JTRISTE, Vol.12, No.2, Oktober 2025, pp. 56~68
ISSN: **2355-3677 | e-ISSN : 2460-8548**

ANALISIS DAN OPTIMASI KINERJA APLIKASI POTIO MENGGUNAKAN LOAD TESTING JMETER

Meike Garjitno[1], Husni Angriani[2*], Yeni Saharaeni[3]
[1,2,3]Program Studi Sistem Informasi, STMIK Kharisma Makassar
e-mail: meikegarjitno_22@kharisma.ac.id, [2]husniangriani@kharisma.ac.id,
[3]yenisaharaeni@kharisma.ac.id

Abstrak

Optimasi database dan kode terbukti menurunkan error hingga 0% serta meningkatkan throughput dan waktu respon aplikasi Potio meskipun beban pengguna tinggi. Potio merupakan aplikasi pengingat konsumsi obat bagi penderita penyakit kronis. Pengujian kinerja dilakukan menggunakan uji beban (load testing) dengan Apache JMeter pada enam titik layanan utama (endpoint) dengan skenario 50, 100, dan 150 pengguna serentak. Hasil awal menunjukkan penurunan performa pada 150 pengguna, ditandai dengan meningkatnya waktu respon, turunnya throughput, dan munculnya error. Dua eksperimen dilakukan, yaitu peningkatan kecepatan jaringan internet serta optimasi database dan kode. Hasil menunjukkan bahwa meskipun peningkatan jaringan memberi dampak positif, optimasi database dan kode lebih konsisten memperbaiki performa, terbukti dari penurunan error hingga 0%. Kesimpulannya, perbaikan sisi sistem lebih efektif dibanding hanya mengandalkan peningkatan jaringan.

Kata kunci: *Aplikasi Potio, Load Testing, Apache JMeter, Waktu Respon, Throughput, Error*

Abstract

Database and code optimizations have been shown to reduce errors by up to 0% and improve throughput and response time for the Potio application, despite high user load. Potio is a medication reminder application for people with chronic diseases. Performance testing was conducted using load testing with Apache JMeter on six primary service endpoints with scenarios of 50, 100, and 150 concurrent users. Initial results showed a performance degradation at 150 users, characterized by increased response time, decreased throughput, and the appearance of errors. Two experiments were conducted: increasing internet network speed and optimizing the database and code. The results showed that while network improvements had a positive impact, database and code optimizations more consistently improved performance, as evidenced by a 0% decrease in errors. In conclusion, system-side improvements were more effective than relying solely on network improvements..

Keywords: *Potio Application, Load Testing, Apache JMeter, Respone Time, Throughput, Error*

1. Pendahuluan

Dalam kehidupan di era modern yang penuh dengan kesibukan dan tekanan, menjaga kesehatan menjadi salah satu tantangan yang sering dihadapi oleh banyak orang seperti jadwal yang padat, tuntutan pekerjaan, dan aktivitas sosial yang membuat seseorang kesulitan memberikan perhatian yang cukup pada kesehatannya [1]. Penyakit kronis seperti diabetes, hipertensi, penyakit jantung, dan penyakit kronis lainnya mengharuskan penderita tersebut

* Corresponding author : Husni Angriani (husniangriani@kharisma.ac.id)

Diterima : *Agustus, 2025* **Disetujui :** *September, 2025* **Dipublikasikan :** *Oktober, 2025*

mengonsumsi obat secara teratur, namun kesibukan seringkali mengakibatkan keterlambatan bahkan tidak sempat untuk mengonsumsi obat tersebut [2]. Aplikasi Potio hadir sebagai solusi yang dirancang untuk menyelesaikan permasalahan tersebut. Aplikasi Potio akan menampilkan notifikasi alarm yang dapat mengingatkan pengguna agar mengonsumsi obat secara tepat waktu, sesuai dengan jadwal yang ditentukan. Aplikasi Potio dapat diunduh secara gratis melalui situs web resmi potio (potio.my.id).

Seiring meningkatnya jumlah pengguna, diperlukan kemampuan aplikasi untuk menangani beban pengguna. Peningkatan kinerja aplikasi penting untuk menjaga kestabilan agar tetap responsif dan optimal. Metode *Load Testing* mampu mengukur respon sistem di bawah berbagai kondisi beban tertentu untuk mengetahui perilaku perangkat lunak saat diakses banyak pengguna secara bersamaan [3]. Melalui *load testing*, pengembang dapat mengukur 3 indikator utama yaitu rata-rata waktu respon (*average*), jumlah *request* yang diproses dalam satuan waktu (*throughput*), dan persentase kesalahan (*error*) pada aplikasi Potio [4]. Salah satu alat yang digunakan adalah Apache JMeter, perangkat lunak open-source yang berfungsi untuk melakukan simulasi beban kerja dengan pengguna virtual dalam jumlah besar guna menguji performa dan kemampuan aplikasi dalam menangani beban [5]. Namun, hingga saat ini belum ada penelitian yang menguji performa aplikasi alarm pengingat obat berbasis load testing secara mendalam, sehingga penelitian ini penting dilakukan.

Tabel 1. Rangkuman Hasil Pengujian Awal Aplikasi Potio

Indikator	*Average*	*Throughput*	*Error*
50 *user*	1405 ms	24.6/*sec*	0.00%
100 *user*	3453 ms	26.1/*sec*	0.00%
150 *user*	4012 ms	24.6/*sec*	6.67%

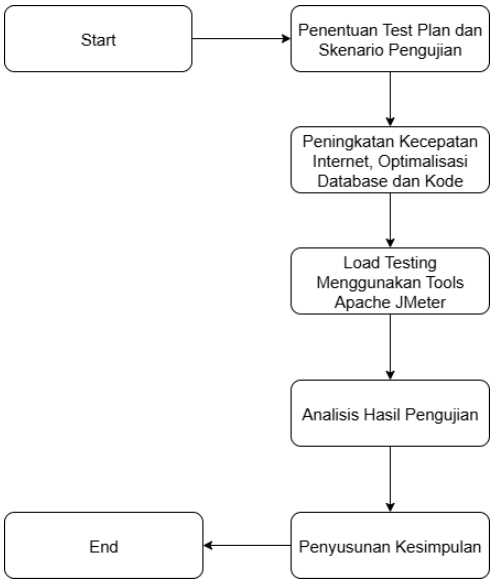
Pengujian awal pada Tabel 1 menggunakan metode Load Testing dengan Apache JMeter dilakukan pada 20 Juni 2025 terhadap enam fungsi Endpoint API aplikasi Potio, yaitu daftar, login, save alarm, save riwayat, get alarm, dan get riwayat. Uji beban dijalankan secara simultan dengan 50, 100, dan 150 pengguna, ramp-up 1 detik, loop count satu kali, menggunakan koneksi internet 16,15 Mbps dan *server* Telkomsel – Sudiang Makassar. *Server* hosting yang digunakan memiliki CPU 1 core, RAM 1 GB, serta SSD Storage untuk mempercepat proses baca tulis data. Hasil pengujian menunjukkan pada 50 pengguna rata-rata waktu respon 1405 ms dengan throughput 24,6/sec tanpa error, pada 100 pengguna meningkat menjadi 3453 ms dengan throughput 26,1/sec tanpa error, dan pada 150 pengguna waktu respon 4012 ms dengan throughput 24,6/sec serta error 6,67%. Dari hasil tersebut dapat disimpulkan bahwa aplikasi Potio mampu menangani hingga 100 pengguna secara bersamaan, namun mengalami penurunan kinerja pada 150 pengguna, sehingga diperlukan peningkatan kecepatan jaringan serta optimasi database dan kode untuk mempercepat waktu respon *server* (Wafiya et al., 2023; Syauqi & Nada, 2023).

Berdasarkan hasil pengujian ini, dapat disimpulkan bahwa aplikasi Potio mampu menangani beban hingga 100 pengguna secara bersamaan, namun mulai menunjukkan penurunan kinerja pada 150 pengguna. Oleh karena itu, diperlukan upaya peningkatan kinerja melalui peningkatan kecepatan jaringan agar *server* dapat melayani permintaan pengguna dengan lebih optimal [6]. Selain itu, optimasi pada *database* dan kode untuk mempercepat waktu respon *server* [7].

Peningkatan kinerja aplikasi Potio bertujuan untuk memastikan aplikasi tetap optimal saat menghadapi beban pengguna yang besar, memberikan pengalaman pengguna yang lebih baik, serta meminimalkan error. Hasil pengujian juga dapat menjadi acuan bagi pengembang dalam menerapkan strategi optimasi kinerja pada aplikasi serupa di masa depan.

2. Metode Penelitian

Dalam penelitian ini, peneliti menggunakan metode *load testing* yang dilakukan pada *tools* Apache JMeter untuk melakukan pengujian terhadap sistem aplikasi Potio dengan menambahkan beban secara bertahap [8]. Penelitian ini dilakukan untuk menganalisis kinerja sistem aplikasi dalam menangani jumlah pengguna yang melakukan satu aktivitas secara bersamaan dalam rentang waktu tertentu [9]. Adapun tahapan penelitian yang dapat dilihat pada Gambar di bawah ini.



Gambar 1. Alur Tahapan Penelitian

Gambar 1 merupakan alur tahapan penelitian *load testing* pada aplikasi Potio. Penelitian ini dimulai dengan penentuan *test plan* (target pengujian) dan skenario pengujian, yang mencakup berbagai aktivitas yang akan diuji seperti jumlah pengguna yang disimulasikan, jenis aktivitas yang dilakukan secara bersamaan, dan parameter lain yang relevan. Setelah itu, dilakukan peningkatan kecepatan internet untuk memastikan koneksi yang stabil, serta optimalisasi *database* dan kode program guna mempercepat waktu respon dan mengurangi beban sistem. Tahap berikutnya adalah melakukan *load testing* menggunakan *tools* Apache

JMeter dengan menambahkan beban secara bertahap melalui peningkatan jumlah pengguna yang mengakses aplikasi Potio secara bersamaan, untuk mengukur sejauh mana aplikasi mampu menangani beban tersebut. Pengujian ini hanya diambil nilai rata-ratanya sehingga hasil yang diperoleh lebih merepresentasikan performa rata-rata aplikasi. Setelah pengujian selesai, dilakukan analisis terhadap hasil *load testing* dengan mengukur indikator seperti waktu respon, *throughput,* dan *error* untuk mengevaluasi performa aplikasi. Terakhir, disusun kesimpulan berdasarkan hasil pengujian dan upaya optimasi yang telah dilakukan.

Penelitian ini menggunakan metode eksperimental untuk mengetahui bagaimana satu perlakuan tertentu dapat mempengaruhi aspek lainnya [10]. Dalam pelaksanaan eksperimen, data dikumpulkan dengan berfokus pada tiga indikator utama, yaitu *Average, Throughput*, dan *Error* karena memberikan gambaran umum kinerja aplikasi [4]. Sumber data hasil pengujian ini merupakan sumber data primer, dimana data tersebut dikumpulkan langsung oleh peneliti setelah melakukan serangkaian pengujian pada aplikasi Potio [11]. Spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam penelitian dapat dlihat pada Tabel 2.

Tabel 2. Spesifikasi Perangkat Keras dan Perangkat Lunak

Komponen	Spesifikasi
Server aplikasi	Hosting (1 Core CPU, 1 GB RAM, SSD Unlimited)
Sistem operasi	Windows 11
Database	MySQLnd 8.1.33
Bahasa pemrograman	PHP 3.7.21, Java (Android Studio)
Tools	Apache JMeter 5.6.3

3. Hasil Dan Pembahasan

3.1. Deskripsi Data

Berdasarkan penelitian yang dilakukan oleh Given dkk, disebutkan bahwa *load time* yang ideal berada di bawah 3 detik [12]. Oleh karena itu, dalam penelitian ini, batas *load time* yang digunakan juga mengacu pada standar tersebut, yaitu tidak lebih dari 3 detik. Untuk mendukung tujuan tersebut, maka *test plan* (target pengujian) aplikasi sebagai berikut :

- *Load time* tidak lebih dari 3 detik.
- Tidak terdapat respon gagal (*error*) dari *request* yang dikirimkan selama pengujian dilakukan.
- Data pengujian berhasil masuk ke dalam fungsi. [8]

Selanjutnya, dilakukan perancangan skenario pengujian agar dapat menguji kinerja aplikasi Potio secara maksimal. Hal ini berfokus pada proses daftar*, login, save* alarm, *save* riwayat, *get* alarm, dan *get* riwayat. Skenario yang telah dirancang akan membantu untuk mengatur variasi waktu yang diperlukan jika terdapat penambahan pengguna atau biasa disebut dengan Periode *Ramp-up,* serta menentukan jumlah perulangan aktivitas yang dilakukan oleh pengguna (*Loop Count*) . Pengujian dilakukan dengan memberikan beban pengguna secara bertahap untuk mengetahui kinerja aplikasi dapat dilihat pada Tabel 3.

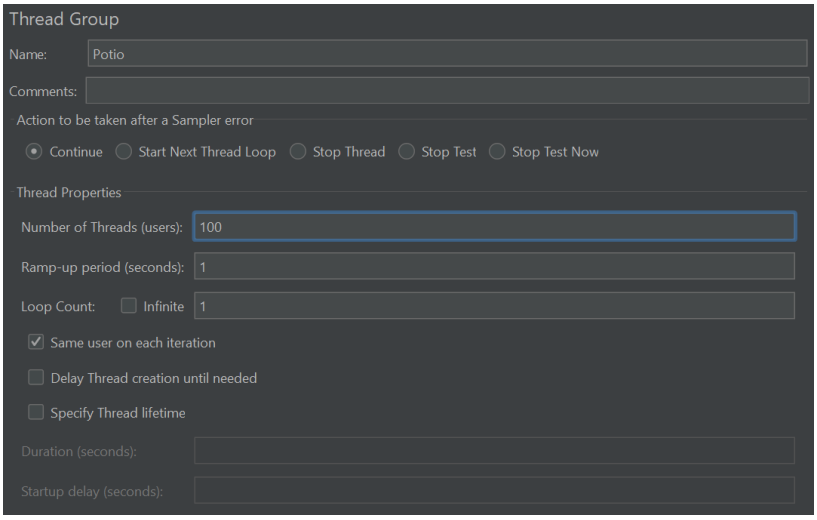
Tabel 3. Skenario Pengujian

Skenario 1	*Number of Threads (users)*	50
	Ramp-up period (seconds)	1
	Loop count	1
Skenario 2	*Number of Threads (users)*	100
	Ramp-up period (seconds)	1
	Loop count	1
Skenario 3	*Number of Threads (users)*	150
	Ramp-up period (seconds)	1
	Loop count	1

3.2. Hasil Pengujian

Langkah - langkah pengujian dan hasil dari pengujian *load testing* yang dilakukan sebagai berikut :

1. Pembuatan dan Konfigurasi *Thread Group*

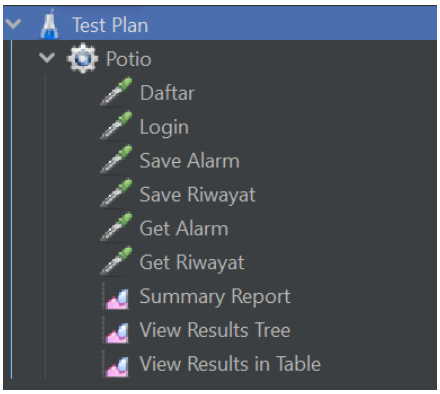


Gambar 2. Konfigurasi parameter pengujian skenario 1

Membuat *thread group* dan mengisi masing-masing parameter berdasarkan skenario yang telah dibuat pada Tabel 3. Pada Gambar 2, dapat dilihat terdapat pengaturan skenario 1 menggunakan 100 *user* dengan *ramp up period* (lama waktu) selama 1 detik dan *loop* (pengulangan) sebanyak 1 kali.

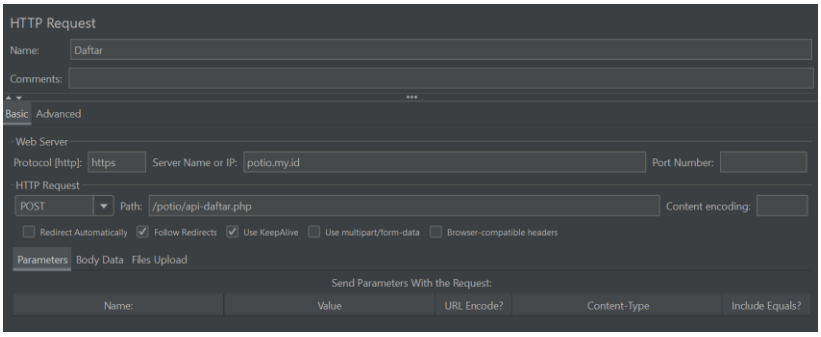
2. Pembuatan dan Konfigurasi HTTP *Request*

Selanjutnya, akan dibuat *sampler* berupa HTTP *Request* untuk setiap *Endpoint* API yang akan diuji. Pengujian ini menguji beberapa *Endpoint* API seperti daftar*, login, save* alarm, *save* riwayat, *get* alarm, dan *get* riwayat.



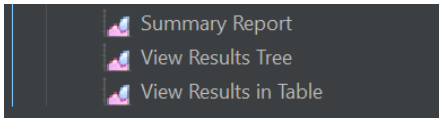
Gambar 3. *Sampler* Pengujian Aplikasi Potio

Kemudian, buat *sampler* HTTP *Request* di *Thread Group* untuk setiap sampel yang akan diuji dengan mengisi konfigurasi berupa Web *Server* yang berisi *Protocol* dan *Server Name or* IP selanjutnya, HTTP *Request* yang berisi metode yang diinginkan dan *patch* yang akan digunakan.



Gambar 4. Konfigurasi HTTP *Request* untuk fungsi Daftar

3. Pembuatan *Event Listener*



Gambar 5. Pembuatan *Event Listener*

Setelah semua fungsi telah dikonfigurasi untuk pengujian perlu di tambahkan beberapa *listener* pada *Thread Group* yakni *Summary Report* untuk menunjukkan hasil rangkuman pengujian. Dalam hal ini, *View Result Tree* berguna untuk menunjukkan detail pengujian, dan *View Result Table* berguna untuk menunjukkan hasil pengujian dalam bentuk tabel [8].

Pengujian dilakukan sebanyak dua kali dengan menguji beberapa aspek yang berbeda, yaitu peningkatan kinerja jaringan internet serta optimalisasi *database* dan basis kode. Adapun hasil pengujian yang diperoleh adalah sebagai berikut:

a) Peningkatan Kecepatan Internet

Peningkatan kecepatan internet dilakukan untuk mengoptimalkan kinerja aplikasi dan mengevaluasi pengaruhnya terhadap proses pengiriman data serta pemrosesan *query* oleh *server* [6]. Pengujian awal dilakukan pada 20 Juni 2025 pukul 5:27 PM dengan kecepatan internet sebesar 16,15 Mbps, menggunakan *server* Telkomsel – Sudiang Makassar. Setelah dilakukan peningkatan, pengujian peningkatan kecepatan internet dilakukan pada 1 Juli 2025 pukul 12:36 PM dengan kecepatan internet meningkat menjadi 100,73 Mbps, menggunakan *server* PT. Telekomunikasi Indonesia – Makassar. Peningkatan sebesar 84,58 Mbps ini diharapkan dapat mempercepat respon sistem dan menurunkan tingkat kesalahan yang terjadi saat proses pengujian performa menggunakan *tools* Apache JMeter, sehingga diperoleh gambaran yang lebih akurat mengenai efektivitas dan efisiensi sistem ketika dijalankan pada kondisi jaringan yang optimal.

Pengujian peningkatan kecepatan internet dengan kecepatan internet sebesar 100,73 Mbps, menggunakan *server* PT. Telekomunikasi Indonesia – Makassar, diperoleh hasil sebagai berikut :

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput
Daftar	50	745	135	1576	427.14	0.00%	21.1/sec
Login	50	861	84	1626	375.02	0.00%	16.2/sec
Save Alarm	50	1073	395	1382	272.02	0.00%	11.7/sec
Save Riwayat	50	1100	841	1382	107.46	0.00%	10.8/sec
Get Alarm	50	971	713	1288	149.57	0.00%	11.1/sec
Get Riwayat	50	847	426	1382	234.19	0.00%	12.5/sec
TOTAL	300	933	84	1626	311.64	0.00%	42.9/sec

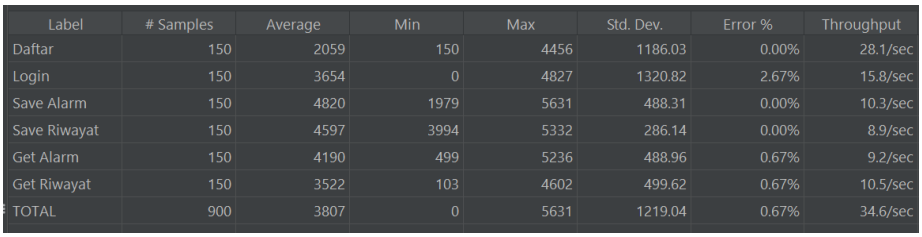
Gambar 6. Hasil Pengujian Skenario 1 Peningkatan Kecepatan Internet

Gambar 6 merupakan hasil pengujian secara simultan menggunakan 50 sampel pada Apache JMeter. Total hasil dari pengujian menyatakan rata-rata waktu respon sebesar 933 ms, *throughput* sebesar 42.9/*sec,* dan tidak terdapat *error* selama pengujian.

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput
Daftar	100	1658	195	3144	984.67	0.00%	24.9/sec
Login	100	2464	354	3061	768.94	0.00%	16.2/sec
Save Alarm	100	3061	2513	3464	200.72	0.00%	11.0/sec
Save Riwayat	100	2873	2355	3281	255.67	0.00%	11.2/sec
Get Alarm	100	2275	1893	3281	383.58	0.00%	12.1/sec
Get Riwayat	100	2021	1738	2672	168.85	0.00%	14.3/sec
TOTAL	600	2392	195	3464	732.13	0.00%	37.0/sec

Gambar 7. Hasil Pengujian Skenario 2 Peningkatan Kecepatan Internet

Gambar 7 merupakan hasil pengujian secara simultan menggunakan 100 sampel pada Apache JMeter. Total hasil dari pengujian menyatakan rata-rata waktu respon sebesar 2392 ms, *throughput* sebesar 37.0/*sec*, dan tidak terdapat *error* selama pengujian.



Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput
Daftar	150	2059	150	4456	1186.03	0.00%	28.1/sec
Login	150	3654	0	4827	1320.82	2.67%	15.8/sec
Save Alarm	150	4820	1979	5631	488.31	0.00%	10.3/sec
Save Riwayat	150	4597	3994	5332	286.14	0.00%	8.9/sec
Get Alarm	150	4190	499	5236	488.96	0.67%	9.2/sec
Get Riwayat	150	3522	103	4602	499.62	0.67%	10.5/sec
TOTAL	900	3807	0	5631	1219.04	0.67%	34.6/sec

Gambar 8. Hasil Pengujian Skenario 3 Peningkatan Kecepatan Internet

Gambar 8 merupakan hasil pengujian secara simultan menggunakan 150 sampel pada Apache JMeter. Total hasil dari pengujian menyatakan rata-rata waktu respon sebesar 3807 ms, *throughput* sebesar 34.6/*sec*, dan terdapat *error* sebesar 0.67%. Rata-rata waktu respon melebihi target dan terdapat *error*, ini menunjukkan aplikasi kurang optimal jika pengguna melebihi 150 *user.* (Tejaya et al., 2023).

Tabel 4. Rangkuman Hasil Peningkatan Kecepatan Internet

Indikator	*Average*	*Throughput*	*Error*
50 *user*	933 ms	42.9/*sec*	0.00%
100 *user*	2392 ms	37.0/*sec*	0.00%
150 *user*	3807 ms	34.6/*sec*	0.67%

b) Optimalisasi *Database* dan Kode

Optimalisasi *database* dan kode dilakukan untuk meningkatkan efisiensi penyimpanan dan mempercepat proses respon data pada *server* [7]. Pada sisi *database*, optimalisasi dilakukan dengan membatasi jumlah karakter pada setiap kolom sesuai kebutuhan, guna mengurangi beban penyimpanan dan mempercepat proses *query*. Sementara itu, di sisi kode PHP, dilakukan validasi tambahan menggunakan fungsi strlen() untuk membatasi panjang karakter input agar tidak melebihi kapasitas kolom pada *database*. Selain itu, digunakan juga fungsi trim() untuk menghapus spasi yang tidak diperlukan pada awal atau akhir *input* [13]. Pendekatan ini tidak hanya menjaga integritas data, tetapi juga membantu mencegah potensi kesalahan saat proses penyimpanan dan pengambilan data dari *server*.

Setelah dilakukan optimalisasi *database* dan kode, didapatkan hasil dari pengujian *Endpoint* API untuk beberapa fungsi pada aplikasi potio sebagai berikut:

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput
Daftar	50	1119	491	2220	372.85	0.00%	18.6/sec
Login	50	871	103	1494	361.50	0.00%	17.0/sec
Save Alarm	50	1154	630	1652	200.12	0.00%	11.8/sec
Save Riwayat	50	1220	876	1746	162.15	0.00%	11.0/sec
Get Alarm	50	1090	625	1564	239.17	0.00%	11.6/sec
Get Riwayat	50	917	407	1537	257.19	0.00%	14.7/sec
TOTAL	300	1062	103	2220	303.96	0.00%	39.5/sec

Gambar 9. Hasil Pengujian Skenario 1 Optimalisasi *Database* dan Kode

Gambar 9 merupakan hasil pengujian secara simultan menggunakan 50 sampel pada Apache JMeter. Total hasil dari pengujian menyatakan rata-rata waktu respon sebesar 1062 ms, *throughput* sebesar 39.5/*sec*, dan tidak terdapat *error* selama pengujian.

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput
Daftar	100	1395	207	2524	697.71	0.00%	28.8/sec
Login	100	1834	174	2510	605.02	0.00%	19.0/sec
Save Alarm	100	2141	667	2568	355.82	0.00%	13.1/sec
Save Riwayat	100	2231	1861	2700	138.73	0.00%	11.1/sec
Get Alarm	100	2026	1718	2554	191.14	0.00%	11.2/sec
Get Riwayat	100	1789	1302	2532	230.46	0.00%	12.1/sec
TOTAL	600	1903	174	2700	507.18	0.00%	44.2/sec

Gambar 10. Hasil Pengujian Skenario 2 Optimalisasi *Database* dan Kode

Gambar 10 merupakan hasil pengujian secara simultan menggunakan 100 sampel pada Apache JMeter. Total hasil dari pengujian menyatakan rata-rata waktu respon sebesar 1903 ms, *throughput* sebesar 44.2/*sec*, dan tidak terdapat *error* selama pengujian.

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput
Daftar	150	2342	215	4969	1311.82	0.00%	25.8/sec
Login	150	3459	161	4470	1042.58	0.00%	15.9/sec
Save Alarm	150	4081	892	4884	530.69	0.00%	10.8/sec
Save Riwayat	150	4406	3705	5096	286.95	0.00%	8.8/sec
Get Alarm	150	3859	3304	5002	379.83	0.00%	9.1/sec
Get Riwayat	150	3576	3204	4796	293.62	0.00%	9.5/sec
TOTAL	900	3621	161	5096	995.91	0.00%	35.8/sec

Gambar 11. Hasil Pengujian Skenario 3 Optimalisasi *Database* dan Kode

Gambar 11 merupakan hasil pengujian secara simultan menggunakan 150 sampel pada Apache JMeter. Total hasil dari pengujian menyatakan rata-rata waktu respon sebesar 3621 ms, *throughput* sebesar 35.8/*sec*, dan tidak terdapat *error* selama pengujian. Rata-rata waktu respon melebihi target pengujian, tetapi sudah tidak terdapat *error* selama pengujian, ini menunjukkan waktu respon kurang optimal jika pengguna melebihi 150 user.

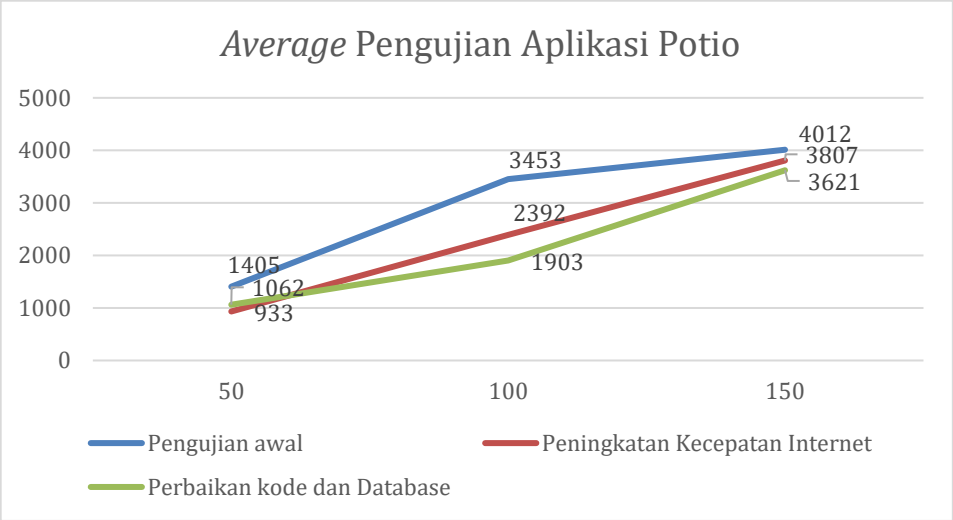
Tabel 5. Rangkuman Hasil Optimalisasi *Database* dan Kode

Indikator	*Average*	*Throughput*	*Error*
50 *user*	1062 ms	39.5/*sec*	0.00%
100 *user*	1903 ms	44.2/*sec*	0.00%
150 *user*	3621 ms	35.8/*sec*	0.00%

3.3. Analisis Hasil Pengujian

1. *Average*

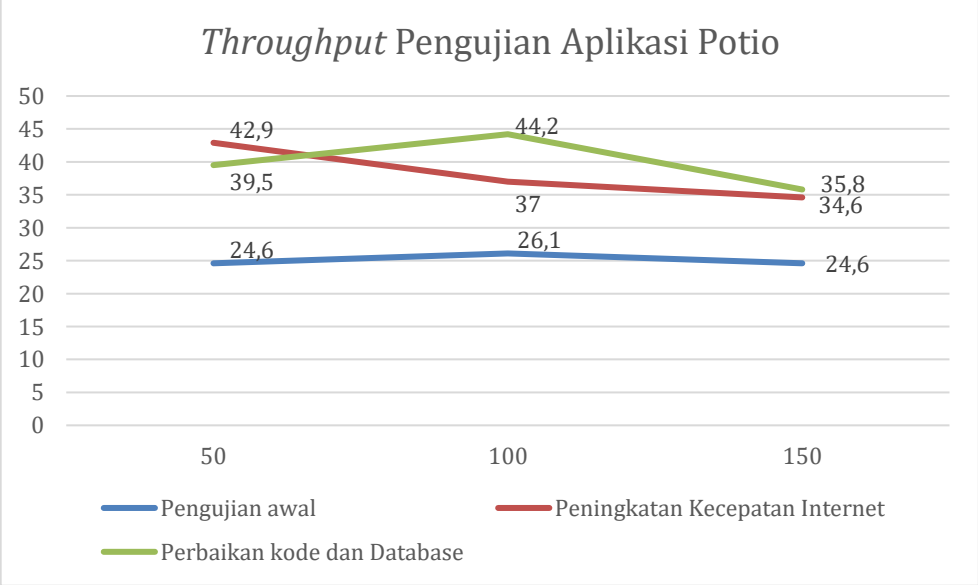
Berdasarkan hasil pengujian yang ditampilkan pada Gambar 13, peningkatan performa aplikasi Potio melalui peningkatan kecepatan internet dan perbaikan kode serta *database* terbukti efektif dalam menurunkan waktu respon. Tanpa optimasi (Tabel 1), waktu respon mencapai 1405 ms pada 50 pengguna, 3453 ms pada 100 pengguna, dan 4012 ms pada 150 pengguna. Setelah peningkatan kecepatan internet (Tabel 4), waktu respon menurun menjadi 933 ms pada 50 pengguna, 2392 ms pada 100 pengguna, dan 3807 ms pada 150 pengguna. Perbaikan paling optimal terjadi setelah optimalisasi kode dan *database* (Tabel 5), dengan waktu respon menjadi 1062 ms pada 50 pengguna, 1903 ms pada 100 pengguna, dan 3621 ms pada 150 pengguna. Meskipun performa meningkat signifikan, waktu respon untuk 150 pengguna masih melebihi target maksimal 3000 ms, sehingga optimalisasi lebih lanjut diperlukan. Sedangkan, optimalisasi *database* dan kode yang telah dilakukan cukup efisien pada pengujian 50–100 pengguna dengan respon *time* sebesar 1903 ms. Grafik peningkatan hasil optimalisasi rata-rata waktu respon (*average*) dapat dilihat pada Gambar 13.



Gambar 12. Grafik *Average* Pengujian Aplikasi Potio

2. *Throughput*

Berdasarkan hasil pengujian yang ditampilkan pada Gambar 14, peningkatan performa aplikasi Potio dilakukan melalui peningkatan kecepatan internet serta perbaikan *database* dan kode. Pada pengujian awal tanpa optimasi (Tabel 1), *throughput* atau jumlah *request* yang berhasil diproses per detik masih tergolong rendah dan stabil, yaitu sebesar 24,6/*sec* pada 50 pengguna, 26,1/*sec* pada 100 pengguna, dan 24,6/*sec* pada 150 pengguna. Setelah dilakukan peningkatan kecepatan internet (Tabel 4), *throughput* meningkat menjadi 42,9/*sec* pada 50 pengguna, 37/*sec* pada 100 pengguna, dan 34,6/*sec* pada 150 pengguna. Setelah optimalisasi *database* dan kode (Tabel 5), dengan *throughput* meningkat menjadi 39,5/*sec* pada 50 pengguna, 44,2/*sec* pada 100 pengguna, dan 35,8/*sec* pada 150 pengguna. Hal ini mengindikasikan bahwa optimalisasi kecepatan internet, *database*, dan kode yang dilakukan cukup efisien, karena jumlah permintaan yang diproses per detik meningkat pada seluruh skenario pengujian, yaitu 50, 100, dan 150 pengguna. Grafik peningkatan *throughput* dapat dilihat pada Gambar 14.

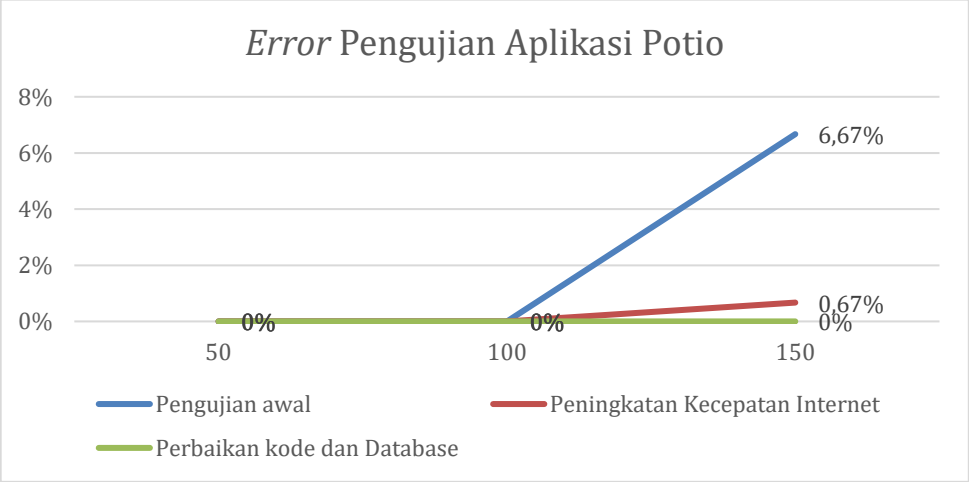


Gambar 13. Grafik *Throughput* Pengujian Aplikasi Potio

3. *Error*

Berdasarkan hasil pengujian yang ditampilkan pada Gambar 15, tingkat *error* aplikasi relatif rendah di semua skenario. Pada pengujian awal, tidak ada *error* pada 50 dan 100 pengguna, namun meningkat menjadi 6,67% saat mencapai 150 pengguna. Setelah peningkatan kecepatan internet, *error* pada 150 pengguna turun menjadi 0,67% yang sebelumnya 6.67%, sementara 50 dan 100 pengguna tetap 0%. Pengujian dengan perbaikan kode dan *database* menunjukkan hasil paling stabil, dengan *error* 0% di semua jumlah pengguna, termasuk 150. Ini menunjukkan bahwa perbaikan kode dan *database* tidak hanya meningkatkan performa, tetapi juga meminimalkan *error*

hingga nol, bahkan di beban tinggi. Untuk melihat grafik *error* pengujian pada aplikasi Potio dapat dilihat pada Gambar 15.



Gambar 14. Grafik *Error* Pengujian Aplikasi Potio

4. Kesimpulan

Penelitian ini menggunakan metode *Load Testing* terhadap *Endpoint* API aplikasi Potio dengan *tools* Apache JMeter. *Load Testing* dilakukan dengan menguji 6 fungsi *Endpoint* API aplikasi yaitu, daftar, login, *save* alarm, *save* riwayat, *get* alarm, dan *get* riwayat, serta menggunakan 50, 100, dan 150 sampel yang dijalankan secara simultan dengan periode *ramp-up* 1 detik dan *loop count* adalah 1. Setelah dilakukan upaya optimasi melalui dua cara, yaitu peningkatan kecepatan internet dan perbaikan kode serta *database*, diperoleh peningkatan performa yang cukup signifikan. Peningkatan kecepatan internet berhasil menurunkan waktu respon dan meningkatkan *throughput*, namun hasilnya belum stabil terutama saat beban pengguna tinggi. Perbaikan pada sisi kode dan struktur *database* menunjukkan hasil yang lebih konsisten, di mana waktu respon rata-rata lebih rendah, *throughput* meningkat pada seluruh jumlah pengguna, serta *error rate* dapat ditekan hingga 0% bahkan saat aplikasi diuji dengan 150 pengguna secara bersamaan. Meskipun waktu respon pada 150 pengguna masih sedikit melebihi batas 3 detik, data pengujian berhasil masuk ke dalam fungsi tanpa kegagalan, dan dua dari tiga target uji telah terpenuhi sepenuhnya. Dengan demikian, optimasi dari sisi sistem terbukti lebih efektif dalam meningkatkan performa aplikasi secara keseluruhan dibandingkan hanya mengandalkan peningkatan jaringan.

5. Saran

Pengembangan aplikasi Potio difokuskan pada optimalisasi sistem melalui penyempurnaan logika pemrograman, efisiensi query, dan perbaikan struktur database agar lebih responsif. Seiring bertambahnya pengguna, diperlukan peningkatan performa server, optimalisasi kode, serta penerapan cache untuk mengurangi beban akses data. Langkah ini diharapkan mampu meningkatkan efisiensi respon dan mendukung skalabilitas aplikasi.

DAFTAR PUSTAKA

[1] A. Y. Setyawan, M. Wahyu, S. Utomo, I. Herdianto, N. K. H, and H. Wisnu, "Implementasi Sistem Aplikasi Klinik Kesehatan," *Semin. Nas. Inform. Bela Negara*, vol. 4, pp. 308–314, 2024.

[2] N. Ariati, J. R. Coyanda, and F. Antony, "Sistem Informasi Reminder Pasien Penyakit Kronis dan Imunisasi Menggunakan WhatsApp Gateway," *MALCOM Indones. J. Mach. Learn. Comput. Sci.*, vol. 4, no. 1, pp. 198–204, 2024, doi: 10.57152/malcom.v4i1.1051.

[3] D. Wardhani, "KOMPARASI ANIMASI FORMAT LOTTIE , JSON , GIF DAN MP4," vol. 1, no. 1, pp. 64–72, 2024.

[4] B. Sabella, R. R. Tina, F. Achmad, A. Lestari, and F. Fathurrahmani, "Pengujian Aplikasi SIHARAPAN Menggunakan Metode Stress Testing," *J. Teknol. Inf. dan Terap.*, vol. 11, no. 1, pp. 7–14, 2024, doi: 10.25047/jtit.v11i1.359.

[5] I. Hamidah, I. Haromain, and I. M. Drehem, "Journal of Digital Business and Technology Innovation (DBESTI) MENUNJANG STABILITAS APLIKASI LAYANAN PERBANKAN PADA PT BANK RAKYAT INDONESIA TBK," vol. 2, no. 1, pp. 114–126, 2025, doi: https://doi.org/10.54914/dbesti.v2i1.1621.

[6] N. Wafiya, R. Afwani, and B. Irmawati, "Pengujian Kinerja, Beban, Dan Stres Website Sistem Informasi Akademik Universitas Mataram," 2023.

[7] A. . Syauqi and N. . Nada, "Analisis Kinerja Website Informatika UPGRIS melalui Pengujian Performa Menggunakan JMeter," *Semin. Nas. Inform. – FTI UPGRIS*, vol. 2023, pp. 965–971, 2023.

[8] W. Tejaya, S. Rahman, and A. Munir, "PENGUJIAN WEBSITE INVITEES MENGGUNAKAN METODE LOAD TESTING DENGAN APACHE JMETER," *KHARISMA Tech*, vol. 18, no. 1, pp. 99–112, 2023, doi: 10.55645/kharismatech.v18i1.305.

[9] Arlinta Christy Barus, Johannes Harungguan, and Efren Manulu, "Pengujian Api Website Untuk Perbaikan Performansi Aplikasi Ditenun," *J. Appl. Technol. Informatics Indones.*, vol. 1, no. 2, pp. 14–21, 2022, doi: 10.54074/jati.v1i2.33.

[10] S. A. Ully and I. P. Dewi, "Pengaruh Game-Based Learning Menggunakan Aplikasi Quizizz Terhadap Hasil Belajar," *Voteteknika (Vocational Tek. Elektron. dan Inform.*, vol. 10, no. 4, p. 79, 2022, doi: 10.24036/voteteknika.v10i4.120039.

[11] M. A. Firmansyah and A. M. Bakti, "Implementasi Metode Fast Untuk Pengembangan Sistem Simpan Pinjam Pada Koperasi Tarbiyah Berbasis Android," *J. Softw. Eng. Ampera*, vol. 3, no. 3, pp. 133–144, 2022, doi: 10.51519/journalsea.v3i3.243.

[12] V. Given, D. P. Mankame, B. Patil, and V. Dhavalgi, "International Journal of Research Publication and Reviews Progressive Web apps : Optimizing Mobile User Experience," vol. 5, no. 5, pp. 4141–4144, 2024.

[13] A. Budi Santoso, *Belajar Pemrograman Web 1 - Dasar PHP dengan Bootstrap MySQLi (Teori dan Praktek)*. 2023. [Online]. Available: http://dx.doi.org/10.1016/j.bpj.2015.06.056%0Ahttps://academic.oup.com/bioinformatics/article-abstract/34/13/2201/4852827%0Ainternal-pdf://semisupervised-3254828305/semisupervised.ppt%0Ahttp://dx.doi.org/10.1016/j.str.2013.02.005%0Ahttp://dx.doi.org/10.10