

Development of Web Service Based E-Canteen Inventory Application [Perancangan Aplikasi Inventory E-Kantin Berbasis Web Service]

Mukhamat Saifudin ^{*,1)}, Irwan A. Kautsar ²⁾

¹⁾Program Studi Teknik Informatika, Universitas Muhammadiyah Sidoarjo, Indonesia

²⁾Program Studi Teknik Informatika, Universitas Muhammadiyah Sidoarjo, Indonesia

*Email Penulis Korespondensi: irwan@umsida.ac.id

Abstract. *Inventory modules are crucial in an information system, including e-canteens. During storage application design, developers often adopt a monolithic architecture, whereas all application components, backend programs, frontends, and databases are housed within one container. However, monolithic apps face limitations in scaling, causing disruptions and necessitating shutdowns if any component fails. Integration with other apps also poses challenges. This study aims to create an easily scalable and integrable e-canteen inventory module. The development employs the REST API method for web service and Flask as the application framework. The outcome is a web-service-based e-canteen inventory module enabling users to perform RESTful CRUD operations on their canteen products. Load testing method used to determine application's performance when faced with a heavy loads. With this module, system scale up and integration will be easier to implement.*

Keywords – E-Canteen; Flask; Inventory; REST API; Web Service

Abstrak. *Modul inventory sangat penting dalam sebuah sistem informasi, termasuk e-canteen. Dalam merancang aplikasi penyimpanan, tidak sedikit dalam pengembangannya developer masih mengadopsi arsitektur monolith. Semua komponen aplikasi, seperti program backend, frontend, database, disimpan dalam satu wadah. Masalahnya, aplikasi monolith memiliki keterbatasan untuk di scale-up, keseluruhan layanan aplikasi terganggu dan diharuskan untuk berhenti beroperasi apabila salah satu komponen aplikasi mengalami gangguan. Masalah ini juga timbul ketika mengintegrasikan aplikasi inventory monolith dengan aplikasi lain. Penelitian ini bertujuan untuk menciptakan modul inventory e-canteen yang mudah untuk scale-up dan diintegrasikan dengan aplikasi lain. Digunakan metode REST API untuk pengembangan web service dan Flask sebagai framework aplikasinya. Hasil dari penelitian ini adalah modul inventory e-canteen berbasis web-service yang memungkinkan user untuk melakukan CRUD produk kantin mereka secara RESTful. Pengujian menggunakan metode load testing untuk mengetahui performa aplikasi apabila dihadapkan dengan beban yang berat. Dengan adanya modul ini, scale up sistem dan integrasi akan lebih mudah diimplementasikan.*

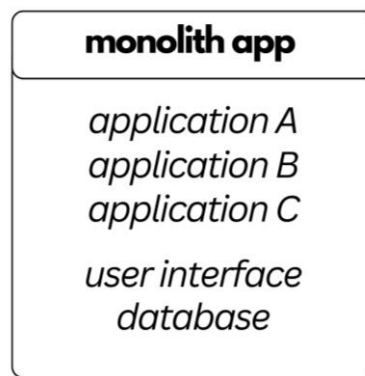
Kata Kunci - E-Kantin; Flask; Inventory; REST API; Web Service

I. PENDAHULUAN

Teknologi website telah banyak digunakan developer untuk beragam bidang yang ada, salah satunya pada sistem inventory atau penyimpanan. Inventori adalah stok yang tersedia pada saat ini juga, suatu list barang yang tersedia dan dapat digunakan pada suatu waktu yang akan datang[1]. Sistem inventory sendiri telah banyak digunakan untuk beragam aplikasi, seperti pada e-commerce, marketplace, aplikasi pergudangan, perpustakaan, dan aplikasi inventory lainnya[2]. Dikembangkannya sistem menjadi terkomputerisasi, memungkinkan pengelolaan inventory makin efektif dan efisien, pelaporan persediaan menjadi makin akurat dan tepat waktu[3].

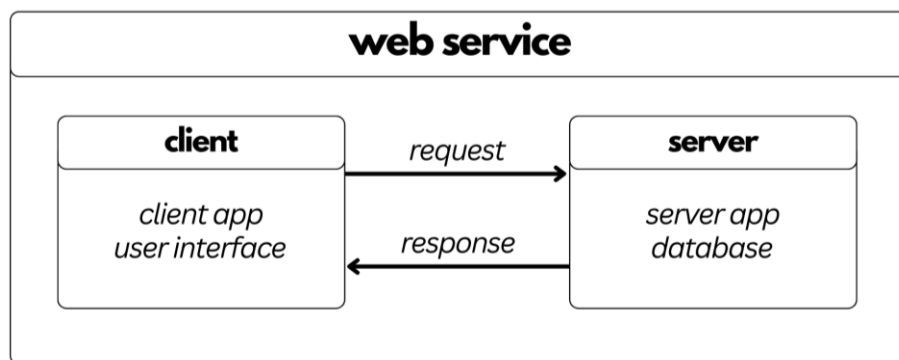
Adaptasi lain dari perkembangan internet adalah trend digitalisasi produk / jasa dengan mengintegrasikannya dengan komputer, perangkat elektronik, ataupun internet, salah satunya adalah e-canteen yang memungkinkan pengguna kantin untuk melihat produk yang tersedia dan melakukan transaksi secara digital menggunakan metode e-payment. E-canteen tidak terlepas pada konsep dan mekanisme stok / inventori produk. Pada penerapannya, sistem management inventory memerlukan akurasi data yang tepat, realtime, dan sinkron, baik untuk aplikasi berskala besar maupun kecil.

Dalam mengembangkan aplikasi atau modul inventory, tidak sedikit developer masih mengadopsi arsitektur monolith karena memang ini adalah bentuk arsitektur yang paling mendasar. Pada arsitektur monolith, seluruh aplikasi berjalan seperti organ-organ yang saling terhubung dalam satu tubuh atau satu wadah. Setiap komponen dari aplikasi ini saling terhubung, apabila salah satu komponen mengalami masalah, maka akan mengganggu komponen lain bahkan fungsi aplikasi bisa terhenti. Aplikasi monolith dibangun dengan basis kode tunggal yang mencakup beberapa layanan yang saling terhubung. Layanan-layanan ini tidak dapat dieksekusi secara independen atau terpisah[4]. Bentuk sederhana arsitektur monolith bisa dilihat pada Gambar 1.



Gambar 1. Arsitektur monolith

Alternatif pengembangan dari arsitektur monolith adalah arsitektur web-service yang memungkinkan dua aplikasi berbeda untuk saling berinteraksi dan bertukar data dalam suatu jaringan yang dapat diakses secara remote, data yang ditukar bisa berupa data berformat JSON atau format lainnya[5]. Pengembangan semacam ini memungkinkan aplikasi berbeda lainnya sekalipun berbeda platform dapat berinteraksi dan mengkonsumsi yang ada pada layanan web service melalui aktivitas request & response antar aplikasi. Seringnya layanan web service direpresentasikan dalam bentuk output format teks, JSON atau XML[6]. Protokol web service yang paling diminati dan paling banyak dipakai adalah REST dan SOAP[7]. Bentuk sederhana dari arsitektur web service bisa dilihat pada Gambar 2.



Gambar 2. Arsitektur web service

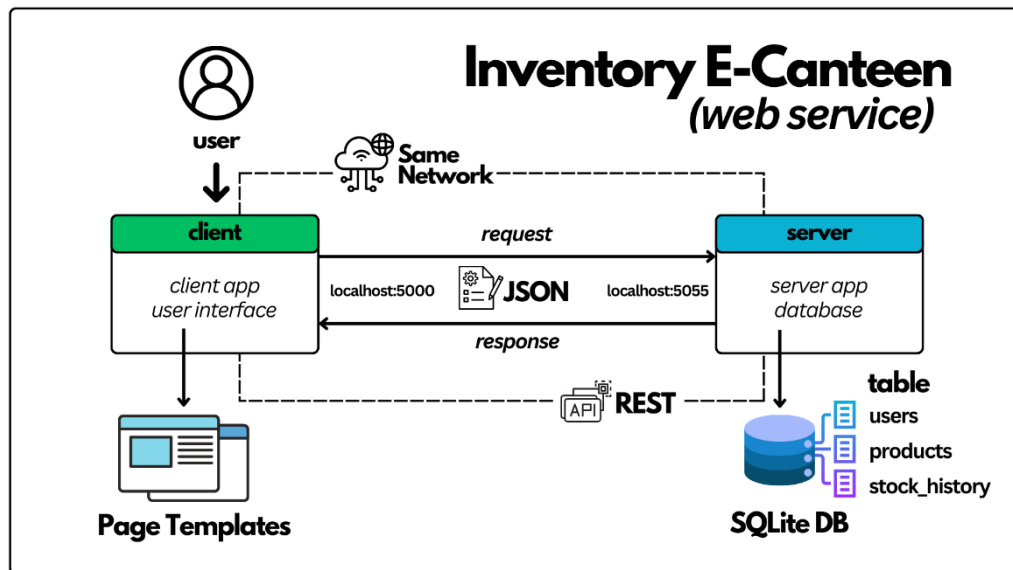
Penelitian terkait inventory dan e-canteen diantaranya adalah [8] tentang modul kantin namun tidak menunjukkan pengujian aplikasi secara spesifik, ada juga [9] tentang sistem informasi inventory sebuah perusahaan namun juga tidak menunjukkan pengujian spesifik. Kemudian ada [10] tentang sistem informasi inventory barang pada apotek dengan pengujian black-box, lalu ada [11] tentang aplikasi inventory peralatan suatu perusahaan dengan metode pengujian User Acceptance Test. Perbedaan penelitian ini dengan empat penelitian tersebut diantaranya adalah perbedaan arsitektur yang digunakan, empat penelitian tersebut mengadopsi arsitektur monolith, selain itu berbeda framework dan database, ketiga penelitian tersebut menggunakan PHP dan database MySQL, serta berbeda metode pengujiannya karena pada tiga penelitian tersebut tidak menggunakan uji performa load testing.

Untuk mengatasi keterbatasan aplikasi monolith seperti pada penelitian terkait, penulis merancang suatu sistem inventory berbasis web service untuk handle stock produk e-canteen secara RESTful dengan tujuan agar aplikasi ini mudah untuk scale-up aplikasi dan integrasi antar sistem makin mudah untuk diimplementasikan. Selain itu, platform website dipilih karena bisa dengan mudah diakses oleh beragam jenis device selama device itu bisa mengakses web browser.

Framework yang digunakan dalam penelitian ini adalah Flask, salah satu framework berbasis Python. Dengan Flask kita dapat membuat situs dengan cepat meskipun library yang digunakan sangat sederhana[12]. Flask juga menyediakan module dan library yang siap digunakan untuk membangun sebuah website[13]. Bahasa pemrograman untuk menggunakan framework Flask adalah Python. Python memiliki sejumlah fitur pendukung, salah satunya adalah pemrograman fungsional dan AOP (Aspect Oriented Programming)[14]. Metode pengujian yang diimplementasikan pada penelitian ini adalah load-testing untuk mengetahui performa aplikasi yang dirancang.

II. METODE

Representational State Transfer (REST) adalah metode yang memungkinkan sistem-sistem berkomunikasi dan berinteraksi melalui permintaan HTTP menggunakan GET, POST, PUT, dan DELETE yang dapat direpresentasikan dalam format JSON atau XML[15]. REST API memiliki endpoints yang merepresentasikan sumber daya atau objek yang ingin diakses atau dimanipulasi. Setiap endpoint memiliki URL yang unik sehingga cocok untuk digunakan pada arsitektur webservice. REST API memiliki sifat stateless, artinya setiap permintaan individu dari klien ke server harus mengandung semua informasi yang diperlukan. Ini mendukung skalabilitas dan pemeliharaan sistem yang lebih baik, karena server tidak perlu menyimpan informasi sesi pada sisi server. Gambar 3 merepresentasikan skema RESTful webservice yang digunakan pada penelitian ini.



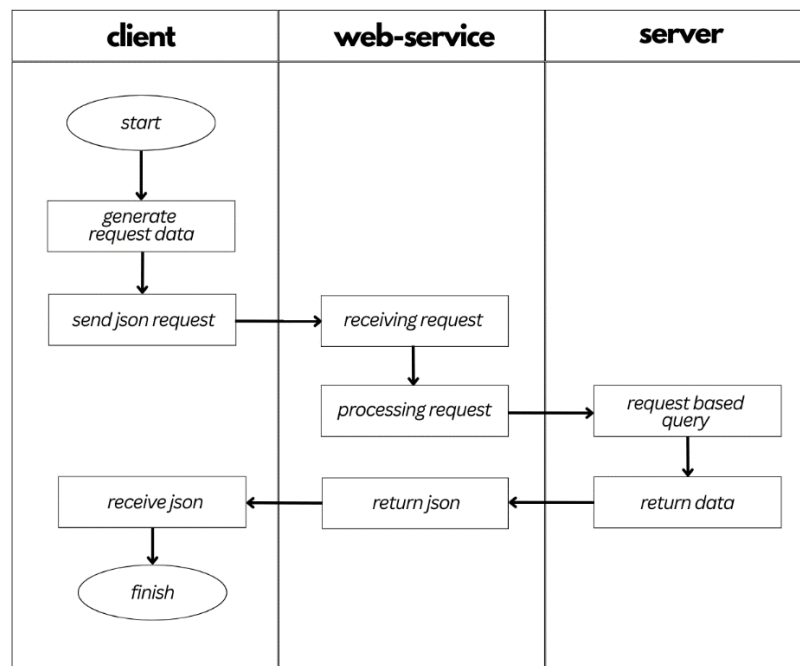
Gambar 3. Skema Modul Inventory E-Canteen

Arsitektur webservice membagi aplikasi menjadi 2 bagian terpisah yang masing-masing bisa berjalan secara independen, artinya apabila sisi client bermasalah ini tidak akan mengganggu sisi server. Satu sisi untuk client yang handle interface, dan sisi server yang berisi aplikasi inti dan database tanpa ada file untuk user interface. Server di sini diposisikan untuk menyimpan data dan program aplikasi yang krusial dan menyimpan database yang termasuk sensitive.

Sisi client dan server berkomunikasi dengan mengirim request ataupun response dalam format JSON. Proses request dan response JSON bisa dilihat pada flowchart Gambar 4. Penulis memilih format JSON dalam penelitian ini karena format JSON lebih simple dan cocok untuk REST API. Format ini didasarkan pada JavaScript yang disebut JavaScript Object Notation atau disingkat JSON[16].

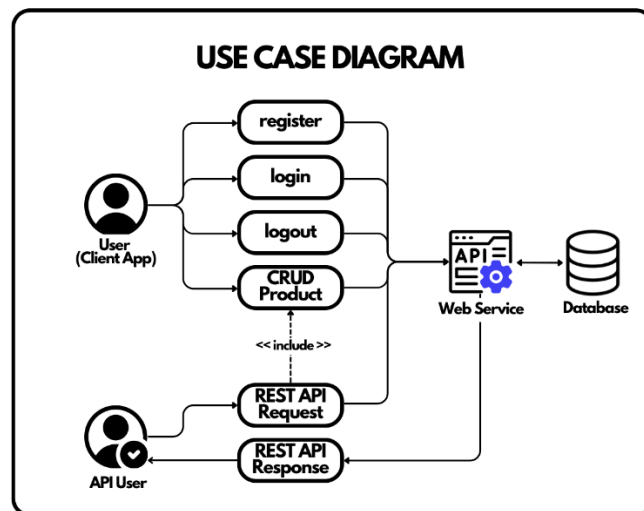
Framework yang digunakan dalam skema modul inventory e-canteen ini adalah Python Flask. Flask memerlukan dua direktori spesifik, yaitu direktori “static” untuk menyimpan semua file static seperti CSS, JavaScript serta file gambar, dan direktori “template” yang berisi semua file HTML ataupun file template jinja[17]. Framework ini bisa dikategorikan ke dalam micro-framework karena tidak memerlukan tools atau library tertentu dalam penggunaannya[18].

Dalam penelitian ini, modul inventory e-canteen dijalankan pada server local, tiap device yang terhubung dalam satu jaringan network yang sama akan dapat menggunakan layanan baik dari sisi client maupun server. Karena berbasis web, bisa diakses beragam device / platform selama bisa mengakses web browser. Pengguna hanya perlu memasukkan URL beserta portnya. Selain itu, karena berbentuk REST API, implementasi lainnya adalah web service bisa diakses oleh developer lain sesuai dengan authentication dan rules yang digunakan.



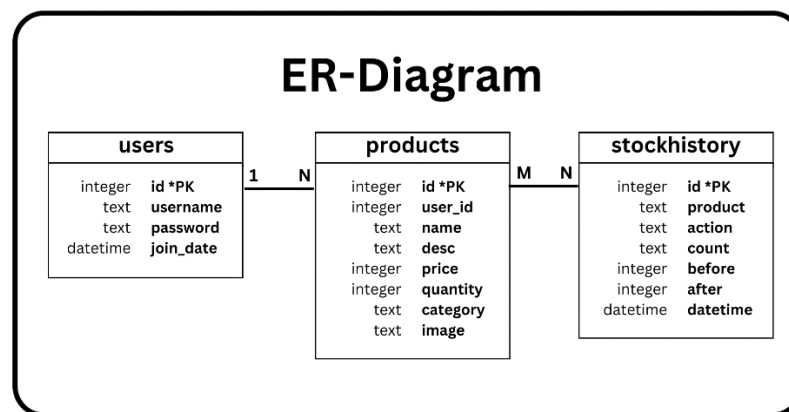
Gambar 4. Flowchart Request & Response

Sebenarnya untuk fitur fungsional, CRUD product sudah cukup untuk mencover kebutuhan akan layanan modul inventory e-canteen. Namun, disini ditambahkan fitur register dan login pada aplikasi client yang dibuat dalam penelitian ini untuk authentication dan representasi scalabilitas untuk multi user. Interaksi antara user dengan sistem direpresentasikan oleh use case diagram pada Gambar 5.



Gambar 5. Use Case Diagram

Untuk sisi database, penulis menggunakan database SQLite. Database ini simple dan portable karena berbasis file dan mudah digenerate menggunakan Python, bisa dikatakan sebagai database paling banyak digunakan secara global mengingat basis data ini digunakan pada mobile device untuk android dan IOS[19]. Relasi antar tabel pada database bisa dilihat pada Gambar 6.



Gambar 6. ER-Diagram

Untuk pengujian sistem, digunakan metode load-testing. Load testing adalah proses menguji kinerja suatu aplikasi atau sistem dengan memberikan beban yang tinggi pada infra-struktur dan mengamati bagaimana sistem tersebut merespons di bawah beban tersebut. Pengujian beban dilakukan pada sebuah sistem (baik prototipe atau sistem yang berfungsi penuh) daripada desain atau model arsitektur[20]. Penulis menggunakan Locust, salah satu library python yang mempermudah developer untuk melakukan uji load-test pada suatu sistem dengan melakukan simulasi swarm spawning.

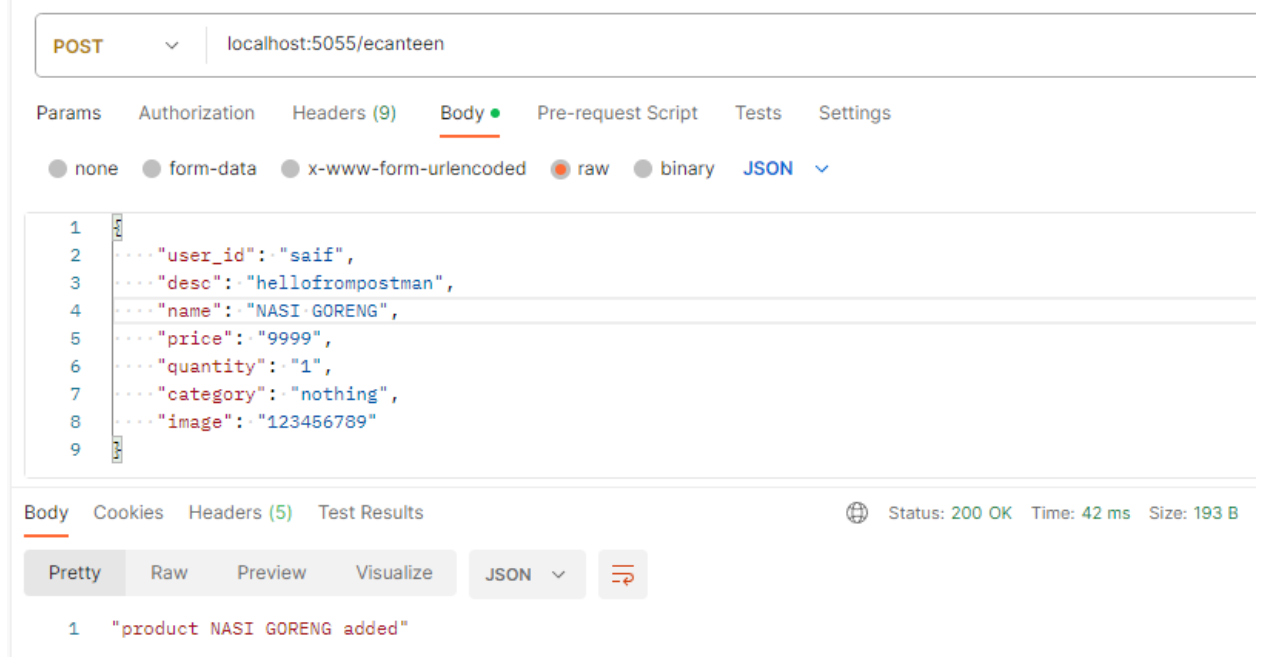
III. HASIL DAN PEMBAHASAN

A. Implementasi

Modul aplikasi inventory e-canteen berhasil dibuat dan semua fitur fungsional CRUD bisa diakses secara RESTful menggunakan http request dengan metode POST, GET, PUT dan DELETE. Pada pengujian ini digunakan aplikasi Postman untuk mengirim request dan menerima response dari web service dengan format JSON.

1. POST

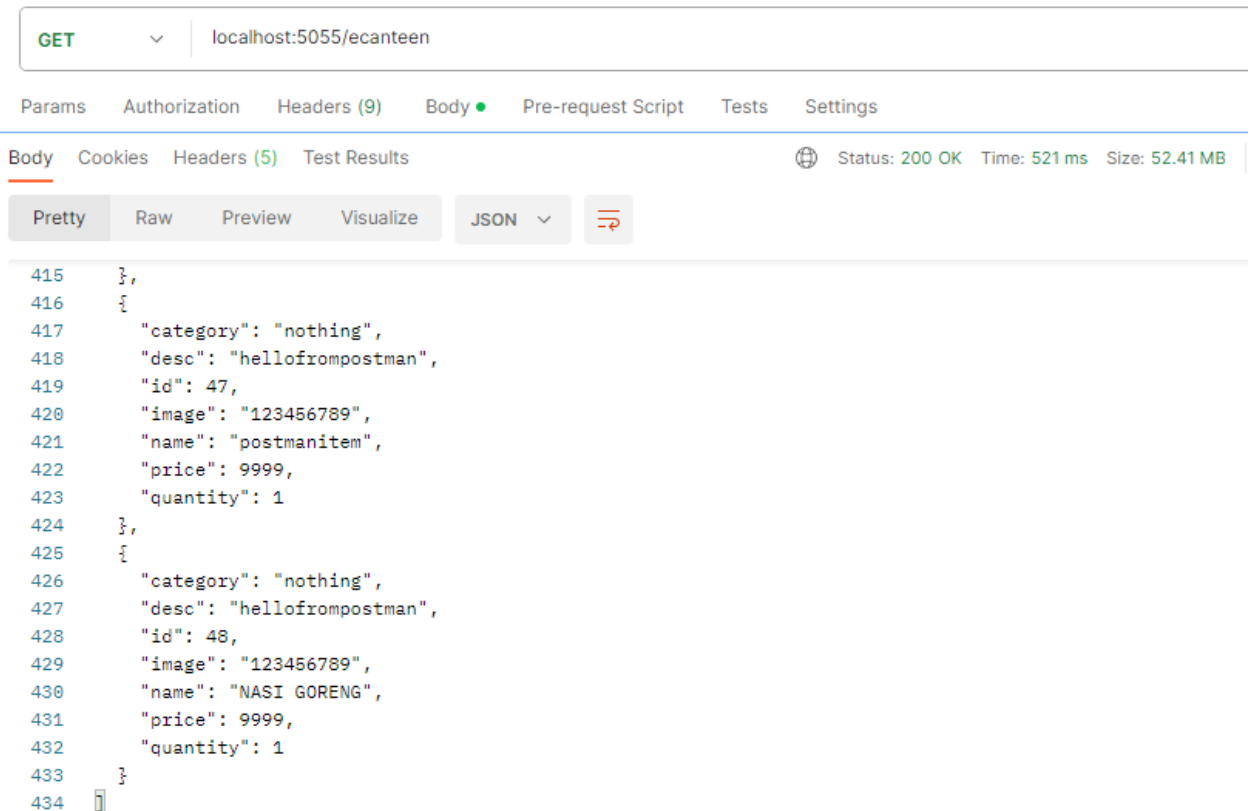
Metode POST digunakan untuk mengirim data baru ke resource. POST merepresentasikan fungsi Create.



Gambar 7. Implementasi POST

2. GET

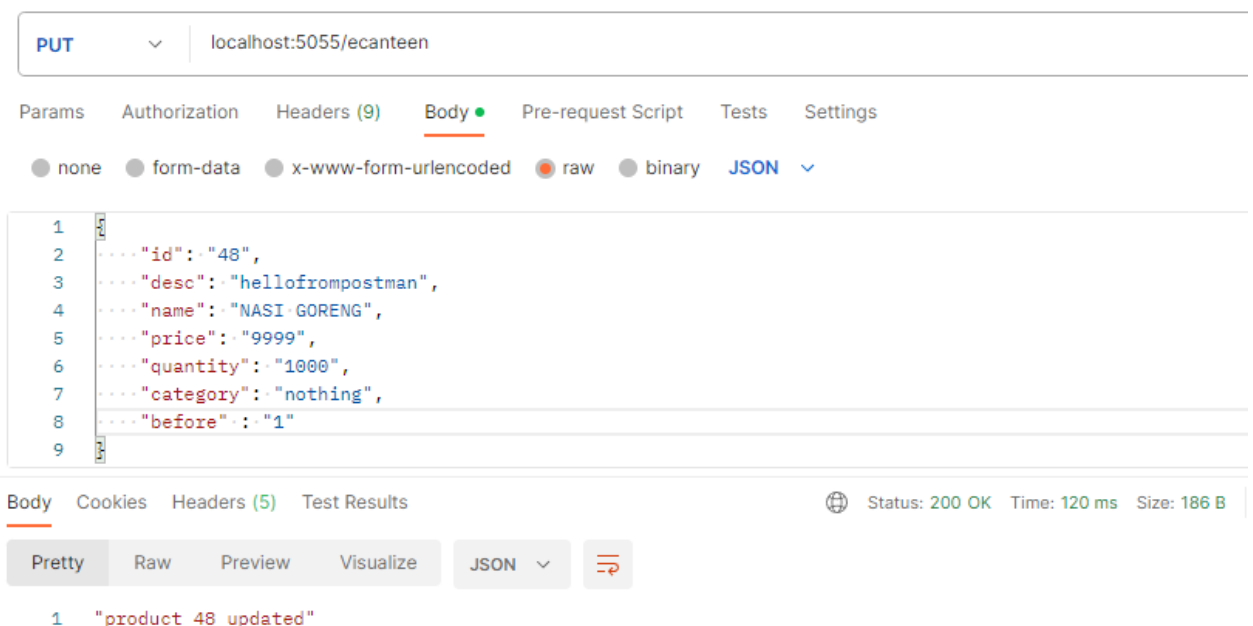
Metode ini digunakan untuk mengambil data dari suatu resource. GET merepresentasikan fungsi Read.



Gambar 8. Implementasi GET

3. PUT

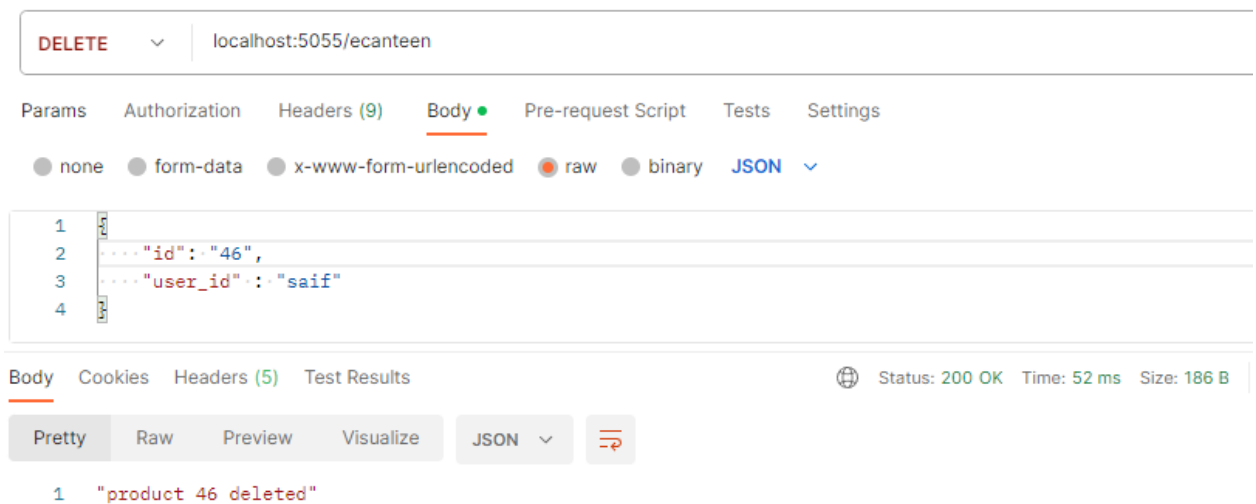
Metode PUT digunakan untuk mengupdate data suatu resource. PUT merepresentasikan fungsi Update.



Gambar 9. Implementasi PUT

4. DELETE

Metode ini digunakan untuk menghapus resource.



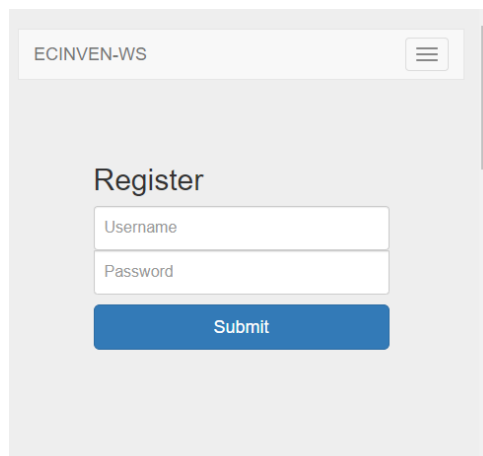
Gambar 10. Implementasi DELETE

B. Tampilan

Berikut adalah tampilan halaman aplikasi berbasis web-service yang telah dirancang. Dibuat untuk mendemokan hasil rancangan dan fungsionalitas aplikasi dari sisi client. Demo aplikasi ketika diimplementasikan pada user non-developer, dimana user mengakses web browser untuk mengakses web service.

1. Register

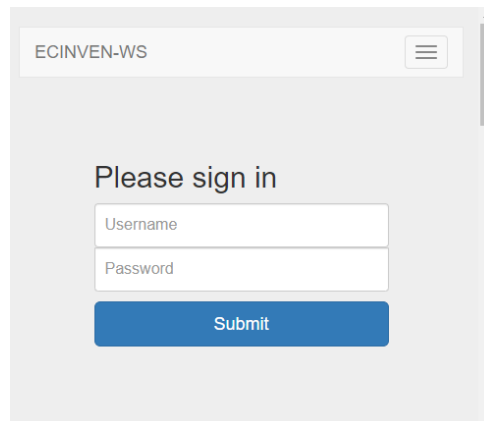
Contoh tampilan halaman untuk mendaftar sebagai mitra (pengguna web service inventory e-canteen).



Gambar 11. Register

2. Login

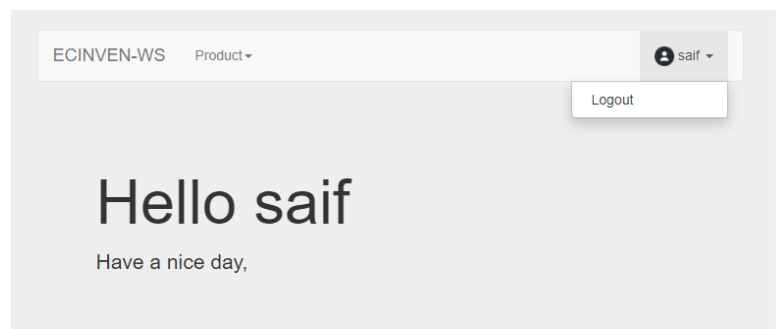
Halaman login untuk user yang sudah terdaftar.



Gambar 12. Login

3. Logout

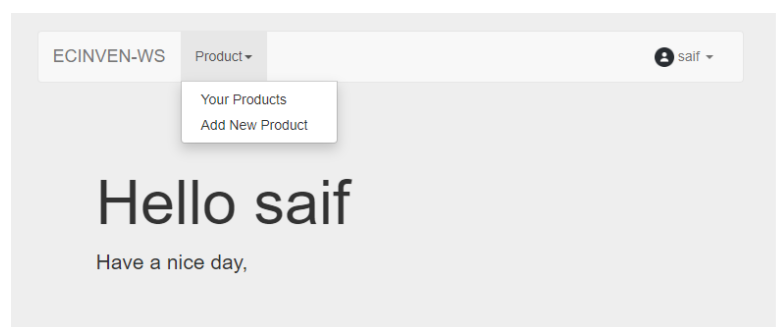
Tombol logout akan muncul pada navbar atas dashboard dengan melakukan klik pada profil user.



Gambar 13. Logout

4. Dashboard

Setelah login, user akan masuk ke dashboard dimana mereka bisa melihat list produk pada Your Products dan menambahkan produk pada halaman Add New Product.



Gambar 14. Dashboard

5. Add Product

Apabila user melakukan klik Add New Products pada dashboard, user akan diarahkan ke halaman berikut. User perlu melakukan input data berupa nama produk, deskripsi produk, harga produk, kuantitas, kategori produk, dan gambar produk. Apabila user melakukan submit dan semua inputan valid maka data akan tersimpan dan produk akan bisa dilihat pada halaman Your Products.

ECINVEN-WS Product

Adding New Product

Strawberry Juice

Its a Juice

1000

999

Category ☐ Makanan ☒ Minuman

Choose File Strawberry-Juice.jpg

Submit

Gambar 15. Menambahkan Produk

6. List Product

Apabila user melakukan klik Your Products pada dashboard, user akan diarahkan ke halaman list produk. Akan terlihat semua produk yang telah user tambahkan beserta datanya, disini user juga bisa mengakses beberapa fitur seperti edit produk, hapus produk dan modifikasi jumlah stok.

ECINVEN-WS Product

Your Product

ID	Name	Price	Category	Image	Quantity
31	Strawberry Juice	1000	minuman		999 Modify Stock? + - Edit Delete
32	Pizza	4444	makanan		9999 Modify Stock? + - Edit Delete
33	Rawon	9999	makanan		9999 Modify Stock? + - Edit

Gambar 16. List Produk

7. Edit Product

Dengan melakukan klik tombol Edit pada halaman list produk. User bisa melakukan edit informasi produk seperti nama produk, harga dan lain sebagainya.

Gambar 17. Edit Produk

8. Delete Product

Untuk melakukan hapus produk, user hanya perlu klik Delete pada halaman list produk.

Gambar 18. Delete Produk

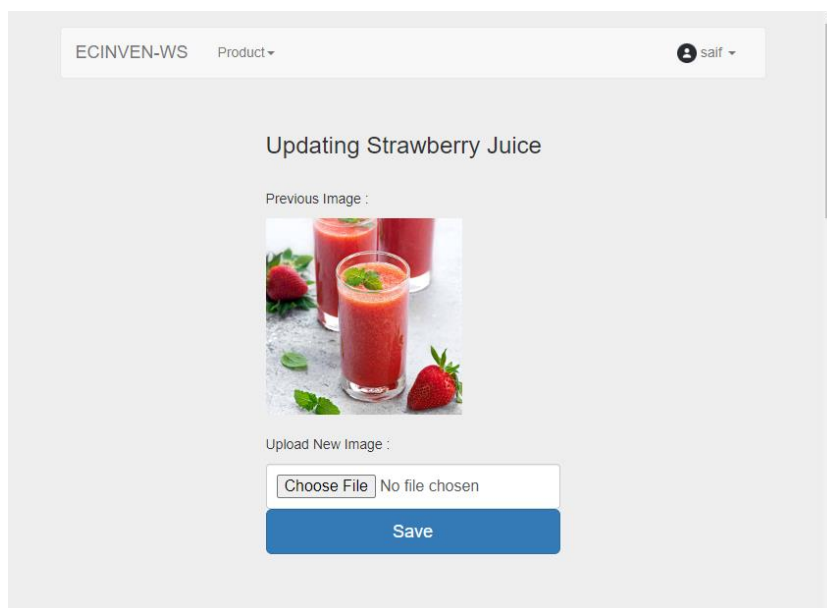
9. Modify Stock

Mengubah stok bisa langsung dilakukan pada halaman list produk agar lebih simpel. User hanya perlu memasukkan jumlah pada bar input, tombol plus biru untuk menambah stok sesuai inputan dan tombol minus merah untuk mengurangi stok sesuai dengan jumlah inputan.

Gambar 19. Modify Stock

10. Change Image

Untuk merubah gambar produk, user hanya perlu melakukan klik pada gambar di dashboard list produk, kemudian melakukan upload gambar baru pada halaman berikut.



Gambar 20. Change Image

C. Fitur Fungsional

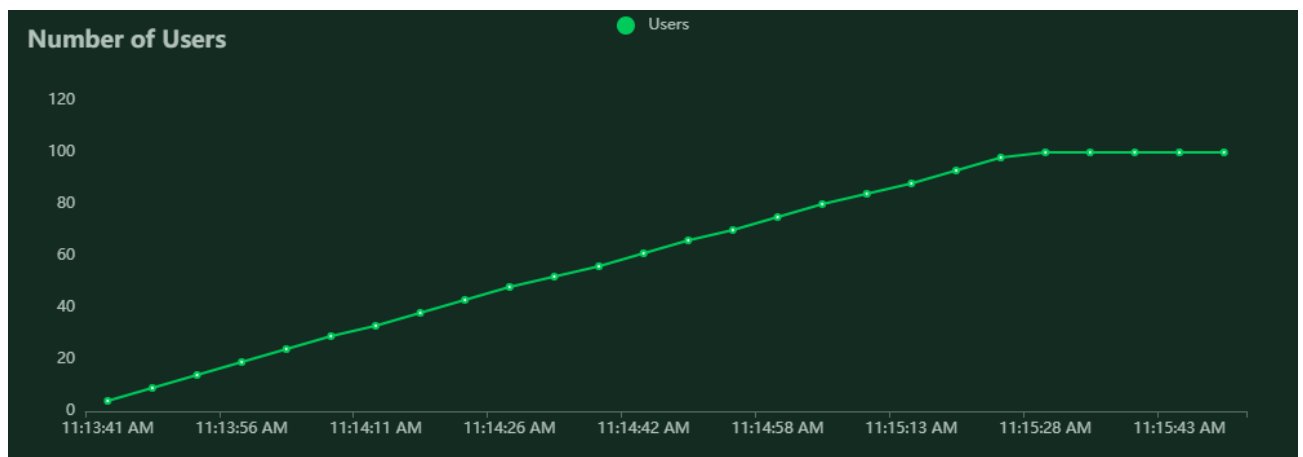
Semua fitur fungsional bisa berjalan dengan lancar secara realtime, baik ketika diakses menggunakan aplikasi client maupun diakses menggunakan request.

Tabel 1. Fitur Fungsional

Feature	Method	Detail
Create	Post	Aplikasi dapat membuat dan menyimpan produk beserta detailnya dengan akurat baik ketika dilakukan melalui aplikasi demo client maupun melalui request
Read	Get	Aplikasi dapat menyajikan data semua produk yang ditambahkan user dengan benar dan akurat baik melalui request maupun melalui aplikasi demo client
Update	Put	Aplikasi dapat mengupdate data yang telah ada sebelumnya dengan akurat melalui aplikasi client maupun melalui request
Delete	Delete	Aplikasi dapat menghapus produk beserta detailnya ketika user menekan tombol delete pada aplikasi client atau mengirim request untuk delete

D. Pengujian

Untuk pengujian, digunakan metode loadtesting untuk mengetahui performa aplikasi yang telah dibuat dengan mengirimkan request dan response berformat JSON. Pengujian dilakukan menggunakan locust dengan mensimulasikan aktivitas user selama berada di aplikasi. Locust akan mengenerate swarm (user) secara berkala selama locust dijalankan sesuai dengan skema spawn user yang digunakan. Digunakan satu komputer sebagai server lokal dengan spesifikasi processor berkecepatan 2Ghz dan RAM sebesar 4GB. Grafik jumlah user selama testing bisa dilihat pada Gambar 21.



Gambar 21. Grafik users

Uji dilakukan dengan mensimulasikan 1 user login hingga 100 user login seperti grafik di atas. Jumlah user akan bertambah secara berkala hingga mencapai 100. Tiap user melakukan registrasi, login, add product, edit product, delete product, dan operasi lainnya hingga logout.

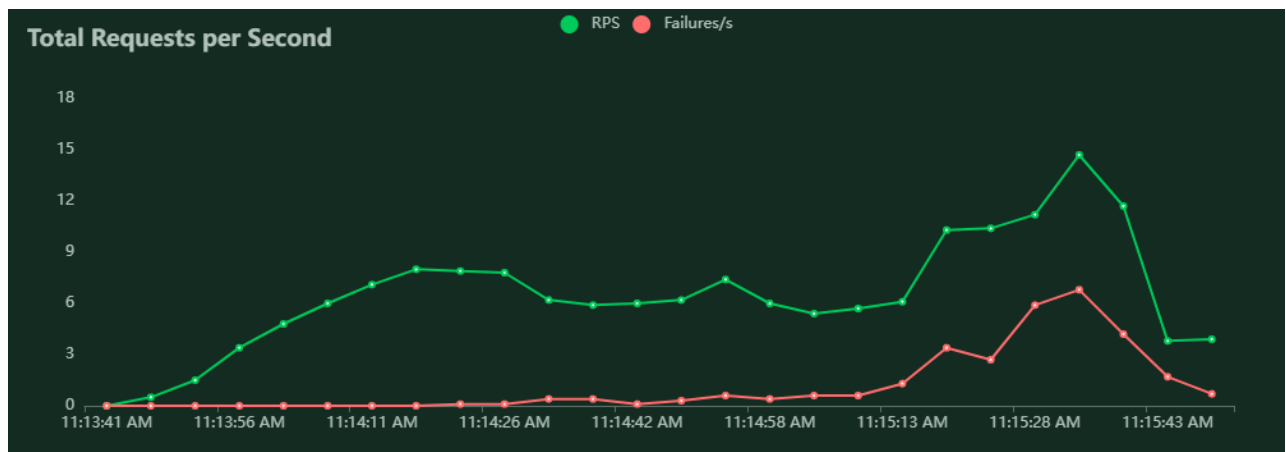
1) Request

Request disini adalah proses permintaan dari sisi client ke server. RPS adalah request per second, merepresentasikan jumlah request tiap detiknuya. Tabel 2 menggambarkan keseluruhan request dari 1 user hingga 100 user.

Tabel 2. Request Statistics

Method	Name	# Requests	# Fails	Average (ms)	Min (ms)	MAX (MS)	Avg size (bytes)	RPS	Failure
POST	/add	95	54	10553	1075	39736	804172	0.7	0.4
POST	/login	134	43	9843	2724	23878	8799	1.0	0.3
POST	/join	146	0	10393	3161	25810	2686	1.1	0.0
GET	/logout	50	0	1613	1009	3237	2686	0.4	0.0
GET	/delete0	60	0	4034	2019	9206	68	0.5	0.0
GET	/update0	74	47	11753	2343	24596	12420	0.6	0.4
GET	/viewall	100	32	13261	3032	51419	12380472	0.8	0.2
GET	/	132	0	1586	1008	3286	3443	1.0	0.0
GET	/view	172	20	5359	2020	24470	396717	1.3	0.2
Aggregated		963	196	7776	1008	51419	1439005	7.2	1.5

Berdasarkan tabel statistik diatas, diperoleh 20.35% failure dari keseluruhan request (963 request), dan 20.85% failure berdasarkan RPS, dengan average response time 7776ms. Grafik RPS bisa dilihat pada Gambar 22.



Gambar 22. Grafik request per second

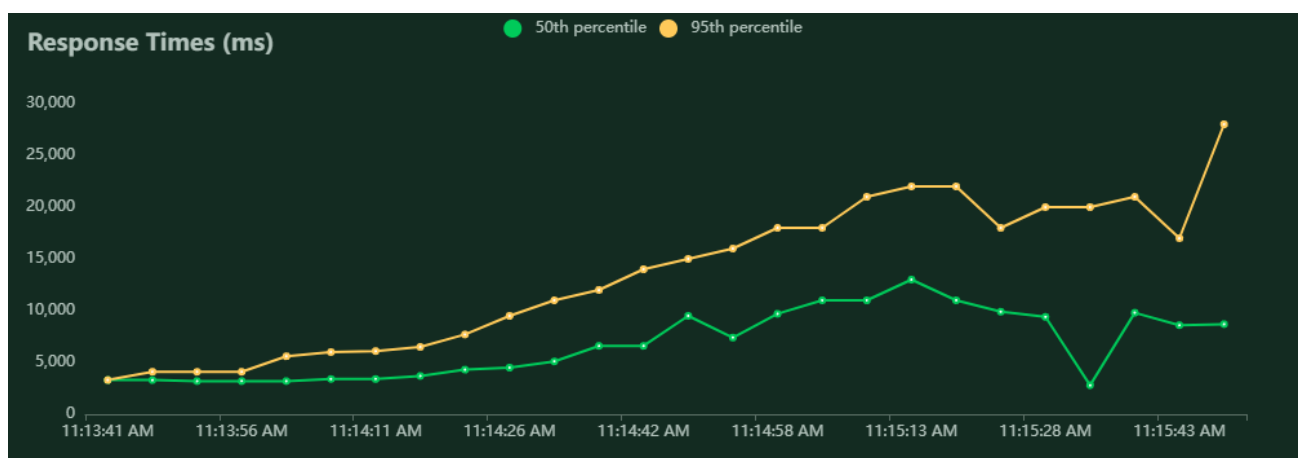
Failure mulai muncul setelah aplikasi menerima RPS pada angka tertentu. Pada kasus ini, failure muncul sesaat setelah menerima 8 RPS dengan total ± 50 user aktif di waktu yang sama (cek Gambar 21 dan 22).

2) Response Time

Lama response time sebanding dengan besar resource yang diproses, dalam kasus ini fungsi /viewall mempunyai response time paling lama karena memproses data besar untuk menampilkan semua produk dari semua user beserta detailnya. Sedangkan response time paling cepat adalah route untuk mengakses index dan logout karena memang proses yang dijalankan membutuhkan resource yang sangat sedikit.

Tabel 3. Response Time

Method	Name	50%ile (ms)	60%ile (ms)	70%ile (ms)	80%ile (ms)	90%ile (ms)	95%ile (ms)	99%ile (ms)	100%ile (ms)
GET	/	1400	1500	1900	2100	2400	3000	3200	3300
POST	/add	6400	10000	15000	21000	25000	27000	40000	40000
GET	/delete0	3100	3300	4900	6200	8300	8600	9200	9200
POST	/join	10000	12000	13000	16000	18000	22000	26000	26000
POST	/login	9700	11000	12000	14000	16000	19000	23000	24000
GET	/logout	1400	1500	2000	2300	2700	3000	3200	3200
GET	/update0	10000	13000	15000	19000	23000	24000	25000	25000
GET	/view	3700	4300	5500	8100	11000	14000	22000	24000
GET	/viewall	12000	14000	17000	20000	23000	27000	51000	51000
	Aggregated	4700	7500	10000	13000	18000	22000	27000	51000



Gambar 23. Grafik response times

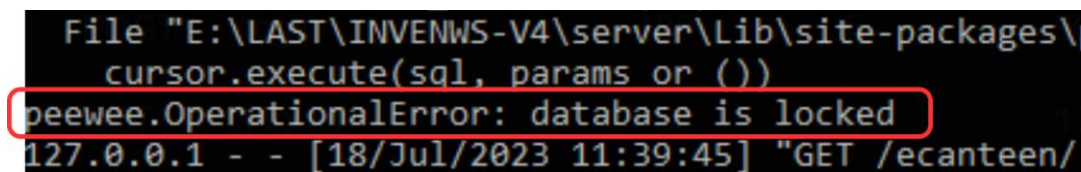
3) Failures

Semua failure pada request adalah internal server error yang terjadi pada beberapa route yang melibatkan database SQLite. Failure ini disebabkan oleh terkuncinya database SQLite ketika terbebani beberapa proses tertentu dalam waktu bersamaan.

Tabel 4. Failures Statistics

Method	Name	# Requests	# Fails
POST	/add	500 Server Error: INTERNAL SERVER ERROR for url: /add	54
POST	/login	500 Server Error: INTERNAL SERVER ERROR for url: /login	43
GET	/view	500 Server Error: INTERNAL SERVER ERROR for url: /view	20
GET	/update0	500 Server Error: INTERNAL SERVER ERROR for url: /update0	47
GET	/viewall	500 Server Error: INTERNAL SERVER ERROR for url: /viewall	32

Pada fase awal database tidak langsung terkunci, melainkan mengalami pending transaction, database baru terkunci total setelah database menerima banyak beban. Ketika sudah terkunci, database tidak bisa langsung terunlock dengan instan meskipun telah digunakan library ORM. Inilah yang menyebabkan tiap request pasti failure pada momen tertentu setelah database terkunci, dalam kasus ini failure mulai terjadi ketika aplikasi mencapai RPS diatas 10 dan ± 100 user aktif dalam waktu bersamaan. Hasil yang berbeda akan muncul apabila pengujian dilakukan pada server yang berbeda spesifikasi hardwarenya.



Gambar 24. Database is locked

Alasan database bisa terkunci adalah karena pada dasarnya SQLite hanya mendukung akses satu proses tulis (write) dan banyak proses baca (read) secara bersamaan. Jika satu proses sedang menulis ke database, proses lain harus menunggu sampai penulisan selesai sebelum dapat membaca atau menulis data. Inilah yang menyebabkan pending transaction, write-lock, hingga database lock.

IV. SIMPULAN

Aplikasi inventory berbasis web service telah berhasil dibuat dan tiap fitur fungsional CRUD sudah bisa berjalan dengan baik menggunakan model arsitektur webservice (client-server). Dengan bentuk arsitektur ini aplikasi akan lebih mudah scale-up dan diintegrasikan dengan aplikasi lain. Ketika dihadapkan dengan beban tertentu, terjadi failure yang diakibatkan oleh terkuncinya database SQLite karena keterbatasan database dalam menerima proses. Namun, apabila user login tidak bersamaan, dan pemakaian aplikasi normal dan tidak padat, aplikasi akan tetap berjalan normal dan database tidak akan terkunci. Aplikasi tetap mempunyai potensi besar untuk diintegrasikan dengan aplikasi lain dengan skala pengguna kecil menengah.

Saran solusi dan alternatif pengembangan lainnya untuk skala yang lebih besar atau global adalah dengan menambahkan fitur load balancing menggunakan algoritma load balance seperti algoritma Round Robin, Least Connections, penggunaan library tambahan PyLB ataupun aplikasi load balancing seperti NGINX.

Penggunaan model arsitektur webservice pada aplikasi inventory e-canteen ini menjadikan aplikasi dan database relatif tahan beban apabila ada banyak request beragam dalam waktu bersamaan dibandingkan dengan arsitektur monolith karena ada pembagian sisi client dan server sedangkan monolith hanya ada dalam satu wadah. Alternatif model arsitektur lainnya adalah model arsitektur microservice dimana tiap proses dan fungsi terbagi menjadi bagian yang lebih kecil lagi dan tetap independen.

REFERENSI

- [1] T. Handayani, A. H. Furqon, and S. Supriyono, "Rancang Bangun Sistem Inventori Pengendalian Stok Barang Berbasis Java Pada PT Kalibesar Artah Perkasa," *sitech*, vol. 3, no. 1, pp. 35–40, Jun. 2020, doi: 10.24176/sitech.v3i1.4884.
- [2] D. Myers and Mario Santana Quintero, D. Myers, A. Dalgity, and I. Avramides, "The Arches heritage inventory and management system: a platform for the heritage field," *Journal of Cultural Heritage Management and Sustainable Development*, vol. 6, no. 2, pp. 213–224, Aug. 2016, doi: 10.1108/JCHMSD-02-2016-0010.
- [3] A. O. Sari and E. Nuari, "RANCANG BANGUN SISTEM INFORMASI PERSEDIAAN BARANG BERBASIS WEB DENGAN METODE FAST(FRAMEWORK FOR THE APPLICATIONS)".
- [4] G. Blinowski, A. Ojdowska, and A. Przybylek, "Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation," *IEEE Access*, vol. 10, pp. 20357–20374, 2022, doi: 10.1109/ACCESS.2022.3152803.
- [5] G. Barbaglia, S. Murzilli, and S. Cudini, "Definition of REST web services with JSON schema: Definition of REST Web-Services with JSON Schema," *Softw. Pract. Exper.*, vol. 47, no. 6, pp. 907–920, Jun. 2017, doi: 10.1002/spe.2466.
- [6] K. Gottschalk, S. Graham, H. Kreger, and J. Snell, "Introduction to Web services architecture," *IBM Syst. J.*, vol. 41, no. 2, pp. 170–177, 2002, doi: 10.1147/sj.412.0170.
- [7] Y. Fauziah, "Aplikasi Iklan Baris Online menggunakan Arsitektur REST Web Service," *Telematika*, vol. 9, no. 2, Sep. 2014, doi: 10.31315/telematika.v9i2.286.
- [8] M. I. Dwitama and I. E. Rosely, "ORSA – APLIKASI KANTIN PINTAR PADA MODUL KASIR & TENAN".
- [9] J. S. Pasaribu, "Development of a Web Based Inventory Information System," *Int. J. Eng. Scie. and Inform. Technology*, vol. 1, no. 2, pp. 24–31, Mar. 2021, doi: 10.52088/ijesty.v1i2.51.
- [10] I. Syarif, "SISTEM INFORMASI INVENTORY BARANG PADA APOTEK SULTAN MENGGUNAKAN METODE FIRST-IN FIRST-OUT (FIFO)".
- [11] A. Oktaviani, M. Nogie, and D. Novianti, "WEB-BASED EQUIPMENT INVENTORY INFORMATION SYSTEM IN THE SERVICE DIVISION OF PT ARISTA SUKSES MANDIRI JAKARTA," *jri*, vol. 3, no. 1, pp. 31–38, Dec. 2020, doi: 10.34288/jri.v3i1.174.
- [12] M. G. L. Putra and M. I. A. Putera, "ANALISIS PERBANDINGAN METODE SOAP DAN REST YANG DIGUNAKAN PADA FRAMEWORK FLASK UNTUK MEMBANGUN WEB SERVICE," 2019.
- [13] P. Vogel, T. Klooster, V. Andrikopoulos, and M. Lungu, "A Low-Effort Analytics Platform for Visualizing Evolving Flask-Based Python Web Services," in *2017 IEEE Working Conference on Software Visualization (VISOFT)*, Shanghai: IEEE, Sep. 2017, pp. 109–113. doi: 10.1109/VISOFT.2017.13.
- [14] A. Kadiyala and A. Kumar, "Applications of Python to evaluate environmental data science problems," *Environ. Prog. Sustainable Energy*, vol. 36, no. 6, pp. 1580–1586, Nov. 2017, doi: 10.1002/ep.12786.
- [15] I. O. Suzanti, N. Fitriani, A. Jauhari, and A. Khozaimi, "REST API Implementation on Android Based Monitoring Application," *J. Phys.: Conf. Ser.*, vol. 1569, no. 2, p. 022088, Jul. 2020, doi: 10.1088/1742-6596/1569/2/022088.
- [16] Z. U. Haq, G. F. Khan, and T. Hussain, "A Comprehensive analysis of XML and JSON web technologies," *Signal Processing*.
- [17] V. R. Vyshnavi and A. Malik, "Efficient Way of Web Development Using Python and Flask," vol. 6, no. 2, 2019.
- [18] N. Chauhan, M. Singh, A. Verma, A. Parasher, and G. Budhiraja, "Implementation of database using python flask framework: college database management system," *int. jour. eng. com. sci*, vol. 8, no. 12, pp. 24894–24899, Dec. 2019, doi: 10.18535/ijecs/v8i12.4390.
- [19] C. Hummert and D. Pawlaszczyk, Eds., *Mobile Forensics – The File Format Handbook: Common File Formats and File Systems Used in Mobile Devices*. Cham: Springer International Publishing, 2022. doi: 10.1007/978-3-030-98467-0.
- [20] Z. M. Jiang and A. E. Hassan, "A Survey on Load Testing of Large-Scale Software Systems," *IEEE Trans. Software Eng.*, vol. 41, no. 11, pp. 1091–1118, Nov. 2015, doi: 10.1109/TSE.2015.2445340.

Conflict of Interest Statement:

The author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.