DESAIN DAN ARSITEKTUR SERVERLESS CLOUD COMPUTING PADA APLIKASI PENGHITUNG KALORI MAKANAN BERBASIS MOBILE MENGGUNAKAN LAYANAN GOOGLE CLOUD PLATFORM

Maulana Bintang Irfansyah, Sofyan Noor Arief, Bagas Satya Dian Nugraha
Teknik Informatika, Jurusan Teknologi Informasi, Politeknik Negeri Malang
Jalan Soekarno Hatta No. 9, Lowokwaru, Kota Malang, Jawa Timur
maulana.bintang.irfansyah@gmail.com

ABSTRAK

Aplikasi Penghitung Kalori Makanan berbasis *mobile* merupakan strategi untuk meningkatkan kesadaran akan pentingnya pola makan sehat dalam masyarakat guna mencegah masalah kesehatan seperti kurang gizi atau obesitas. Namun, membangun aplikasi *mobile* yang dapat menangani banyak pengguna secara efisien membutuhkan infrastruktur yang kompleks dan mahal. Penelitian ini bertujuan untuk mengatasi masalah tersebut dengan memanfaatkan teknologi *cloud computing* menggunakan Google Cloud Platform untuk menyediakan infrastruktur *serverless* yang efisien. Pendekatan arsitektur *serverless cloud computing* pada Aplikasi Penghitung Kalori Makanan yang telah dibuat bertujuan untuk meningkatkan skalabilitas, efisiensi, dan performa aplikasi. Penelitian ini menggunakan metode studi literatur terkait informasi *serverless cloud computing*, analisis kebutuhan *cloud service* yang digunakan, perancangan dan implementasi arsitektur *serverless computing*, serta pengujian khususnya *Black Box Testing* dan *Load Testing*. Hasil pengujian fungsional aplikasi telah menunjukkan nilai efektivitas sebesar 100% yang berarti aplikasi sudah sangat sesuai dengan kriteria pengujian yang telah ditetapkan. Selain itu, hasil pengujian non-fungsional aplikasi dapat memproses permintaan dengan baik, terbukti dengan mampu menangani 300 pengguna menggunakan dua *instance* dalam server Google Cloud Platform.

Kata kunci : *Cloud Computing, Serverless Computing, Google Cloud Platform, Load Testing, Black Box Testing*

1. PENDAHULUAN

Saat ini, banyak industri di Indonesia telah memanfaatkan teknologi informasi secara maksimal. Berbagai aspek kehidupan manusia kini telah terbantu atau bahkan digantikan oleh kehadiran teknologi digital. Tujuan dari teknologi informasi ini adalah untuk memperoleh efisiensi optimal terkait pengembangan model bisnis inovatif yang menggunakan teknologi digital. Komputasi awan (*cloud computing*) merupakan salah satu dari sejumlah teknologi yang tengah populer. Teknologi tersebut dapat diartikan sebagai model layanan teknologi informasi yang menawarkan fleksibilitas kepada pengguna melalui penggunaan server virtual, tingkat skalabilitas yang tinggi, serta manajemen layanan yang efesien [1].

Dalam beberapa tahun terakhir, teknologi canggih yang berperan penting dalam mendorong transformasi digital adalah *cloud computing*. Data dari Kaspersky menemukan bahwa adopsi cloud computing di Indonesia terus berkembang pesat. Saat ini, 19,4% perusahaan di Indonesia telah menggunakan *cloud computing*. Diperkirakan, dalam 12 bulan ke depan, jumlah ini akan meningkat menjadi 32,1%. Hasil data tersebut menunjukkan bahwa *cloud computing* telah menjadi solusi krusial bagi perusahaan-perusahaan di Indonesia untuk memperkuat posisi kompetitif mereka dalam konteks digital [2].

Akses ke sumber daya seperti perangkat lunak, server, jaringan, dan penyimpanan dari lokasi mana pun dan kapan saja sebagai layanan dimungkinkan oleh *cloud computing*. Teknologi ini juga dapat diartikan sebagai mekanisme yang memungkinkan pengelolaan kumpulan sumber daya teknologi informasi, termasuk infrastruktur dan aplikasi terhubung secara hampir tak terbatas dan sepenuhnya dimiliki serta dikelola oleh pihak ketiga. Pelanggan dapat mengakses sumber daya tersebut secara dinamis melalui jaringan privat maupun publik [3].

Platform-as-a-Service (PaaS) adalah sebuah paradigma komputasi tanpa server, memungkinkan pengembang untuk membawa kode mereka dan menyebarkannya dengan menyerahkan tanggung jawab pengelolaan dan penskalaan *Virtual Machine* (VM) yang mendasarinya kepada vendor [4].

Dalam PaaS, aplikasi pelanggan berjalan dan dieksekusi di dalam *container*, dengan kontrol yang lebih sedikit terhadap operasi dan infrastuktur secara keseluruhan. Ketika sebuah *container* tidak lagi menerima permintaan proses apa pun untuk waktu yang singkat, *container* tersebut diskalakan menjadi nol, dan pelanggan tidak ditangih untuk itu, bahkan ketika VM yang mendasarinya terus berfungsi di latar belakang dan tersembunyi dari pandangan pelanggan [5].

Dengan menyediakan infrastruktur komputasi tanpa server, layanan infrastruktur yang dioptimalkan, dan adanya model biaya operasional yang lebih terjangkau, komputasi awan atau *cloud computing* telah merevolusi penggunaan layanan organisasi secara fleksibel. Salah satu pendekatan dalam cloud computing adalah arsitektur *serverless*, yang

memungkinkan aplikasi dijalankan tanpa perlu mengelola server.

Arsitektur *serverless* dapat membantu organisasi menghemat biaya dan meningkatkan skalabilitas [6]. Pengembang (*developer*) semakin tertarik pada arsitektur *serverless* dalam proses pengembangan aplikasi yang menggunakan komputasi awan karena keharusan untuk mengurus infrastruktur secara langsung ditiadakan oleh pendekatan ini dan memungkinkan fokus yang utama pada pengembangan kode secara efektif [7].

Google Cloud Platform atau yang sering disebut GCP (akronim) memiliki berbagai produk untuk membantu para *developer* dalam membangun program, mulai dari situs web sederhana hingga aplikasi kompleks serta menyediakan layanan distribusi hingga ke seluruh dunia. Komputasi *serverless* merupakan evolusi dalam cloud computing yang dimanfaatkan dalam membantu pengembangan aplikasi tanpa perlu mengkhawatirkan pengelolaan server, perangkat keras dan perangkat lunak [8].

Komputasi *serverless* telah terbukti bermanfaat dan menarik secara ekonomi untuk meng-hosting beban kerja untuk aplikasi web, back-end REST API, administrasi back-office, transformasi data yang ringan, dan masih banyak lagi. Studi menunjukkan penggunaan yang paling menonjol untuk komputasi *serverless* termasuk hosting aplikasi web dan *back-end* REST API [9].

Berdasarkan penjelasan di atas, penelitian dilakukan untuk mengembangkan *Mobile Cloud Platform* dengan arsitektur *serverless cloud computing*. Dengan kemampuan *auto-scaling*, sumber daya yang diperlukan tidak perlu dikonfigurasi secara manual pada komputer *serverless* sehingga-untuk membangun sebuah *Mobile Cloud Platform* ada beberapa kebutuhan yang harus dipenuhi yaitu komponen komunikasi dan komponen manajemen data. Komponen komunikasi dari perangkat mobile menuju platform menggunakan *Representational State Transfer* (REST), sedangkan komponen manajemen data menangani mekanisme penyimpanan data dan akses data.

Aplikasi Penghitung Kalori Makanan berbasis mobile belum pernah diuji dari segi fungsional maupun non-fungsional. Pengujian diperlukan untuk memastikan bahwa semua fitur dalam aplikasi ini beroperasi sesuai dengan harapan dan untuk mengevaluasi kinerja aplikasi secara keseluruhan. Oleh karena itu, peneliti perlu melakukan pengujian pada aplikasi ini guna mengidentifikasi kegagalan dan masalah yang ada, sehingga dapat dipastikan bahwa aspek fungsional dan non-fungsional aplikasi telah beroperasi sesuai dengan persyaratan yang ditetapkan oleh pengembang.

2. TINJAUAN PUSTAKA

2.1. *Serverless*

Serverless adalah frasa yang mengacu pada layanan yang hanya diakses dan dibayar saat diperlukan. Layanan tersebut telah disediakan oleh *cloud-provider* seperti Amazon Web Service (AWS), Google Cloud Platform (GCP), dan Microsoft Azure yang biasanya menawarkan jaminan ketersediaan, keandalan, dan kinerja. Biaya layanan ini bervariasi, mencakup penggunaan kapasitas penyimpanan, transfer data, daya komputasi, dan lain-lain. Kondisi tersebut mendorong pengembang atau *developer* untuk merancang arsitektur aplikasi yang meminimalkan pembayaran. Oleh karena itu, komputasi tanpa server atau serverless sangat tepat untuk perangkat lunak yang menekankan pembaharuan cepat serta pengembangan-fitur [10].

2.2. *Docker*

Docker adalah proyek yang dibuat tersedia secara bebas yang digunakan oleh banyak pengguna khususnya pengembang atau developer untuk mengembangkan dan mengeksekusi aplikasi dalam suatu wadah (*container*). *Docker* menerapkan arsitektur *client-server*, di mana terdapat sebuah *daemon docker* yang bertanggung jawab atas pembangunan, pelaksanaan, dan penyebaran *container docker*. Sistem yang sama dapat digunakan untuk menjalankan klien *docker* dan *daemon docker*, dan komunikasi dilakukan melalui RESTful API menggunakan antarmuka jaringan. Beberapa komponen utama dari arsitektur yakni "*docker container*, *docker client*, *docker images*, *docker daemon*, *docker registry*, dan *docker client*." Teknologi *client-server* ini menyalurkan koneksi antara*docker-client* dan *docker-daemon* [11].

2.3. *Resource-Oriented-Architecture* (ROA)

Sejumlah keterbatasan terdapat dalam proses pengembangan layanan web RESTful yang ditetapkan oleh *Resource-Oriented Architecture* (ROA), yaitu *addressability*, *statelessness*, *uniform interfaces*, *representations*, dan *connectedness*. Penerapan batasan-batasan ini secara menyeluruh akan menghasilkan sistem dengan keuntungan seperti skalabilitas interaksi komponen, keseragaman antarmuka, kemandirian komponen, penggunaan komponen penengah untuk mengurangi latensi, penekanan pada aspek keamanan, dan enkapsulasi *legacy-system* [12].

2.4. *Google Cloud Platform*

Google menyediakan berbagai kategori layanan, termasuk sumber daya komputasi, sumber daya penyimpanan, basis data, layanan jaringan, manajemen identitas dan keamanan, alat pengembangan, alat manajemen, serta layanan khusus. Pengguna menggunakan Google Cloud Platform (GCP) untuk mengakses bagian dari infrastruktur internal *Google*. Ragam layanan yang lazim tersedia di berbagai penyedia layanan *cloud*, seperti penyediaan mesin virtual sesuai permintaan melalui *Google Compute Engine*, serta fasilitas penyimpanan objek melalui *Google Cloud Storage* .

2.5. *Load Testing*

Teknik uji kinerja yaitu load testing mengukur respons sistem di bawah berbagai kondisi lonjakan pengguna. Pengujian tersebut dapat membantu dalam menentukan perilaku perangkat lunak saat diakses oleh banyak pengguna bersamaan. Pengujian perfoma atau load testing membutuhkan simulasi akses bersamaan ke aplikasi web dan metode ini dianggap lebih efisien dibandingkan dengan mengundang sejumlah besar orang mengakses situs peramban pada saat bersamaan [14].

2.6. *Black Box-Testing*

Pengujian *black-box* dilakukan berdasarkan kebutuhan dan spesifikasi perangkat lunak tanpa memerlukan pemahaman tentang sistem internal, struktur, atau kode program. Kesalahan fungsional dalam sistem, termasuk pada antarmuka, struktur data, inisialisasi, terminasi, dan kinerja, dideteksi melalui metode ini [15].

3. METODE PENELITIAN

Gambar di bawah (Gambar 1) menggambarkan dengan terperinci uraian langkah-langkah metodologi penelitian yang diterapkan dalam penelitian ini.

Gambar 1. Diagram Alir dari Metode Penelitian

3.1. Studi Literatur

Pengumpulan pustaka dan informasi yang terkait pada topik penelitian dilakukan di tahap ini. Studi literatur dimulai dengan mengumpulkan referensi dan landasan teori yang mendukung pelaksanaan penelitian. Referensi dan landasan teori ini diperoleh dari buku, jurnal, penelitian sebelumnya, serta dokumentasi yang tersedia dari *Google*, khususnya terkait pengaplikasian *Google Cloud Platform*.

3.2. Analisis Kebutuhan

Tahap ini bertujuan sebagai mengidentifikasi ketentuan yang diperlukan untuk Mobile Cloud Platform berbasis serverless computing. Pembagian kebutuhan ini terdiri dari dua kategori kebutuhan, yakni Kebutuhan Non-Fungsional Layanan dan Kebutuhan Non-Fungsional Infrastruktur yang dijelaskan dalam rincian berikut:

Tabel 1. Kebutuhan Layanan Non-Fungsional

"Kebutuhan Non-Fungsional"	**Penjelasan**
Adanya layanan (*service*) berbasis *Mobile Service*.	Layanan (*service*) yang digunakan pada infrastruktur *server* adalah perangkat lunak yang berorientasi *Mobile Service*.
Mobile service bersifat *statefull*.	*Mobile Service* yang digunakan pada infrastrtuktur *server* bersifat *statefull*.
Mobile service di-*build* ke dalam *image docker* sehingga tidak bergantung pada bahasa pemrograman.	Penggunaan m*obile service* di-*build* menggunakan *docker*, lalu dijadika *image docker* agar m*obile service* dapat diterapkan dengan bahasa pemrograman apa pun.
Mobile service memanfaatkan teknologi *container docker* untuk menjadi *multi-platform*.	Menerapkan *container docker* agar *mobile service* yang dibangun mampu diterapkan di *server* berbagai *platform*.
Mobile Service menggunakan *port* yang dapat diakses secara *public*.	Salah satu *port* berjalan pada *mobile service* sehingga dapat dipakai menjadi *public access*.
Mobile service menggunakan *database* SQL	Basis data (*database*) berjenis SQL diperlukan *mobile service* sebagai penyimpanan data.
Mobile Service memerlukan *storage assets*.	Jika *mobile service* membutuhkan penyimpanan untuk menyimpan *assets* seperti foto, video, file, dll., maka harus ada server penyimpanan yang tidak bergantung pada server.

Tabel 2. Kebutuhan Non-Fungsional Infrastruktur

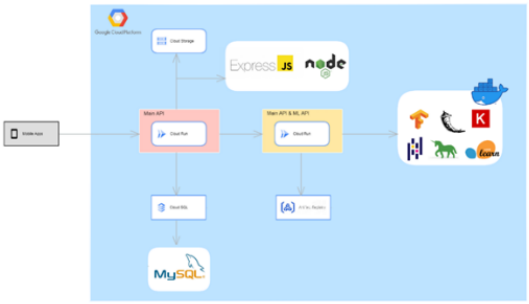
"Kebutuhan Non-Fungsional"	**Penjelasan**
Kemampuan *auto increment capacity* yang tersedia dalam *database*.	Kapasitas *disk* dapat ditingkatkan secara otomatis oleh *server* agar data dan *assets* dapat disimpan tanpa khawatir kehabisan kapasitas *disk*.
Cloud provider menyediakan infrastruktur server di beberapa zona	*Cloud provider* menyediakan infrastruktur *server* di beberapa zona agar dapat mengawasi ketersediaan ketika terjadi kegagalan di wilayah spesifik.

"Kebutuhan Non-Fungsional"	Penjelasan
Dapat mengatur infrastruktur *server* untuk melakukan *scaling-up* dan *scaling-down*.	Dapat dengan mudah mengontrol kapasitas infrastruktur *server* ketika ada permintaan yang tinggi atau rendah yang disebabkan oleh peningkatan atau penurunan jumlah pengguna.

3.3. Perancangan dan Implementasi

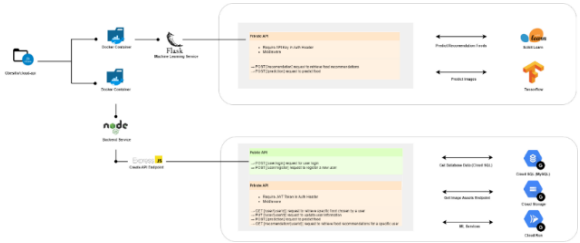
Perancangan *cloud computing* dilakukan di tahap ini agar Aplikasi Penghitung Kalori Makanan berbasis *mobile* dapat menjalankan alur dari kebutuhan yang telah disebutkan, sehingga *Mobile Cloud Platform* dapat digunakan. Perancangan pertama adalah pada *Google Cloud Run* dengan menyiapkan layanan *Google Artifact Registry* untuk mengelola *Docker container image Back-end* dan *Machine Learning service*. Lalu, perancangan kedua menyiapkan layanan *Google Cloud SQL* untuk meyimpan dan mengelola data pengguna. Perancangan terakhir, menyiapkan layanan *Google Cloud Storage* untuk menyimpan data gambar pengguna.

Implementasi adalah penerapan dari desain yang telah disusun pada tahap perancangan. Bagian ini melibatkan penerapan *serverless cloud computing* menggunakan layanan *Google Cloud Platform*. Tahap ini akan dibagi menjadi beberapa bagian, yaitu implementasi *Google Cloud Storage*, implementasi *Google Cloud SQL*, implementasi *Google Artifact Registry*, dan implementasi *Google Cloud Run*.



Gambar 2. Perancangan *Serverless Cloud Computing*

Berikut ini merupakan desain arsitektur API dari perancangan *serverless cloud computing* pada Gambar 3. Desain ini menjelaskan kebutuhan API untuk bagian *back-end* dan *machine learning service* dari Aplikasi Penghitung Kalori Makanan berbasis *mobile* yang diterapkan pada layanan *Google Cloud Platform*.



Gambar 3. Desain Arsitektur API *Cloud Computing*

Layanan dari *Google Cloud Platform* digunakan di penelitian ini sesuai dengan rancangan yang ditampilkan pada Gambar 3, berikut penjelasan setiap layanan yang digunakan tercantum dalam tabel berikut:

Tabel 3. Rincian Layanan *Cloud Computing*

Layanan yang digunakan	Spesifikasi yang dapat memenuhi kebutuhan sistem
Google Cloud SQL (*Relational Database Management System*)	a. Ada beberapa *database engine* dari jenis MySQL, PostgreSQL, dan SQL Server yang tersedia untuk memenuhi kebutuhan basis data. b. Basis data dapat memanfaatkan fungsionalita dari fitur *automatic storage increases*. c. Metode untuk menjaga *availability* dari *server* dapat digunakan oleh *database*. d. Spesifikasi untuk server, seperti CPU dan memori, dapat disesuaikan kapan pun diperlukan.
Google Cloud Storage	a. Konfigurasi dibuat untuk mengakomodasi persyaratan penyimpanan server. b. *Google Cloud Platform* sepenuhnya mengelola *storage*. c. *Availability Service Level Agreement* yang disediakan oleh *Google Cloud Platform* mencapai "99.95%" d. Dapat mengonfigurasi penyimpanan data di beberapa zona atau beberapa wilayah.
Google Cloud Run	a. Mendukung *docker container* agar aplikasi dapat dijalankan dalam *container*. b. Dapat menangani skalabilitas secara otomatis. c. Dapat men-*deploy* aplikasi dengan cepat dan mudah. d. Dapat mendukung berbagai bahasa pemrograman dan *runtime environment* yang berbeda.
Artifact Registry	a. Dapat mengatur konfigurasi untuk memenuhi kebutuhan *Artifact Registry Server*. b. *Google Cloud Platform* sepenuhnya mengelola *registry*. c. *Image* dari *docker container* disimpan di beberapa wilayah (*multiple-region*). d. *Artifact Registry Server* mengontrol *versioning* aplikasi dari *mobile service* yang digunakan.

3.4. Pengujian

Pada tahap ini, peneliti akan melakukan rencana pengujian yang terbagi menjadi dua bagian, yakni Pengujian Non-Fungsional dan Pengujian Fungsional.

a. Pengujian Fungsional Aplikasi
Pengujian ini membuktikan bahwa meskipun terjadi perubahan arsitektur sistem menjadi *serverless cloud computing*, fitur-fitur pada aplikasi penghitung kalori makanan berbasis *mobile* tetap berjalan dengan baik. Pengujian *black-box* digunakan sebagai pengujian fungsional aplikasi.

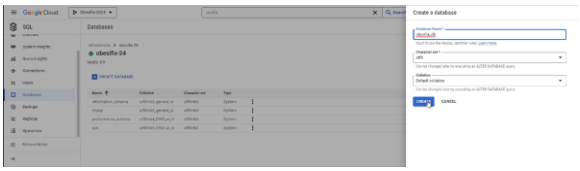
b. Pengujian Non-Fungsional Aplikasi
Pengujian ini akan fokus terhadap mengevaluasi sejauh mana sistem mampu menangani beban kerja yang meningkat seiring dengan jumlah pengguna atau permintaan. Pengujian *load testing* dilakukan dengan menggunakan perangkat lunak *Apache JMeter*. Rencana pengujian akan mencakup skenario-skenario yang berbeda, termasuk pengujian dengan jumlah pengguna yang bertambah.

4. HASIL DAN PEMBAHASAN

Penelitian ini melibatkan implementasi pada layanan Google Cloud Platform dan pengujian baik dari segi fungsional dan non-fungsional terhadap desain arsitektur *serverless cloud computing* yang telah dibuat dengan pembahasan sebagai berikut

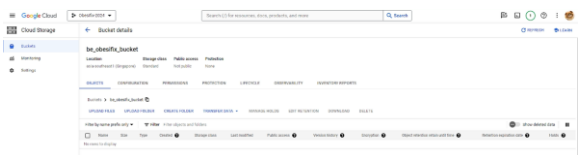
4.1. Hasil Implementasi Google Cloud Platform

a. Cloud SQL:
Cloud SQL dimanfaatkan untuk penyimpanan dan pengelolaan basis data aplikasi secara terpusat. Basis data ini berfungsi untuk menyimpan informasi mengenai data pengguna yang diperlukan oleh Aplikasi Penghitung Kalori Makanan dalam rangka menyediakan rekomendasi dan analisis makanan yang akurat.



Gambar 4. Database yang dibuat pada Cloud SQL

b. Cloud Storage:
Cloud *Storage* digunakan untuk menyimpan data, seperti gambar yang telah di-*input*-kan oleh pengguna pada Aplikasi Penghitung Kalori Makanan.

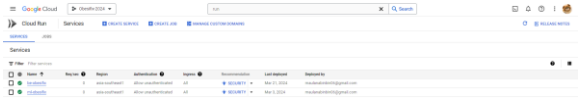


Gambar 5. Bucket yang dibuat pada Cloud Storage

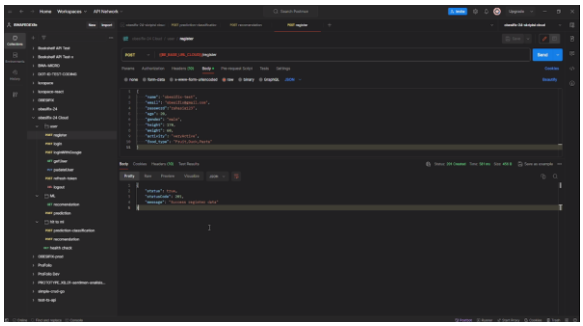
c. Artifact Registry:
Artifact *Registry* dimanfaatkan untuk penyimpanan dan pengelolaan *Docker container image* dari layanan *Back-end* dan *Machine Learning* pada Aplikasi Penghitung Kalori Makanan.

Gambar 6. Docker Images pada Artifact Registry

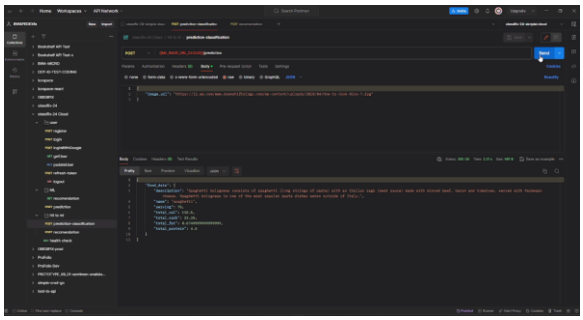
d. Cloud Run:
Cloud Run digunakan untuk menjalankan aplikasi sebagai layanan tanpa server (*serverless*), yang memungkinkan proses *deployment* dan penskalaan otomatis berdasarkan permintaan pengguna. Hal ini berkontribusi dalam mengoptimalkan biaya dan memastikan ketersediaan aplikasi dengan kinerja yang optimal.



Gambar 7. Services yang Diunggah pada Cloud Run



Gambar 8. Hasil API Back-end pada Postman



Gambar 9. Hasil API Machine Learning pada Postman

4.2. *Pengujian Fungsional - Black Box Testing*

Metode *black box* menggunakan metode *equivalence partitions* ini membantu untuk pembuatan *test case* menemukan kesalahan pada aplikasi yang tidak terdeteksi pada saat *handling error*. Pengujian ini difokuskan pada enam fitur, yaitu: Fitur *Register*, Fitur *Login*, Fitur *Calculate food*, Fitur *Recommendation food*, Fitur *Scan Food*, dan Fitur *Edit Profile*. Hasil pengujian ini pada Tabel 4 menunjukkan bahwa kualitas aplikasi penghitung kalori makanan berbasis *mobile* sudah sangat layak digunakan sebagai aplikasi kesehatan.

Tabel 4. Hasil Pengujian *Black Box Testing*

Nama Pengujian	Tujuan Pengujian	Skenario Pengujian	Hasil Pengujian
Fitur Login	Memastikan pengguna masuk ke dalam aplikasi dengan baik.	Masukkan *username* dan *password*.	Aplikasi akan mengarahkan ke halaman utama aplikasi.
Fitur Register	Memastikan pengguna mendaftar ke dalam aplikasi dengan baik.	Masukkan nama, *email*, *password*, umur, berat badan, tinggi badan, pilih jenis kelamin, *daily activities*, *food & beverage preference*.	Data registrasi berhasil tersimpan, sehingga akan muncul pesan "Success register data".
Fitur Calculate Food	Memastikan fitur *calculate food* dapat dijalankan dengan baik.	Pilih aksi untuk tombol 'Add Food' agar dapat menambahkan foto makanan dari perangkat.	Aplikasi menampilkan kalkulasi nutrisi makanan dari hasil *scan food* yang diambil dari kamera atau galeri.
Fitur Recommendation Food	Memastikan fitur *recommendation food* dapat dijalankan dengan baik.	Klik 'view' all' pada bagian "Food Recommendation" di halaman utama untuk menampilkan beberapa rekomendasi makanan bagi *user*.	Aplikasi menampilkan daftar rekomendasi makanan yang ditujukan kepada *client* (*users*).
Fitur Scan Food	Memastikan fitur *scan food* dapat dijalankan dengan baik.	Pilih menu 'Scan', lalu klik aksi tombol 'Take Camera' atau 'Gallery' agar dapat menambahkan gambar makanan dari perangkat, kemudian klik tombol 'Scan Food'.	Aplikasi menampilkan gambar yang telah diunggah dari perangkat.
Fitur Edit Profile	Memastikan fitur *edit profile* dapat dijalankan dengan baik.	Pilih menu 'Profile', lalu klik bagian 'My Profile' serta pilih kolom informasi yang ingin di-*update*.	Aplikasi menampilkan perubahan data *profile client* (*user*).

Pengujian *black box testing* dengan penerapan *equivalence partitions* dilakukan terhadap enam fitur yang ada di aplikasi penghitung kalori makanan berbasis *mobile*. Pengujian ini memerlukan evaluasi langsung terhadap aplikasi untuk menilai fungsionalitas sistem. Pengujian fungsional memberikan pemahaman menyeluruh terhadap aplikasi dan memungkinkan analisis mendalam terhadap isinya, sehingga dapat menyusun rekomendasi yang tepat. Hasil pengujian *black box testing* pada aplikasi mengacu pada Tabel 4.

Tabel 5. Kriteria Persentase Pengujian Fungsional

Nilai Persentase	Keterangan Persentase
0% – 25%	Pengujian Tidak Sesuai
25% – 50%	Pengujian Kurang Sesuai
50% – 75%	Pengujian Cukup Sesuai
75% – 100%	Pengujian Sangat Sesuai

Tabel 5 menampilkan kriteria persentase dari pengujian *test scenario*. ersentase hasil pengujian diperoleh dengan menjumlahkan jumlah pengujian yang berhasil dengan total jumlah pengujian yang dilakukan. Pengujian dianggap berhasil jika persentase hasil pengujian *test scenario* melebihi 75%. Perhitungan efektivitas digunakan untuk mengevaluasi kesesuaian hasil pengujian skenario dengan harapan yang diinginkan. Rincian perhitungan efektivitas dapat dilihat di rumus bawah ini.

$$\%Efektivitas = \frac{Jumlah\ Test\ Scenario\ Berhasil}{Jumlah\ Seluruh\ Test\ Scenario}\ x\ 100\% \quad (1)$$

$$\%Efektivitas = \frac{6}{6}\ x\ 100\% \quad (2)$$

$$\%Efektivitas = 100\ \text{(Sangat Sesuai)} \quad (3)$$

Efektivitas sebesar 100% diperoleh berdasarkan hasil perhitungan efektivitas persentase dari pengujian *black-box*, diperoleh efektivitas sebesar 100%. Hal ini menunjukkan bahwa seluruh pengujian fitur aplikasi telah berhasil dilakukan secara efektif dan sesuai dengan skenario uji yang telah ditetapkan.

4.3. Pengujian Non Fungsional – *Load Testing*

Tabel di bawah ini merupakan hasil dari pengujian non-fungsional atau *performance testing*. Pada Tabel 6 menunjukkan hasil dari pengujian *performance testing* untuk beberapa *endpoint* dan jumlah *client* (*users*) yang berbeda dengan menggunakan Apache JMeter. Sementara itu, Tabel 7 menunjukkan hasil total untuk ketiga indikator pada masing-masing *endpoint*, yaitu: *Response time*, *Throughput*, dan *Error rate*.

Berdasarkan hasil pengujian *load testing* yang ditunjukkan dalam Tabel 6 dan Tabel 7, performa setiap titik akhir (endpoint) API pada kelima skenario pengujian menunjukkan keterkaitan yang signifikan dengan beban permintaan dari klien (pengguna). Rata-rata waktu tanggapan dalam interval waktu (*miliseconds*) secara langsung berhubungan dengan sejumlah permintaan; semakin meningkat jumlah permintaan pengguna, semakin tinggi pula rata-rata waktu tanggapan, seperti yang diindikasikan oleh hasil pengujian pada indikator *response time*. Selanjutnya, indikator *throughput* menunjukkan bahwa semakin besar jumlah permintaan, waktu pemrosesan oleh server dalam satuan detik juga meningkat, mengakibatkan peningkatan beban dan penurunan performa API. Walau begitu, semua permintaan pada setiap titik akhir masih mampu ditangani, sebagaimana ditunjukkan oleh tingkat kesalahan 0,0% untuk setiap

proses. Dengan demikian, performa aplikasi penghitung kalori makanan berbasis *mobile* secara keseluruhan mampu menangani permintaan (*request*) sangat baik.

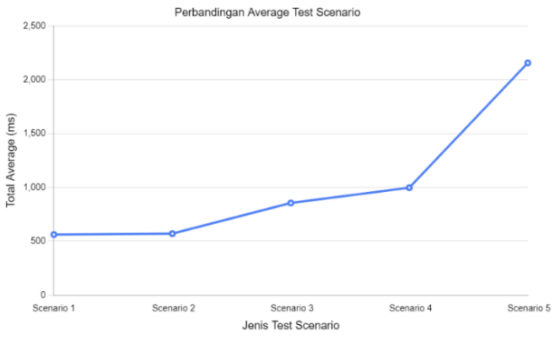
Tabel 6. Hasil Pengujian Performa pada Apache JMeter

Endpoint	***Number of Threads (users)***	***Response time (miliseconds)***	***Throughput (request per seconds)***	***Error Rate***
/register	60	278	1.00	0.0%
/register	120	285	1.00	0.0%
/register	180	299	1.00	0.0%
/register	240	293	1.00	0.0%
/register	300	431	1.00	0.0%
/login/authjwt	60	273	1.00	0.0%
/login/authjwt	120	280	1.00	0.0%
/login/authjwt	180	290	1.00	0.0%
/login/authjwt	240	312	1.00	0.0%
/login/authjwt	300	384	1.00	0.0%
/prediction	60	547	1.00	0.0%
/prediction	120	448	1.00	0.0%
/prediction	180	486	1.00	0.0%
/prediction	240	529	1.00	0.0%
/prediction	300	3650	1.00	0.0%
/recommendation	60	1154	59.5	0.0%
/recommendation	120	1235	59.7	0.0%
/recommendation	180	2349	59.8	0.0%
/recommendation	240	2857	59.9	0.0%
/recommendation	300	4164	60.0	0.0%

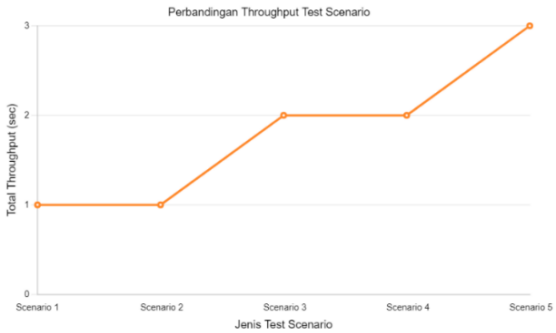
Tabel 7. Hasil Total Ketiga Indikator *Load Testing*

Indikator	**Test Scenarios**				
	60 Users	**120 Users**	**180 Users**	**240 Users**	**300 Users**
Average	563ms	571ms	856ms	998ms	2157ms
Throughput	1.00s	1.00s	2.02s	2.98s	3.19s
Error Rate	0.00%	0.00%	0.00%	0.00%	0.00%

Pada gambar di bawah ini menunjukkan grafik garis (*line graph*) untuk masing-masing ketiga indikator, yaitu: *Response time*, *Throughput*, *Error rate* (Gambar 10, Gambar 11, serta Gambar 12).



Gambar 10. Grafik Perbandingan *Average time*



Gambar 11. Grafik Perbandingan *Throughput*

Gambar 12. Grafik Perbandingan *Error rate*

5. KESIMPULAN DAN SARAN

Aplikasi Penghitung Kalori Makanan berbasis mobile menggunakan desain dan arsitektur serverless *cloud computing* pada layanan *Google Cloud Platform* telah berhasil dilakukan. Hasil penelitian menunjukkan bahwa penggunaan layanan *Google Cloud Platform*, seperti *Google Cloud Run*, *Google Artifact Registry*, *Google Cloud SQL*, dan *Google Cloud Storage*, memungkinkan *developer* untuk men-*deploy* aplikasi dengan mudah serta menyimpan dan menyinkronkan data dengan baik. Pendekatan arsitektur *serverless cloud computing* yang diterapkan pada aplikasi ini berhasil meningkatkan skalabilitas, efisiensi, dan performa aplikasi. Di sisi lain, hasil yang memuaskan juga ditunjukkan oleh pengujian yang telah berhasil dilakukan, baik yang bersifat fungsional maupun non-fungsional. Metode pengujian *black box* digunakan sebagai pengujian fungsional aplikasi yang-menunjukkan bahwa-seluruh fitur dari aplikasi berjalan efektif dan sesuai dengan skenario pengujian yang telah ditetapkan.

Sementara itu, pengujian non fungsional dengan metode *load testing* menunjukkan bahwa meskipun terjadi peningkatan beban request dari pengguna, aplikasi mampu menangani permintaan dengan baik, ditandai dengan error rate yang rendah dan kemampuan untuk tetap memberikan respons dalam waktu yang wajar meskipun dengan beban yang tinggi. Secara keseluruhan, hasil penelitian ini menegaskan bahwa penggunaan desain dan arsitektur serverless *cloud computing* pada Aplikasi Penghitung Kalori Makanan berbasis mobile merupakan tindakan yang tepat dalam meningkatkan efisiensi dan performa aplikasi.

DAFTAR PUSTAKA

[1] N. Ramsari and A. Ginanjar, "Implementasi Infrastruktur server berbasis cloud computing untuk web service berbasis teknologi google cloud platform," in *Conf. Senat. STT Adisutjipto Yogyakarta*, 2022, vol. 7.

[2] S. Setiawan and A. Gui, "FAKTOR-FAKTOR PENENTU YANG MEMPENGARUHI ADOPSI CLOUD COMPUTING DI INDONESIA," *Infotech J. Technol. Inf.*, vol. 9, no. 1, pp. 1–8, 2023.

[3] D. L. F. Anissa and R. Andryani, "Penerapan Cloud Computing Dalam Aplikasi Panggil Teknisi Berbasis Android Menggunakan Google Cloud Platform," *J-SAKTI (Jurnal Sains Komput. dan Inform.*, vol. 6, no. 2, pp. 1292–1300, 2022.

[4] D. Moreau, K. Wiebels, and C. Boettiger, "Containers for computational reproducibility," *Nat. Rev. Methods Prim.*, vol. 3, no. 1, p. 50, 2023.

[5] O. Paakkunainen, "Serverless computing and FaaS platform as a web application backend," 2019.

[6] P. Sbarski, Y. Cui, and A. Nair, *Serverless Architectures on AWS, Second Edition*. Manning, 2022.

[7] A. G. Purwanto, T. Timotius, Y. H. Pesik, and D. Darmanto, "Analisis Dan Desain Sistem Manajemen Proyek Menggunakan Cloud Computing Dengan Arsitektur Serverless," *J. Nas. Komputasi dan Teknol. Inf.*, vol. 6, no. 3, pp. 308–319, 2023.

[8] R. B. A. Pradana and A. Bhawiyuga, "Pengembangan Platform IoT Cloud berbasis Layanan Komputasi Serverless Google Cloud Platform (GCP)," *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 6, no. 4 SE-, pp. 1841–1847, Feb. 2022, [Online]. Available: https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/10933.

[9] E. Jonas *et al.*, "Cloud programming simplified: A berkeley view on serverless computing," *arXiv Prepr. arXiv1902.03383*, 2019.

[10] L. D. Samsumar and S. Hadi, "PENGEMBANGAN JARINGAN KOMPUTER NIRKABEL (WiFi) MENGGUNAKAN MIKROTIK ROUTER (STUDI KASUS PADA SMA PGRI AIKMEL)," *Method. J. Tek. Inform. dan Sist. Inf.*, vol. 4, no. 1 SE-Articles, pp. 1–9, Jun. 2021, doi: 10.46880/mtk.v4i1.59.

[11] S. Dwiyatno, E. Rachmat, A. P. Sari, and O. Gustiawan, "Implementasi virtualisasi server berbasis docker container," *PROSISKO J. Pengemb. Ris. Dan Obs. Sist. Komput.*, vol. 7, no. 2, pp. 165–175, 2020.

[12] E. N. Hamdana and M. E. Apriyani, "Analisis Implementasi Restfull Web Service Menggunakan Resource-Oriented Architecture," *J. Inform. Polinema*, vol. 6, no. 2, pp. 57–60, 2020.

[13] D. Sullivan, *Official Google Cloud Certified Associate Cloud Engineer Study Guide*. John Wiley & Sons, 2019.

[14] D. Permatasari *et al.*, "PENGUKURAN THROUGHPUT LOAD TESTING MENGGUNAKAN TEST CASE SAMPLING GORILLA TESTING," *Semin. Nas. Sist. Inf.*, vol. 3, no. 1 SE-Articles, Sep. 2019, [Online]. Available: https://jurnalfti.unmer.ac.id/index.php/senasif/article/view/268.

[15] H. R. Aktsar, F. Pradana, and F. A. Bachtiar, "Pengembangan Sistem Manajemen Bimbingan Skripsi Mahasiswa Berbasis Web (Studi Kasus : Fakultas Ilmu Komputer Universitas Brawijaya)," *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 5, no. 6 SE-, pp. 2655–2662, Jun. 2021, [Online]. Available: https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/9389.