedangkan untuk *Cumulative Layout Shift* (CLS) menunjukkan 0.003 %.

Untuk pengkategorian dari hasil pengujian ke-4 dapat dilihat pada Tabel XIII berikut:

TABEL XIII
PENGKATEGORIAN HASIL PENGUJIAN KE-4 *SERVERLESS*

Aspek	Nilai	Kategori
FCP	0,6	Cepat (Hijau)
FID	20,0	Lambat (Merah)
Speed Index	6,6	Lambat (Merah)
TBT	14.460	Lambat (Merah)
LCP	6,6	Lambat (Merah)
CLS	0,003	Cepat (Hijau)
Final Score	31	Lambat (Merah)

Berikut adalah hasil pengujian ke-5 dari performa arsitektur Serverless:

Pada Gambar 20 menunjukkan hasil pengujian ke-5, Arsitektur *Serverless* mendapatkan nilai 33 pada keseluruhan performa yang masuk ke dalam kategori lambat. Pada aspek *First Contentful Paint* (FCP) menunjukkan 0,6 *seconds*, untuk aspek *Time to Interactive* atau *First Input Delay* (FID) menunjukkan 18,9 *seconds*, untuk aspek *Speed Index* menunjukkan 5,7 *seconds*, untuk aspek *Total Blocking Time* (TBT) menunjukkan 13.260 *miliseconds*, untuk aspek *Large Contentful Paint* (LCP) menunjukkan 6,1 *seconds*, sedangkan untuk *Cumulative Layout Shift* (CLS) menunjukkan 0.003 %.

Untuk pengkategorian dari hasil pengujian ke-5 dapat dilihat pada Tabel XIV berikut:

TABEL XIV
PENGKATEGORIAN HASIL PENGUJIAN KE-5 *SERVERLESS*

Aspek	Nilai	Kategori
FCP	0,6	Cepat (Hijau)
FID	18,9	Lambat (Merah)
Speed Index	5,7	Lambat (Merah)
TBT	13.260	Lambat (Merah)
LCP	6,1	Lambat (Merah)
CLS	0,003	Cepat (Hijau)
Final Score	33	Lambat (Merah)

Setelah mendapatkan semua hasil dari proses pengujian, selanjutnya akan dilakukan rata-rata setiap parameter dan hasil akhir dari ke-5 proses pengujian sebelumnya. Berikut adalah hasil rata-rata pada arsitektur *Microservice*:

TABEL XV
HASIL RATA-RATA PADA ARSITEKTUR *SERVERLESS*

Aspek	Uji ke-1	Uji ke-2	Uji ke- 3	Uji ke-4	Uji ke-5	Rata-Rata
FCP	0,6	0,6	0,6	0,6	0,6	0,6
FID	22,7	19,8	19,5	20,0	18,9	20,18
Speed Index	6,7	6,0	5,9	6,6	5,7	6,18
TBT	16.990	14.330	14.120	14.460	13,260	15.232
LCP	6,4	6,8	6,6	6,6	6,1	6,5
CLS	0	0,003	0.003	0,003	0,003	0,0024
Final Score	32	31	32	31	33	31,8

Dari Tabel XV, menunjukkan arsitektur Serverless pada aspek First Contentful Paint (FCP) rata-rata memerlukan 0,6 seconds yang masuk ke dalam kategori cepat, untuk aspek First Input Delay (FID) rata-rata memerlukan 20,18 seconds yang masuk ke dalam kategori lambat, pada aspek Speed Index rata-rata memerlukan 6,18 seconds yang masuk ke dalam kategori lambat, pada aspek Total Blocking Time (TBT) rata-rata memerlukan 15.232 miliseconds yang masuk kedalam kategori lambat, pada aspek Large Contentful Paint (LCP) rata-rata memerlukan 6,5 seconds yang masuk ke dalam kategori lambat, untuk aspek Cumulative Layout Shift (CLS) rata-rata menunjukkan 0,0024 % yang masuk ke dalam kategori cepat, sehingga keseluruhan final score menunjukkan 32.8 yang masuk ke dalam kategori lambat.

D. Pembahasan

Berdasarkan hasil penelitian yang dilakukan, dapat disimpulkan pada Tabel XVI berikut:

TABEL XVI
PERBANDINGAN RATA-RATA HASIL ANALISIS

Aspek	*Microservice*	*Serverless*
FCP	0,6	0,6
FID	19,54	20,18
Speed Index	21,22	6,18
TBT	8.528	15.232
LCP	4,92	6,5
CLS	0,12	0,24
Final Score	32,8	31,8

Hasil penelitian menunjukkan pada aspek First Contentful Paint (FCP) dimana konten pertama kali dimuat pada arsitektur Microservice memerlukan rata-rata waktu 0,6 seconds, begitu juga dengan arsitektur Serverless memerlukan waktu yang sama yaitu 0,6 seconds.

Pada aspek First Input Delay (FID) dimana pengguna dapat melakukan interaksi pertama kali pada aplikasi, arsitektur Microservice lebih unggul karena memerlukan rata-rata waktu 19,54 seconds dibandingkan dengan arsitektur Serverless yang memerlukan waktu 20,18 seconds.

Pada aspek Speed Index dimana konten visual pertama kali dimuat saat dalam proses memuat atau loading suatu proses pengolahan data di sebuah halaman, arsitektur Serverless lebih unggul karena memerlukan rata-rata waktu 6,18 seconds dibandingkan dengan arsitektur Microservice yang memerlukan waktu 21,22 seconds.

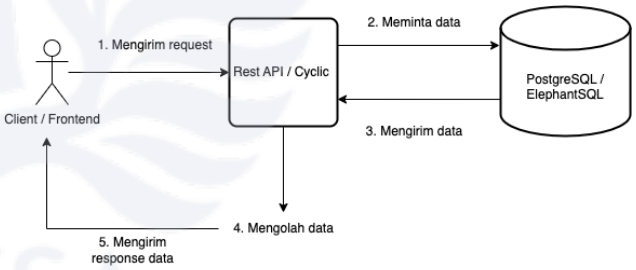
Pada aspek Total Blocking Time (TBT) dimana waktu blocking antara aspek FCP dan FID yang melebihi 50 miliseconds, arsitektur Microservice lebih unggul karena memiliki rata-rata selisih 8.528 miliseconds dibandingkan dengan arsitektur Serverless yang memerlukan waktu 15.232 miliseconds.

Pada aspek Largest Contentful Paint (LCP) dimana waktu yang diperlukan untuk memuat konten terbesar pada suatu halaman, arsitektur Microservice lebih unggul karena memerlukan rata-rata waktu 4,92 seconds dibandingkan dengan arsitektur Serverless yang memerlukan waktu 6,5 seconds.

Pada aspek Cumulatife Layout Shift (CLS) yang merupakan jarak perpindahan letak posisi teks atau gambar ketika berhasil dimuat, arsitektur Microservice lebih unggul karena memiliki jarak perpindahan dengan rata-rata 0,12 % dibandingkan dengan arsitektur Serverless yang memiliki jarak perpindahan dengan rata-rata 0,24 %.

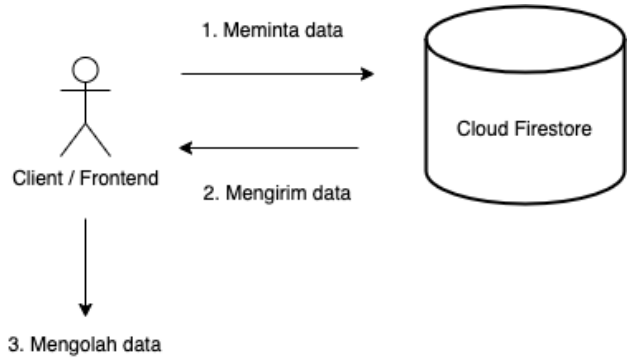
Pada penilaian keseluruhan dari Google Lighthouse dengan pembombotan paramater yang sudah ditentukan, arsitektur Microservice lebih unggul dengan nilai rata-rata 32,8 dibandingkan dengan arsitektur Serverless yang memiliki nilai rata-rata 31.8.

Berikut bagan arsitektur yang nantinya akan berkaitan dengan pembahasan teori hasil penelitian ini:



Gbr. 21 Bagan Arsitektur *Microservice*.

Pada Gambar 21 menunjukkan bahwa pada arsitektur *Microservice*, *Rest API* memiliki proses/*thread* meminta data ke *database*, mengolah data yang diterima dan juga mengirim *response* ke *FrontEnd.*



Gbr. 22 Bagan Arsitektur *Serverless*.

Pada Gambar 22 menunjukkan bahwa arsitektur *Serverless* proses/*thread* pengolahan data ada pada sisi *client/FrontEnd*.

Pada aspek *First Contentful Paint* (FCP) dimana kedua arsitektur bernilai sama, hal ini dikarenakan sesuai dengan penjelasannya yang diambil dari dokumentasi resmi *Web Vitals* [15] dimana konten pertama kali dimuat. Aspek FCP akan muncul sebelum aplikasi sisi *frontend* berjalan pertama kali sebelum dapat terhubung ke koneksi diluar sisi *frontend* seperti *Backend API* pada *Microservice* dan *Cloud Firestore* pada *Serverless*. Karena aplikasi belum terhubung ke masing-masing pengolahan data di setiap arsitektur, maka nilai yang muncul pun akan sama.

Pada aspek *Speed Index* dimana arsitektur *Serverless* yang lebih unggul dibandingkan dengan arsitektur *Microservice*, hal ini dikarenakan pada proses pengolahan data di *Serverless* dilakukan pada sisi *frontend* yang bergantung dengan spesifikasi perangkat pengguna sehingga proses *main thread* dari *Cloud Firestore* lebih cepat sesuai dengan yang dijelaskan dari penelitian terdahulu yang berjudul "*Serverless Applications: Why, When, and How?"* dilakukan oleh Eisman dkk pada tahun 2021 [14]. Berbeda dengan arsitektur *Microservice* yang bergantung dengan spesifikasi perangkat dan juga koneksi internet yang berada pada server API. Sehingga sesuai dengan penjelasan Speed Index dari dokumentasi resmi *Google Lighthouse* [13], aspek tersebut bergantung pada kecepatan main thread suatu program sehingga arsitektur *Serverless* lebih unggul dibandingkan dengan arsitektur *Microservice*.

Meskipun arsitektur *Serverless* memiliki proses *main thread* lebih cepat dari sisi *Cloud Firestore*, namun karena terdapat beban tambahan pada sisi *FrontEnd* untuk mengola data sebelum masuk ke proses *Rendering* sehingga menyebabkan *Blocking* sesuai dengan aspek TBT yang dimaksud dari dokumentasi resmi *Web Vitals* [15]. Karena hal yang sama pun menyebabkan proses LCP dimana konten terbesar pun semakin lama dimuat begitu juga dapat menyebabkan CLS dimana perubahan konten jadi lebih besar pada sisi Serverless sesuai dengan penjelasan aspek yang di jelaskan pada dokumentasi resmi *Web Vitals* [15].

IV. PENUTUP

A. Kesimpulan

Penelitian ini dilakukan dengan tujuan agar dapat mengetahui perbandingan performa *rendering* antara arsitektur *Microservice* dan arsitektur *Serverless*. Parameter yang dihitung diambil dari model pengujian Web Vitals dengan bantuan *Google Lighthouse* sehingga dapat diambil kesimpulan sebagai berikut :

1) Proses penganalisaan dimulai dengan membuat 2 aplikasi sama dengan nama *Amazon Alexa Reviews* yang melakukan rendering dengan jumlah data yang banyak, kedua aplikasi dibuat dengan arsitektur *Microservice* dan *Serverless*. Selanjutnya akan dilakukan pengujian performa menggunakan tools dari Google yaitu *Google Lighthouse* dengan 6 aspek yaitu *First Contentful Paint* (FCP), *First Input Delay* (FID), *Speed index*, *Total Blocking Time* (TBT), *Largest Contentful Paint* (LCP), dan juga *Cumulative Layout Shift* (CLS).
2) Hasil penelitian ini menunjukkan bahwa Arsitektur *Microservice* lebih unggul karena memiliki nilai yang lebih tinggi dalam 4 aspek yaitu *First Input Delay* (FID), *Total Blocking Time* (TBT), *Largest Contentful Paint* (LCP), dan juga *Cumulative Layout Shift* (CLS) dibandingkan dengan arsitektur Serverless yang lebih unggul hanya dalam 1 aspek yaitu *Speed Index*. Akan tetapi pada aspek *First Contentful Paint* (FCP) pada kedua arsitektur memiliki nilai yang sama. Selain aspek tersebut, penilaian keseluruhan dari *Google Lighthouse* juga menunjukkan arsitektur *Microservice* memiliki nilai lebih banyak dibandingkan dengan arsitektur *Serverless*.

B. Saran

Walaupun dalam sisi performa arsitektur *Microservice* lebih unggul dibandingkan dengan arsitektur *Serverless*, namun bukan berarti arsitektur *Serverless* tidak dapat dipertimbangkan ketika ingin mengembangkan aplikasi untuk kedepannya, karena sesuai dengan penelitian terdahulu dari Eismann dkk pada tahun 2021 yang berjudul "*Serverless Applications: Why, When, and How*?" [14]. Penelitian tersebut menjelaskan kelebihan lainnya dari arsitektur *Serverless*, seperti cepatnya waktu pengembangan dan efisiensi biaya yang dibutuhkan.

Saran bagi pengembangan penelitian berikutnya yaitu diharapkan dapat melakukan analisis perbandingan dari sisi lain seperti efisiensi biaya, efisiensi waktu dalam pengembangan dan semacamnya. Karena dengan adanya penelitian dari sisi lain dapat membantu pengembang aplikasi ketika ingin menentukan suatu arsitektur yang cocok untuk studi kasus aplikasi yang ingin dibuat.

REFERENSI

[1] Padillah, R. (2021). Implementasi Revolusi Industri (4.0) Pada Ukm Ayam Broiler Melalui Mesin Pakan Ayam Otomatis Berbasis Internet Of Things (IoT). JATI EMAS (Jurnal Aplikasi Teknik Dan Pengabdian Masyarakat), 5(1), 1–4. https://doi.org/10.36339/JE.V5I1.382.

[2] Gunawan, R., Aulia, S., Supeno, H., Wijanarko, A., Uwiringiyimana, J. P., Mahayana, D., & Teknik, S. (2021). Adiksi Media Sosial dan Gadget bagi Pengguna Internet di Indonesia. TECHNO-SOCIO EKONOMIKA, 14(1), 1–14. https://doi.org/10.32897/TECHNO.2021.14.1.544.

[3] Pradiani, T. (2018). PENGARUH SISTEM PEMASARAN DIGITAL MARKETING TERHADAP PENINGKATAN VOLUME PENJUALAN HASIL INDUSTRI RUMAHAN. Jurnal Ilmiah Bisnis Dan Ekonomi Asia, 11(2), 46–53. https://doi.org/10.32812/JIBEKA.V11I2.45.

[4] Septian, C. (2020). Pemanfaatan Mikrokontroler Untuk Monitoring Penggunaan Air Dan Pembayaran Air Bulanan Anggota Ksm Suka Jernih Berbasis Website. http://elibrary.unikom.ac.id.

[5] Arwin Dermawan, D., Chamdan Mashuri, M., Ginanjar Setyo Permadi, Mk., & Duta Alif Gunawan Dini Widiasih, Mk. S. (2022). Membuat Game Berbasis Website Menggunakan Bahasa Javascript dan PHP. www.rcipress.rcipublisher.org.

[6] Pramustia, R. F. D., & Suyatno, D. F. (2021). Analisis Pengaruh Kualitas Website Terhadap Kepuasan dan Loyalitas Pelanggan Sociolla Menggunakan WebQual 4.0 (Studi Kasus: Pelanggan Sociolla di Jawa Timur). Journal of Emerging Information System and Business Intelligence (JEISBI), 2(3), 94–100. https://ejournal.unesa.ac.id/index.php/JEISBI/article/view/41857.

[7] Shahir Daya, Nguyen Van Duy, Kameswara Eati, Carlos M Ferreira, Dejan Glozic, Vasfi Gucer, Manav Gupta, Sunil Joshi, Valerie Lampkin, Marcelo Martins, Shishir Narain, & Ramratan Vennam. (2016). Microservices from Theory to Practice: Creating Applications in IBM Bluemix Using the Microservices Approach. IBM Redbooks. https://shorturl.at/EHJQ9.

[8] Mufrizal, R., & Indarti, D. (2019). Refactoring Arsitektur Microservice Pada Aplikasi Absensi PT. Graha Usaha Teknik. Jurnal Nasional Teknologi Dan Sistem Informasi, 5(1), 57–68. https://doi.org/10.25077/teknosi.v5i1.2019.57-68.

[9] Suryotrisongko, H. (2017). Arsitektur Microservice untuk Resiliensi Sistem Informasi. Sisfo, 06(02), 231–246. https://doi.org/10.24089/J.SISFO.2017.01.006.

[10] Oktaviani, D., Papilaya, F. S., & Tanaem, P. F. (2021). Perancangan Aplikasi E-Menu Restaurant dengan Menggunakan Cloud Computing dan Serverless Architecture Lambda. Explore: Jurnal Sistem Informasi Dan Telematika (Telekomunikasi, Multimedia Dan Informatika), 12(1), 1–9. https://doi.org/10.36448/JSIT.V12I1.1887.

[11] Krishnan Andi, H., & Krishnan Andi, H. (2021). Analysis of Serverless Computing Techniques in Cloud Software Framework. Article in Journal of ISMAC, 03(03), 221–234. https://doi.org/10.36548/jismac.2021.3.004Li, B., Peng, X., Xiang, Q., Wang, H., Xie, T., Sun, J., Liu, X., & Bertolino, A. (n.d.). Enjoy your observability: an industrial survey of microservice tracing and analysis. https://doi.org/10.1007/s10664-021-10063-9.

[12] Nichifor, E., Lixandroiu, R. C., Chițu, I. B., Bratucu, G., & Trifan, A. (2021). How does mobile page speed shape in-between touchpoints in the customer journey? A research regarding the most trusted retailers in Romania. Journal of Theoretical and Applied Electronic Commerce Research, 16(5), 1369–1389. https://doi.org/10.3390/jtaer16050077

[13] Google. (2022). Lighthouse - Chrome Developers. https://developer.chrome.com/docs/lighthouse, tanggal akses: 24 Februari 2023.

[14] Eismann, S., Scheuner, J., van Eyk, E., Schwinger, M., Grohmann, J., Herbst, N., Abad, C. L., & Iosup, A. (2021). Serverless Applications: Why, When, and How? IEEE Software, 38(1), 32–39. https://doi.org/10.1109/MS.2020.3023302.

[15] Walton, P. (2022, April 30). Web Vitals. https://web.dev/vitals, tanggal akses: 24 Februari 2023.