Implementasi Dan Analisis Cloud Computing Pada Aplikasi Foodit Untuk Menghitung Zat Gizi Makro Pada Makanan

1st Kanz Muhammad Hanif
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
kanzmh@student.telkomuniversity.ac.id

2nd Suryo Adhi Wibowo
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
suryoadhiwibowo@telkomuniversity.ac.id

3rd Koredianto Usman
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
korediantousman@telkomuniversity.ac.id

Abstrak— Gizi makro menjadi faktor penting dalam keseharian seseorang karena merupakan sumber energi utama untuk tubuh manusia. Masih banyak orang yang tidak memiliki pengetahuan dasar mengenai gizi makro, sehingga banyaknya orang yang terkena penyakit akibat kekurangan/kelebihan gizi makro. Aplikasi Foodit menjadi solusi permasalahan tersebut karena memiliki fungsi untuk menghitung jumlah gizi makro harian yang dibutuhkan oleh tubuh, dan mengkalkulasi asupan gizi makro yang dikonsumsi tiap harinya sehingga gizi makro seseorang dapat terkontrol dan tidak menimbulkan penyakit. Pada aplikasi Foodit terdapat layanan *cloud computing* yang digunakan, yaitu Firebase dan Google Cloud Platform. Layanan *cloud computing* memudahkan penulis dalam mengolah *database* dan penggunaan fitur dalam aplikasi. Pengujian yang dilakukan untuk menguji *cloud computing* aplikasi Foodit adalah API *Testing* dan *Load Testing*, dimana API *Testing* berfungsi untuk memvalidasi fungsional API, dan *load testing* dilakukan untuk menguji performansi dari *server* yang digunakan. Hasil pengujian menunjukkan API telah berjalan sesuai perancangan, serta performansi *server* memiliki kekurangan saat menangani pengguna lebih dari 200 dan dengan durasi lebih dari 10 menit.

Kata kunci— cloud computing, firebase, google cloud platform, api *testing*, *load testing*.

I. PENDAHULUAN

Faktor penting dalam keseharian seseorang adalah kesehatan tubuh. Memiliki tubuh yang sehat membuat aktivitas sehari-hari dapat berjalan dengan lancar. Gizi seimbang merupakan langkah utama untuk memiliki tubuh yang sehat. Masih banyak orang yang tidak memiliki pengetahuan dasar mengenai gizi seimbang, sehingga masih banyak orang yang terkena penyakit akibat kekurangan/kelebihan gizi. Gizi dibagi menjadi dua bagian, gizi makro yang merupakan sumber energi utama untuk tubuh, dan gizi mikro yang berfungsi menjaga kesehatan dan fungsi tubuh yang optimal. Aplikasi Foodit merupakan aplikasi yang dapat menghitung jumlah gizi makro yang dibutuhkan oleh tubuh, dan mengkalkulasi asupan gizi makro yang dikonsumsi tiap harinya.

Cloud Computing adalah model komputasi dimana sumber daya komputasi seperti *server*, penyimpanan data, dan perangkat lunak disediakan dan diakses melalui internet. Dengan menggunakan *cloud computing*, penulis dapat dengan mudah mengakses dan mengelola sumber daya sesuai kebutuhan tanpa harus memiliki infrastruktur fisik sendiri. Model ini memberikan fleksibilitas, efisiensi, dan akses keseluruhan kepada penulis.

Pada aplikasi Foodit terdapat dua fitur utama, yaitu perhitungan kebutuhan energi menggunakan regresi, dan pengenalan makanan menggunakan *object recognition*. Kedua fitur tersebut memiliki model yang membutuhkan implementasi *cloud computing* yaitu *deployment* model. Pada aplikasi Foodit *cloud computing* juga berfungsi untuk pengelolaan *database* dan fitur autentikasi. Dalam implementasi *cloud computing* menggunakan dua platform utama, yaitu Firebase dan Google Cloud Platform.

II. DASAR TEORI

Pada aplikasi Foodit, platform utama *cloud computing* yang digunakan terdapat dua platform, antara lain Firebase dan Google Cloud Platform. Fitur yang digunakan pada Firebase adalah Cloud Firestore, Cloud Storage, dan Firebase Authentication. Sedangkan untuk platform GCP fitur yang digunakan adalah fitur VM Instance. Pengujian yang akan dilakukan untuk menguji performansi adalah API *Testing* dan *Load Testing*.

A. Firebase

Firebase merupakan sebuah platform pengembangan aplikasi yang dikembangkan oleh Google yang memiliki fokus utama pada pengembangan aplikasi web dan *mobile*. Firebase menyediakan berbagai fitur seperti *hosting*, *database*, *storage*, autentikasi, *messaging*, dan analitik untuk membantu proses pengembangan aplikasi. Pada implementasi *cloud computing* aplikasi Foodit menggunakan fitur Cloud Firestore dan Cloud Firebase yang berfungsi sebagai *database*, dan Firebase Authentication yang digunakan sebagai fitur autentikasi aplikasi.

1. Cloud Firestore

Cloud Firestore merupakan *database* NoSQL yang fleksibel dan skalabel untuk pengembangan aplikasi [1], dan berfungsi untuk menyimpan dan mengelola data terstruktur dalam bentuk koleksi, dokumen, dan *field*. Pada aplikasi

Foodit fitur Cloud Firestore digunakan untuk menyimpan data dari kandungan gizi makanan, ekstraksi kata kunci yang akan digunakan untuk proses pencarian, data-data penting terkait makanan, serta menyimpan data hasil yang dikirimkan oleh pengguna pada aplikasi yang nantinya akan ditampilkan kembali pada aplikasi.

2. Cloud Storage

Cloud Storage memiliki fungsi yang mirip dengan Cloud Firestore, yaitu berfungsi sebagai *database* namun untuk menyimpan dan mengelola data dalam bentuk file seperti gambar, video, dan dokumen. Cloud Storage memiliki kelebihan utama antara lain operasi yang stabil, keamanan yang kuat, dan skalabilitas yang tinggi [2]. Fitur ini digunakan pada aplikasi untuk menyimpan file gambar dari *database* makanan, hasil pengambilan gambar untuk fitur *object recognition*, dan foto profil pengguna. Gambar-gambar tersebut nantinya akan ditampilkan pada aplikasi.

3. Firebase Authentication

Firebase Authentication merupakan layanan yang menyediakan fitur-fitur autentikasi untuk pengguna aplikasi. Dengan menggunakan Firebase Authentication penulis dapat memproses *login* pengguna menggunakan alamat *email* dan sandi [3]. Pada aplikasi Foodit Firebase Authentication digunakan untuk mengimplementasikan fitur autentikasi seperti registrasi/pendaftaran akun pengguna dan login untuk pengguna memasuki aplikasi.

B. Google Cloud Platform (GCP)

Google Cloud Platform atau GCP adalah platform *cloud computing* yang disediakan oleh Google yang menyediakan berbagai macam fitur seperti komputasi, *virtual machine*, *storage*, *function*, *run*, dan masih banyak layanan yang lainnya. Pada aplikasi Foodit fitur GCP yang digunakan adalah fitur VM Instance. Fitur VM Instance membuat penulis dapat mengoptimalkan performa, keamanan, dan skalabilitas aplikasi Foodit secara efisien.

VM Instance merupakan mesin virtual yang dijalankan dan di-*hosting* pada infrastruktur *cloud* GCP [4]. Dengan menggunakan VM Instance, penulis dapat menjalankan aplikasi pada lingkungan yang terisolasi dan aman. VM Instance memiliki berbagai macam konfigurasi sesuai kebutuhan antara lain ukuran CPU (*Central Processing Unit*), *disk memory*, dan lokasi *server*. Pada aplikasi Foodit terdapat beberapa *file* yang akan disimpan di dalam VM Instance seperti *file* model *object recognition* yaitu *file* best.pt, Flask yang berisi alur prediksi model *object recognition*, dan *file* requirements.txt yang berisi daftar kebutuhan paket python yang diperlukan oleh Flask. VM Instance akan berperan sebagai *server environtment* yang berfungsi untuk menjalankan flask API.

C. API *Testing*

API (*Application Programming Interface) Testing* merupakan proses pengujian yang berfungsi untuk memverifikasi dan memvalidasi fungsionalitas, keandalan, dan kinerja dari API yang digunakan. Pada pengujian akan dikirimkan permintaan API menggunakan berbagai metode HTTP yaitu GET, POST, PUT, dan DELETE sesuai dengan parameter yang digunakan. Tujuan utama dari pengujian ini yaitu memastikan apakah API merespons permintaan dengan benar, menghasilkan *output* sesuai dengan rancangan, serta tidak menghasilkan *error* atau kesalahan yang terjadi.

D. *Load Testing*

Load Testing adalah proses menguji kinerja suatu sistem dengan memberikan beban atau *load* secara konstan dan akan bertambah seiring waktu hingga sistem mencapai puncak batas yang dapat diterima. Tujuan dilakukan *load testing* untuk mengukur performansi dari sistem yang diuji, mengetahui batas kemampuan sistem dalam volume pengguna yang tinggi, dan mengetahui apa saja masalah yang akan terjadi jika sistem diberikan beban yang tinggi. Pada aplikasi Foodit *load testing* dilakukan untuk menguji performansi dari *server* VM Instance GCP. Perangkat lunak yang akan digunakan untuk menjalankan pengujian *load testing* yaitu K6. K6 merupakan perangkat untuk menyimulasikan banyak pengguna virtual yang mengakses aplikasi secara bersamaan. Sehingga penulis dapat membuat skema konfigurasi yang diinginkan untuk mengetahui performansi *server* pada beban yang ditentukan, dan melakukan analisis.

Terdapat dua parameter yang akan digunakan untuk mengontrol beban uji, yaitu jumlah waktu pengujian (*duration*), dan jumlah target pengguna virtual pada pengujian (*target*). Sedangkan untuk parameter hasil akan menggunakan RED Method [4], dimana RED Method merupakan pendekatan yang digunakan untuk menganalisis performansi sistem secara efektif. Pendekatan ini banyak digunakan untuk pengujian *load testing* khususnya saat menguji aplikasi atau sistem yang menggunakan arsitektur yang terdistribusi. Parameter dari RED Method antara lain:

1. Request Rate

Rate mengacu pada tingkat beban yang diberikan pada sistem saat melakukan *load testing*. *Rate* akan diukur sebagai banyaknya permintaan yang diterima dalam satu detik. *Rate* memberikan informasi berapa banyak permintaan atau lalu lintas yang harus ditangani sistem pada titik tertentu dalam waktu. *Monitoring rate* akan membantu memahami bagaimana sistem memberi respon saat diberikan beban yang berbeda.

2. Error rate

Parameter kedua dari RED Method berkaitan dengan kesalahan yang terjadi selama *load testing*. *Error* dapat terjadi ketika sistem mengalami masalah saat menangani permintaan pengguna. Pada *load testing* parameter *error* akan diukur sebagai persentase banyaknya permintaan yang *error* dibagi dengan jumlah permintaan. Dengan parameter ini kita dapat mengetahui kapan sistem akan mengalami *error* pada waktu atau jumlah pengguna tertentu.

3. Duration

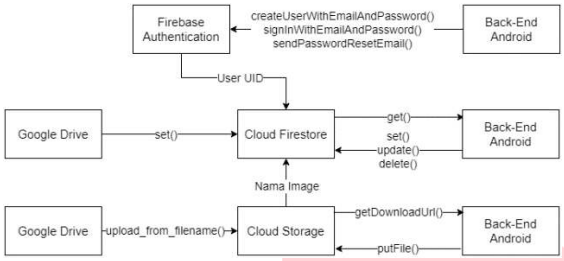
Duration merupakan waktu respons sistem terhadap permintaan yang diberikan. Fokus parameter *duration* adalah distribusi waktu respons yang diukur dari saat permintaan dikirimkan hingga sistem memberikan respons kembali. Hal ini dapat membantu mengetahui apakah terdapat perubahan kinerja seiring perubahan waktu pengujian dilakukan.

III. PERANCANGAN SISTEM

Perancangan sistem *cloud computing* dikatakan berhasil jika penulis dapat merancang dan mengembangkan arsitektur *cloud* dari aplikasi Foodit. Penulis harus mampu menghubungkan seluruh komponen seperti Cloud Firestore,

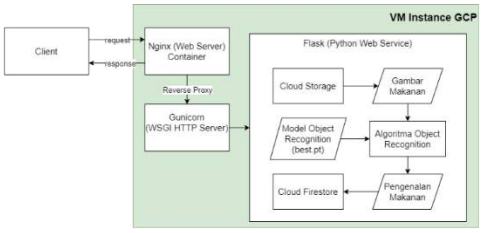
Cloud Storage, Firebase Authentication, Back-End Android, Google Drive, *Client*, dan *Server*. Keberhasilan ditandai dengan integrasi yang berjalan dan berfungsi sesuai dengan rencana antara semua komponen tersebut. Hal tersebut memastikan sistem dapat berjalan efisien dan memberikan nilai tambah yang diinginkan oleh pengguna.

A. Implementasi



GAMBAR 1
Diagram Alir 1 *Back-End Cloud Computing*

Gambar 1 merupakan diagram alir *Back-End Cloud Computing* yang berhubungan dengan *Back-End Android Development* aplikasi Foodit. Pada bagian *Back-End Android Development* dengan Firebase Authentication terdapat tiga fungsi antara lain, createUserWithEmailAndPassword() yang memiliki fungsi membuat akun pengguna aplikasi, signInWithEmailAndPassword() memiliki fungsi untuk memasuki akun pengguna aplikasi, dan sendPasswordResetEmail() yang berfungsi untuk mengirim permintaan atur ulang sandi akun pengguna aplikasi, nantinya ketiga fungsi akan dikirimkan ke Firebase Authentication. Firebase Authentication akan mengirimkan *User* UID atau kode unik identifikasi pengguna kepada Cloud Firestore saat pengguna membuat akun. Pada bagian Back-End Android dengan Cloud Firestore terdapat empat fungsi antara lain, fungsi get() untuk mengambil data dari Cloud Firestore ke *Back-End Android*, fungsi set() untuk melakukan penambahan data, fungsi update() untuk melakukan perubahan data, dan delete() untuk melakukan penghapusan data pada Cloud Firestore dari *Back-End Android*. Pada bagian *Back-End Android* dengan Cloud Storage terdapat dua fungsi yaitu fungsi getDownloadUrl() untuk mengambil *file* gambar pada Cloud Storage ke Back-End Android, dan fungsi putFile() untuk mengirim *file* gambar sebaliknya. Penulis menggunakan Google Drive untuk melakukan automasi penginputan gambar, terdapat dua fungsi yaitu fungsi set() untuk mengirimkan data ke Cloud Firestore, dan fungsi upload_from_filename() untuk mengirimkan *file* gambar ke Cloud Storage. Saat pengguna menunggah gambar, Cloud Storage akan mengirim nama gambar ke Cloud Firestore.



GAMBAR 2
Diagram Alir 2 *Back-End Cloud Computing*

Gambar 2 adalah diagram alir *Back-End Cloud Computing* untuk *request* dan *response client* aplikasi Foodit. Diagram ini menjelaskan proses klien mendapatkan respon setelah mengirimkan permintaan. Langkah pertama klien akan mengirimkan *request* ke Nginx sebagai *web server* yang berperan sebagai *reverse proxy server*. Selanjutnya permintaan yang diterima oleh Nginx akan diteruskan ke Gunicorn yang berfungsi sebagai WSGI (*Web Server Gateway Interface*) yang merupakan *server* untuk menjalankan aplikasi Flask. Pada Flask yang berfungsi sebagai *python web service*, akan dilakukan pengiriman gambar makanan untuk fitur *object recognition* dari Cloud Storage, lalu akan dilakukan proses algoritma pengenalan gambar menggunakan model *object recognition* best.pt. Hasil algoritma pengenalan gambar akan disimpan pada Cloud Firestore. Selanjutnya setelah Flask selesai melakukan proses akan menghasilkan *response* yang diteruskan Gunicorn kepada Nginx. Langkah terakhir Nginx akan mengirim respon kembali kepada klien.

B. Langkah-langkah Implementasi

Berikut merupakan langkah-langkah dari implementasi cloud computing:

1. Merancang arsitektur *cloud* yang digunakan pada aplikasi Foodit.
2. Membuat dan konfigurasi Cloud Firestore, Cloud Storage, dan Firebase Authentication, setelahnya ketiga fitur tersebut dihubungkan dengan *Back-End Android Development*.
3. Opsional membuat automasi menggunakan *file* python dengan platform Google Colab untuk menginput data ke Cloud Firestore dan Cloud Storage, dengan menghubungkan kedua fitur tersebut dengan platform Google Drive.
4. Menyiapkan file Flask, requirements.txt, dan model *object recognition* yang digunakan.
5. Membuat dan melakukan konfigurasi VM Instance GCP.
6. Memasukkan file Flask API, requirements.txt, dan model *object recognition* ke dalam VM Instance GCP.
7. Melakukan proses *deployment* Flask API pada VM Instance GCP.
8. Lakukan pengujian dan evaluasi terhadap arsitektur *cloud* yang telah dibuat.

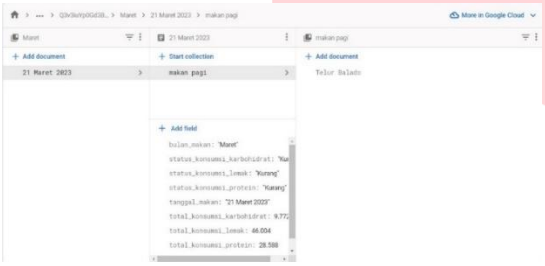
IV. PENGUJIAN DAN ANALISIS

Pengujian dilakukan untuk memastikan apakah implementasi *cloud computing* telah berjalan sesuai dengan rancangan. Metode pengujian pertama yang dilakukan yaitu API *testing* menggunakan aplikasi Postman untuk mengetahui apakah Flask yang dibuat telah berjalan dengan lancar. Selanjutnya akan dilakukan pengujian *Load Testing* menggunakan aplikasi K6 untuk menguji performansi *server*. Setelah dilakukan kedua pengujian, akan dilakukan analisis untuk mengetahui kemampuan sistem yang digunakan.

A. API *Testing*

GAMBAR 3
Api *Testing* Postman

Pada Gambar 3, terlihat tampilan dari aplikasi Postman yang digunakan untuk melakukan pengujian. Akan dilakukan simulasi dengan mengirimkan permintaan POST ke *endpoint* API yang berisi data sesuai dengan parameter API yang sudah dirancang sebelumnya. Sesuai dengan alur yang telah dirancang, jika data yang dikirimkan berhasil mencapai Cloud Firestore, maka pengujian dianggap berhasil dilakukan. Langkah berikutnya untuk mengetahui hasil lebih lanjut, kita dapat memeriksa *database* pada Cloud Firestore.



GAMBAR 4
Api *Testing* Cloud Firestore

Berdasarkan Gambar 4, dapat disimpulkan bahwa pengujian telah berhasil dilakukan dan menghasilkan bukti bahwa Flask berjalan sesuai dengan rancangan. Kesimpulan tersebut didasarkan oleh pengiriman permintaan POST yang dikirimkan, telah berhasil tersimpan pada Cloud Firestore dan berisikan data yang sesuai pada pengiriman permintaan. Ditunjukkan Flask telah berjalan dan berfungsi dengan baik, dan data dapat dikirimkan dan disimpan dengan benar. Sehingga pengujian selanjutnya yaitu *Load Testing* dapat dilaksanakan.

B. Load Testing

Skema konfigurasi *Load Testing* yang dilakukan yaitu durasi pengujian selama 5, 10, 15, dan 20 menit, serta target pengguna virtual sebesar 20, 50, 100, dan 200 target. Pengujian akan dilakukan sebanyak 16 kali sesuai konfigurasi dan menghasilkan data berupa *output* yaitu rata-rata jumlah permintaan per detik (*rate*), persentase kegagalan permintaan (*error*), dan durasi rata-rata permintaan (*duration*).

TABEL 1
Hasil Pengujian *Load Testing*

Output		***Rate* (Req/s)**	***Error rate***	***Duration* (s)**
***Duration* (m)**	**Target pengguna**			
5	20	0,19	0	41,797
	50	0,2	0	81,208
	100	0,20	0	121,912
	200	0,19	0	133,291
10	20	0,12	0	67,654
	50	0,20	0	90,319
	100	0,18	0	162,839
	200	0,26	0,491	223,873
15	20	0,21	0	38,719
	50	0,20	0	93,497
	100	0,20	0	172,590
	200	0,27	0,636	245,795
20	20	0,19	0	41,710
	50	0,21	0	93,333
	100	0,19	0	187,489
	200	0,26	0,719	263,814

Tabel 1 merupakan data hasil *Load Testing* dengan skema konfigurasi yang telah ditentukan. Berdasarkan hasil pengujian *Load Testing* dapat ditarik kesimpulan. Pada parameter pertama yaitu *rate*, mendapatkan hasil yang cukup optimal dengan rentang nilai *rate* sebesar 0,12-0,27 permintaan per detik. Parameter kedua yaitu persentase kegagalan permintaan (*error*), terdapat hasil yang sangat baik pada sebagian besar pengujian dengan hasil rata-rata *error* 0. Namun terdapat *error* yang cukup besar pada saat menguji dengan durasi waktu 10, 15, dan 20 menit, dan jumlah target pengguna sebesar 200 target. Sehingga dapat disimpulkan bahwa sistem kurang optimal saat menangani pengguna dengan jumlah 200 dan durasi lebih dari 10 menit.

V. KESIMPULAN

Layanan *cloud computing* yang digunakan pada aplikasi Foodit antara lain layanan Firebase dengan fitur Cloud Firestore dan Cloud Storage yang berfungsi sebagai *database* aplikasi, dan Firebase Authentication untuk fitur autentikasi aplikasi, serta layanan Google Cloud Platform menggunakan fitur VM Instance yang berfungsi sebagai *virtual server* aplikasi untuk menjalankan Flask yang berisi model *object recognition*. Proses pengujian yang dilakukan yaitu API *Testing* untuk mengetahui apakah API telah berjalan sesuai dengan rancangan dan *Load Testing* untuk mengetahui

performansi dari *server* VM Instance saat diberikan beban tertentu. Hasil analisis dari pengujian yang telah dilakukan adalah API telah berjalan dengan baik ditandai dengan respons yang diberikan telah sesuai dengan rancangan, sehingga selanjutnya dilakukan *Load Testing*. Hasil dari pengujian *Load Testing* yaitu *server* VM Instance kurang optimal saat menangani pengguna dengan jumlah 200 dan durasi waktu lebih dari 10 menit.

REFERENSI

[1] Firebase, "Cloud Firestore," *Firebase Google, Julio*, 2021. https://firebase.google.com/docs/firestore?authuser=0 (accessed Dec. 30, 2022).

[2] A. Setup, W. Setup, F. Setup, and U. Setup, "Cloud Storage for Firebase Key capabilities," 2023. https://firebase.google.com/docs/storage (accessed Dec. 30, 2022).

[3] I. Firebase, "Firebase Authentication | Firebase," *Firebase, Inc.*, 2018. https://firebase.google.com/docs/auth%0Ahttps://firebase.google.com/docs/auth/?gclid=CjwKCAjw8uLcBRACEiwAaL6MSX1xXFqWXYCnrqL-sqPLz5fvkbY1dyAmanpjbmgXz4DjFd_HW-1HuhoCqIAQAvD_BwE%0Ahttps://firebase.google.com/docs/auth/ (accessed Feb. 09, 2023).

[4] T. Wilkie, "The RED Method: How to monitoring your microservices.," 2018. https://grafana.com/files/grafanacon_eu_2018/Tom_Wilkie_GrafanaCon_EU_2018.pdf (accessed Apr. 12, 2023).