JITET (Jurnal Informatika dan Teknik Elektro Terapan)
Vol. 14 No. 3, pISSN: 2303-0577 eISSN: 2830-7062 http://dx.doi.org/10.23960/jitet.v14i3.9647

PENERAPAN SINGLETON PATTERN PADA SISTEM AUTENTIKASI JWT UNTUK MENINGKATKAN PERFORMA APLIKASI

Rizky Fahrureza[1*], Muhammad Rizqi Fadhilah[2]

[1]Teknik Informatika, Universitas Singaperbangsa Karawang; Jl. HS.Ronggo Waluyo, Puseurjaya, Telukjambe Timur, Karawang, Jawa Barat 41361
[2]Teknik Informatika, Universitas Singaperbangsa Karawang; Jl. HS.Ronggo Waluyo, Puseurjaya, Telukjambe Timur, Karawang, Jawa Barat 41361

Keywords:
Singleton Pattern;
JWT;
Autentikasi;
Performa; Skalabilitas;
Penggunaan Memori.

Corespondent Email:
2210631170149@student.unsika.ac.id

Abstrak. Penelitian ini bertujuan untuk menganalisis penerapan Singleton Pattern pada sistem autentikasi berbasis JWT dan dampaknya terhadap performa aplikasi. Sistem autentikasi berbasis JWT menawarkan keamanan yang baik dengan menggunakan token akses yang efisien, namun tantangan muncul pada pengelolaan sesi dan alokasi sumber daya. Dalam eksperimen ini, dua skenario diuji: sistem tanpa penerapan Singleton Pattern dan sistem dengan penerapan Singleton Pattern pada JWT Validator dan Session Manager. Hasil penelitian menunjukkan peningkatan kinerja yang signifikan, dengan pengurangan waktu respon 65%, pengurangan penggunaan memori 55,3%, serta peningkatan throughput 186,8%. Skalabilitas meningkat dengan sistem mampu mempertahankan performa lebih stabil meski menghadapi beban lebih tinggi.

Copyright © 2026 The Author(s), Published by Jurnal Informatika dan Teknik Elektro Terapan. This is an open-access article distributed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0).

Abstract. *This study aims to analyze the application of Singleton Pattern in JWT-based authentication systems and its impact on application performance. Two scenarios were tested: without Singleton Pattern and with Singleton Pattern on JWT Validator and Session Manager. Results show significant performance improvements: 65% response time reduction (5.06 ms to 1.77 ms), 55.3% memory usage reduction (28.4 MB to 12.7 MB), and 186.8% throughput improvement (197 to 565 req/s). Scalability also improved with 65% performance retention at 1000 concurrent users. Singleton Pattern proves effective in managing resources efficiently and improving JWT-based authentication performance.*

1. PENDAHULUAN

Di era digital yang terus berkembang, sistem informasi menjadi tulang punggung berbagai aktivitas dalam bisnis, pemerintahan, pendidikan, hingga kehidupan sehari-hari. Keamanan sistem informasi memiliki peran penting untuk menjaga kerahasiaan, integritas, dan ketersediaan data, sekaligus memastikan kelancaran operasional tanpa hambatan. Dengan meningkatnya ancaman keamanan seperti akses tidak sah, peretasan, dan pencurian data, perlindungan sistem informasi menjadi kebutuhan mendesak.

Keamanan sistem informasi didasarkan pada tiga pilar utama: Kerahasiaan (Confidentiality), Integritas (Integrity), dan Ketersediaan (Availability) [1].

Sistem autentikasi yang efektif meningkatkan keamanan aplikasi dengan mengendalikan akses pengguna, mengurangi ancaman seperti Keystroke Logging, Duplicate Login Pages, dan Shoulder Surfing, serta menyediakan mekanisme kuat yang beradaptasi dengan strategi serangan cyber yang berkembang [2].

Tantangan dalam manajemen sesi pengguna termasuk penanganan urutan/panjang sesi, urutan interaksi internal, jenis tindakan, ketersediaan informasi pengguna, dan struktur data. Faktor-faktor ini mempersulit pembelajaran ketergantungan dan memengaruhi kinerja aplikasi [3].

Penelitian ini menggunakan JWT (JSON Web Token) yang menawarkan keuntungan seperti skalabilitas, pengurangan beban server, dan peningkatan keamanan melalui token akses berumur pendek [4]. Penelitian ini bertujuan untuk menganalisis dampak penerapan Singleton Pattern terhadap performa sistem autentikasi berbasis JWT, mencakup waktu respon, penggunaan memori, throughput, dan skalabilitas.

2. TINJAUAN PUSTAKA

2.1. Pola Desain

Pola desain adalah solusi intelektual yang mengatasi masalah pengembangan perangkat lunak umum, meningkatkan desain sistem melalui pedoman yang ditetapkan [5]. Pola Singleton membatasi instansiasi kelas ke satu instance, memastikan satu titik kontrol untuk mengelola sesi pengguna dan proses autentikasi. Hal ini meningkatkan kinerja aplikasi dengan mengurangi penggunaan memori dan waktu instansiasi, menyebabkan waktu respons yang lebih cepat dan peningkatan throughput [6].

2.2. Sistem Autentikasi dalam Pengembangan Aplikasi

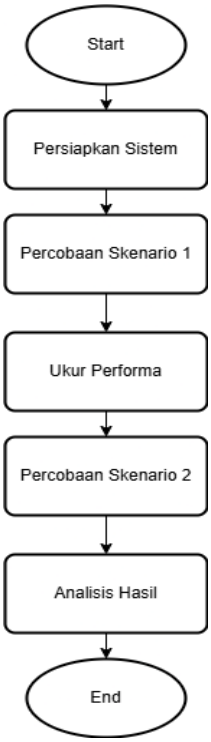
Sistem autentikasi memverifikasi identitas pengguna untuk memastikan hanya pengguna yang sah yang dapat mengakses aplikasi, menggunakan teknik canggih seperti autentikasi dua faktor dan multi-faktor [7]. JSON Web Token (JWT) adalah standar terbuka (RFC 7519) untuk mentransmisikan informasi dengan aman melalui string JSON. JWT terdiri dari Header, Payload, dan Signature yang memastikan integritas data [8]. Penggunaan token akses dan penyegaran meningkatkan pengalaman pengguna dengan memungkinkan autentikasi mulus tanpa sering login [4].

2.3. Optimasi Performa Aplikasi

Kinerja aplikasi dinilai melalui metrik waktu respons, penggunaan memori, throughput, dan skalabilitas [9]. Manajemen memori yang efisien termasuk sistem caching dan penghindaran duplikasi data meningkatkan kecepatan pemrosesan [10]. Throughput mengukur jumlah data yang diproses dalam kerangka waktu tertentu dan sangat penting dalam komputasi awan [11]. Teknik manajemen memori yang efektif menurunkan beban CPU dan memastikan aplikasi berjalan lebih lancar [15].

3. METODE PENELITIAN

Pendekatan penelitian yang digunakan terutama melibatkan metode eksperimental, memungkinkan evaluasi praktis kinerja dan keamanan sistem dalam skenario dunia nyata. Pola Singleton memastikan bahwa hanya satu contoh layanan autentikasi dibuat, meminimalkan konsumsi sumber daya dan mengoptimalkan pengelolaan token autentikasi.



Gambar 1. Flowchart metode penelitian

Gambar 1 menyajikan alur metodologi penelitian secara sistematis. Alur dimulai dari tahap persiapan sistem di mana dua skenario implementasi disiapkan secara paralel: Skenario 1 tanpa Singleton Pattern dan Skenario 2 dengan Singleton Pattern. Setiap skenario melewati tahap percobaan yang identik, kemudian hasil pengukuran performa dari kedua skenario dikumpulkan dan dianalisis secara komparatif. Pendekatan komparasi terkontrol ini memastikan bahwa perbedaan performa yang terukur semata-mata disebabkan oleh penerapan Singleton Pattern, bukan faktor eksternal lain.

Adapun tahapan yang dilakukan berdasarkan Gambar 1 adalah:

1) Start: Persiapan dasar memastikan semua komponen yang diperlukan tersedia dan siap diuji.

2) Persiapkan Sistem: Sistem dibangun dengan dua skenario eksperimen: tanpa Singleton Pattern dan dengan Singleton Pattern pada JWT Validator dan Session Manager [12].

3) Percobaan Skenario 1: Pengujian baseline performa mencakup waktu respon, penggunaan memori, dan throughput sistem tanpa Singleton Pattern.

4) Ukur Performa: Metrik diukur menggunakan alat load testing untuk mensimulasikan beban autentikasi dan memantau penggunaan sumber daya [13].

5) Percobaan Skenario 2: Pengujian setelah penerapan Singleton Pattern pada JWT Validator dan Session Manager untuk memastikan hanya satu instance yang mengelola validasi token dan sesi.

6) Analisis Hasil: Perbandingan antara kedua skenario mengungkapkan implikasi signifikan. Singleton Pattern meminimalkan konsumsi sumber daya dengan memastikan hanya satu instance pengelola autentikasi dibuat [14].

3.1. Arsitektur Sistem

Sistem autentikasi berbasis JWT dalam penelitian ini terdiri dari dua komponen utama: lapisan klien (client layer) dan lapisan layanan backend (backend service layer). Lapisan klien mengirimkan permintaan melalui HTTP REST API ke backend yang memvalidasi kredensial dan menerbitkan token JWT. Backend dibangun secara modular dengan dua komponen kritis yang dioptimasi: JWT Validator dan Session Manager [5].

JWT Validator bertanggung jawab atas verifikasi tanda tangan token, pengecekan masa berlaku, dan dekoding payload JWT. Session Manager mengelola status sesi aktif pengguna. Pada Skenario 1, setiap permintaan menginstansiasi kedua objek secara baru dan independen, menyebabkan overhead signifikan. Pada Skenario 2 dengan Singleton Pattern, keduanya hanya diinstansiasi sekali selama siklus hidup aplikasi dan digunakan bersama oleh seluruh permintaan yang masuk [6].

3.2. Implementasi Singleton Pattern

Singleton Pattern diimplementasikan untuk memastikan kelas JWT Validator dan

Session Manager hanya memiliki satu instance yang dibuat dan diakses secara global. Implementasi menggunakan pendekatan thread-safe dengan mekanisme double-checked locking. Struktur kelas menggunakan atribut privat _instance = None dan method getInstance() yang memeriksa keberadaan instance sebelum membuat yang baru. Kunci kriptografis untuk verifikasi JWT di-load sekali pada inisialisasi dan di-cache dalam instance tunggal tersebut [5].

Tahapan implementasi thread-safe Singleton: (1) pemeriksaan instance pertama tanpa kunci untuk efisiensi; (2) jika belum ada, dilakukan penguncian thread-safe; (3) pemeriksaan kedua setelah penguncian untuk mencegah race condition; (4) instance baru dibuat hanya pada kondisi belum tersedia. Inisialisasi konfigurasi validasi token dan koneksi pool hanya terjadi satu kali, menghilangkan overhead berulang pada setiap permintaan autentikasi [14].

3.3. Spesifikasi Perangkat Pengujian

Seluruh pengujian dilaksanakan pada perangkat yang sama untuk memastikan konsistensi hasil pengukuran. Tabel 1 merinci spesifikasi perangkat keras dan perangkat lunak yang digunakan selama eksperimen.

Komponen	Spesifikasi
Sistem Operasi	Windows 10 Home 64-bit (Build 19045)
Prosesor (CPU)	Intel Core i5-1135G7 (4 Core, 8 Thread, 2,40 GHz)
Memori (RAM)	16 GB DDR4 3200 MHz
Penyimpanan	512 GB SSD NVMe
Bahasa Pemrograman	Python 3.11
Framework Backend	FastAPI 0.103
Alat Load Testing	Apache JMeter 5.6.3
Library JWT	PyJWT 2.8.0

Tabel 1. Spesifikasi Perangkat Pengujian

3.4. Alat dan Metodologi Pengujian

Pengujian beban dilakukan menggunakan Apache JMeter 5.6, alat open-source yang banyak digunakan untuk mengevaluasi performa sistem web. JMeter mensimulasikan concurrent users yang melakukan permintaan autentikasi JWT secara bersamaan. Pemantauan memori dan CPU dilakukan secara real-time menggunakan modul memory_profiler dan psutil di sisi server [9].

Setiap skenario dijalankan dalam tiga fase: (1) warm-up 60 detik dengan 10 concurrent users untuk menstabilkan interpreter; (2) pengujian utama 300 detik dengan variasi 10, 50, 100, 500, dan 1000 concurrent users; (3) cool-down 30 detik untuk observasi perilaku sistem saat beban berkurang. Setiap konfigurasi diulang tiga kali dan hasilnya dirata-ratakan untuk mengurangi variabilitas pengukuran [13].

3.5. Skenario Pengujian

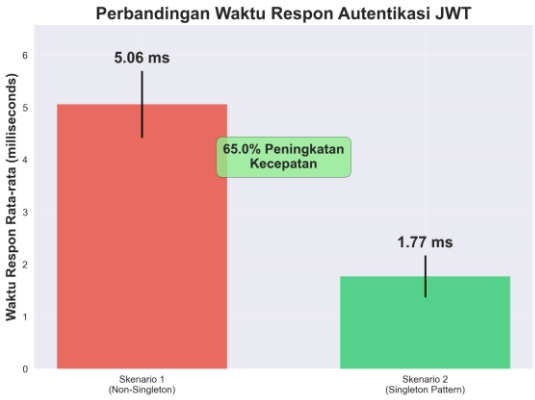
Penelitian ini membandingkan dua skenario yang dirancang untuk mengukur dampak Singleton Pattern secara objektif. Skenario 1 merupakan baseline tanpa Singleton Pattern, di mana setiap permintaan autentikasi memicu instansiasi baru dari JWT Validator dan Session Manager, mencerminkan praktik pengembangan umum tanpa optimasi memori.

Skenario 2 menggunakan Singleton Pattern, di mana JWT Validator dan Session Manager diinstansiasi sekali pada startup dan digunakan bersama oleh seluruh permintaan. Metrik yang diukur meliputi: waktu respon rata-rata (ms), penggunaan memori heap (MB), throughput (req/s), utilisasi CPU (%), dan error rate (%) pada berbagai tingkat concurrent users [10].

4. HASIL DAN PEMBAHASAN

Pengujian dilakukan pada perangkat lokal dengan spesifikasi: OS Windows 10 Home, RAM 16 GB, CPU Intel Core i5-1135G7. Evaluasi dilakukan dengan membandingkan dua skenario: tanpa Singleton Pattern (Skenario 1) dan dengan Singleton Pattern pada JWT Validator dan Session Manager (Skenario 2).

4.1. Waktu Respon



Gambar 2. Diagram perbandingan waktu respon autentikasi JWT

Gambar 2 menampilkan diagram batang yang membandingkan waktu respon rata-rata autentikasi JWT antara Skenario 1 dan Skenario 2. Sumbu vetikal menunjukkan waktu respon dalam milidetik (ms), sedangkan sumbu horizontal membedakan kedua skenario pengujian. Batang Skenario 1 terlihat jauh lebih tinggi (5,06 ms) dibandingkan Skenario 2 (1,77 ms), secara visual mengkonfirmasi pengurangan waktu respon sebesar 65,0%. Perbedaan yang mencolok ini menggambarkan betapa signifikan dampak eliminasi overhead instansiasi berulang terhadap kecepatan pemrosesan setiap permintaan autentikasi.

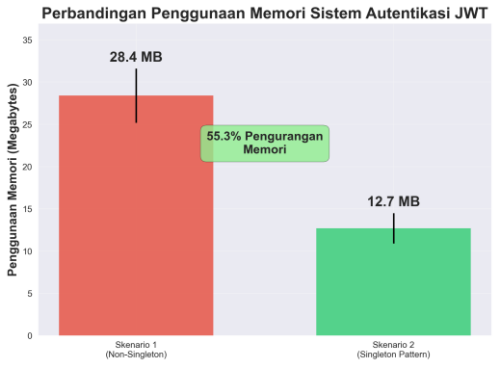
Hal tersebut menunjukkan bahwa waktu rata-rata respon pada Skenario 1 (tanpa Singleton Pattern) adalah 5,06 ms. Sistem tanpa Singleton Pattern menunjukkan waktu respon lebih tinggi karena overhead akibat instansiasi berulang komponen validasi JWT dan pengelolaan sesi.

Pada Skenario 2, waktu respon menurun menjadi 1,77 ms. Singleton Pattern memungkinkan pengelolaan objek yang lebih efisien karena hanya satu instance menangani validasi token dan sesi. Peningkatan kecepatan autentikasi mencapai 65,0% yang disebabkan oleh pengurangan overhead pembuatan instance baru setiap kali permintaan autentikasi terjadi.

Perbedaan waktu respon sebesar 3,29 ms mencerminkan overhead yang dihilangkan melalui eliminasi instansiasi berulang. Pada Skenario 1, setiap permintaan memerlukan alokasi memori baru, pemuatan ulang konfigurasi kriptografis, dan inisialisasi koneksi pool sesi. Proses-proses ini bersifat kumulatif dan semakin terasa pada beban tinggi karena tekanan garbage collector yang membersihkan objek-objek tidak aktif [10].

Hasil ini sejalan dengan temuan Alnuhait et al. [9] bahwa object reuse merupakan strategi paling efektif dalam optimasi waktu respon aplikasi web. Peningkatan 65,0% menunjukkan bahwa manajemen instance melalui Singleton Pattern memberikan dampak nyata pada pengalaman pengguna akhir, terutama pada aplikasi yang memproses volume autentikasi tinggi seperti microservices dan API gateway yang beroperasi dalam skala enterprise.

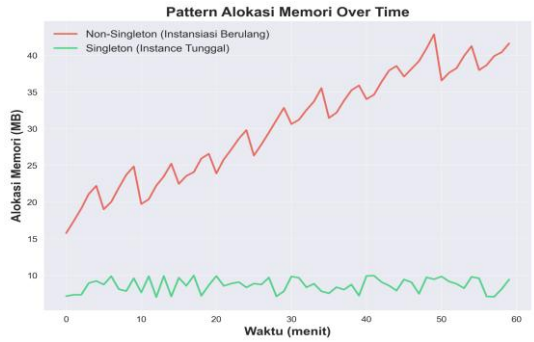
4.2. Penggunaan Memori



Gambar 3. Diagram perbandingan penggunaan memori pada autentikasi JWT

Gambar 3 menampilkan perbandingan penggunaan memori heap antara kedua skenario dalam kondisi beban steady-state. Diagram menunjukkan bahwa Skenario 1 mengonsumsi 28,4 MB memori dengan tren yang cenderung meningkat seiring waktu, mencerminkan akumulasi objek-objek yang menunggu dibersihkan oleh garbage collector. Sebaliknya, Skenario 2 hanya menggunakan 12,7 MB dengan pola yang jauh lebih stabil, menunjukkan bahwa instance tunggal Singleton mampu melayani seluruh permintaan tanpa perlu alokasi memori tambahan per-request.

Hal tersebut menunjukkan penggunaan memori di Skenario 1 adalah 28,4 MB, meningkat seiring waktu karena instansiasi ulang objek yang menyebabkan fragmentasi memori tinggi. Setelah penerapan Singleton Pattern (Gambar 4), penggunaan memori turun menjadi 12,7 MB. Pengurangan mencapai 55,3% karena hanya satu instance menangani pengelolaan token dan sesi, mengurangi overhead memori.



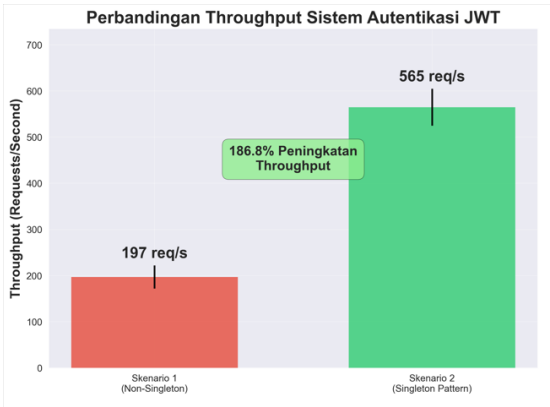
Gambar 4. Alokasi memori pada sistem autentikasi JWT

Gambar 4 menggambarkan pola alokasi memori secara detail selama pengujian berlangsung. Pada Skenario 1, kurva alokasi memori menunjukkan pola sawtooth (gigi gergaji) yang khas, di mana memori naik tajam akibat instansiasi objek baru setiap permintaan, kemudian turun saat garbage collector membersihkannya. Siklus allocate-collect yang terus berulang ini mengonsumsi sumber daya CPU tambahan. Pada Skenario 2, kurva memori jauh lebih datar dan stabil karena tidak ada pembuatan objek JWTValidator atau SessionManager baru setelah inisialisasi pertama, sehingga GC nyaris tidak terpicu selama pengujian.

Pengurangan 55,3% merupakan konsekuensi langsung dari eliminasi duplikasi objek di memori heap. Pada Skenario 1, setiap instance JWTValidator dan SessionManager membawa salinan konfigurasi, kunci kriptografis, dan buffer internal masing-masing. Dengan throughput 197 req/s, ratusan objek redundan dikelola bersamaan oleh garbage collector, meningkatkan latensi operasi GC secara signifikan [11].

Analisis alokasi memori (Gambar 4) menunjukkan pola fundamental berbeda antara kedua skenario. Pada Skenario 1, grafik memori menunjukkan pola sawtooth khas dari siklus allocate-then-collect yang intensif. Pada Skenario 2, pola memori jauh lebih stabil dan linier, mengindikasikan berkurangnya tekanan GC sehingga runtime dapat mengalokasikan lebih banyak sumber daya untuk pemrosesan logis [9].

4.3. Throughput



Gambar 5. Diagram perbandingan throughput pada autentikasi JWT

Gambar 5 menyajikan perbandingan throughput sistem dalam satuan request per

detik (req/s). Diagram batang menunjukkan bahwa Skenario 2 dengan Singleton Pattern mampu memproses 565 req/s, hampir tiga kali lipat dibandingkan Skenario 1 yang hanya mencapai 197 req/s. Peningkatan throughput sebesar 186,8% ini mencerminkan bahwa dengan menghilangkan bottleneck instansiasi objek, sistem dapat melayani jauh lebih banyak permintaan autentikasi dalam satuan waktu yang sama, meningkatkan kapasitas layanan secara signifikan tanpa penambahan infrastruktur hardware.

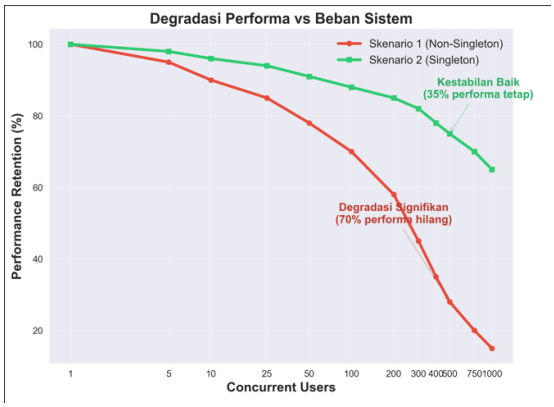
Dari data diatas menunjukkan throughput pada Skenario 1 adalah 197 req/s, terbatas karena overhead tinggi akibat instansiasi objek berulang. Pada Skenario 2 dengan Singleton Pattern, throughput meningkat menjadi 565 req/s, peningkatan sebesar 186,8%. Singleton Pattern mengurangi overhead instansiasi, mengelola sumber daya lebih efisien, dan menghilangkan bottleneck.

Peningkatan 186,8% mencerminkan kapasitas pemrosesan sistem yang meningkat drastis. Dengan Singleton Pattern, thread-thread tidak perlu menunggu instansiasi selesai sebelum mengakses JWT Validator atau Session Manager. Akses langsung ke instance yang sudah siap pakai mengurangi latensi per-operasi secara signifikan dan meningkatkan concurrency efektif sistem secara keseluruhan.

Implikasi ekonomis dari peningkatan throughput ini sangat signifikan. Dengan 565 req/s, satu instance server dapat menggantikan hampir tiga instance Skenario 1 untuk beban setara, menghasilkan penghematan biaya infrastruktur yang substansial. Temuan ini konsisten dengan Hasim et al. [6] yang mendemonstrasikan efisiensi Singleton Pattern dalam konteks microframework berbasis pola desain serupa.

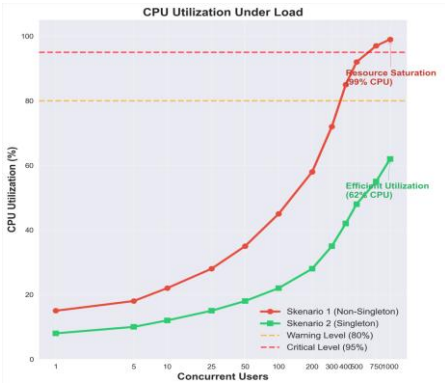
4.4. Skalabilitas

Pada Skenario 1, aplikasi tanpa Singleton Pattern menunjukkan penurunan performa signifikan seiring bertambahnya concurrent users.



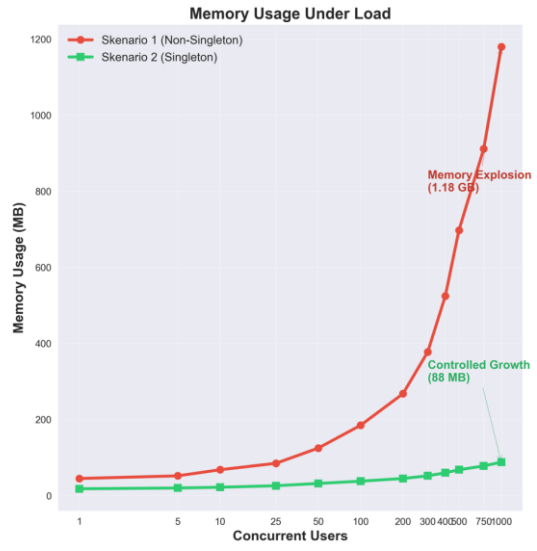
Gambar 6. Degradasi performa vs beban sistem

Gambar 6 menunjukkan grafik garis yang menggambarkan degradasi performa (dalam persentase retensi performa) seiring meningkatnya jumlah concurrent users. Kurva Skenario 1 mengalami penurunan yang tajam dan tidak linier, terutama ketika jumlah pengguna melampaui 100, mencerminkan sistem yang mulai kewalahan menangani instansiasi objek secara masif. Kurva Skenario 2 menurun secara lebih gradual dan terkontrol, mempertahankan retensi performa 65% bahkan pada 1000 concurrent users, membuktikan bahwa Singleton Pattern secara fundamental mengubah karakteristik skalabilitas sistem autentikasi.



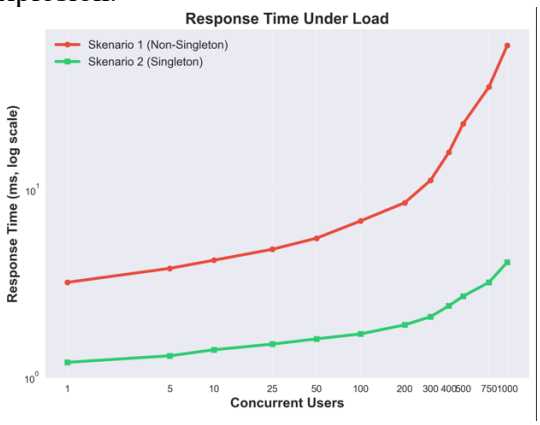
Gambar 7. Penggunaan CPU di bawah beban

Gambar 7 memvisualisasikan utilisasi CPU (%) pada berbagai tingkat beban concurrent users. Kurva Skenario 1 meningkat dengan cepat dan mencapai saturasi penuh pada 99% ketika beban mencapai 1000 pengguna, menandakan CPU sepenuhnya terbebani oleh kombinasi pemrosesan permintaan dan manajemen garbage collection yang intensif. Sebaliknya, Skenario 2 menunjukkan utilisasi CPU yang lebih rendah dan terkendali di sekitar 62% pada beban yang sama, mengindikasikan masih tersedianya kapasitas komputasi yang memadai untuk menangani lonjakan trafik tambahan tanpa risiko system overload.



Gambar 8. Penggunaan memori di bawah beban

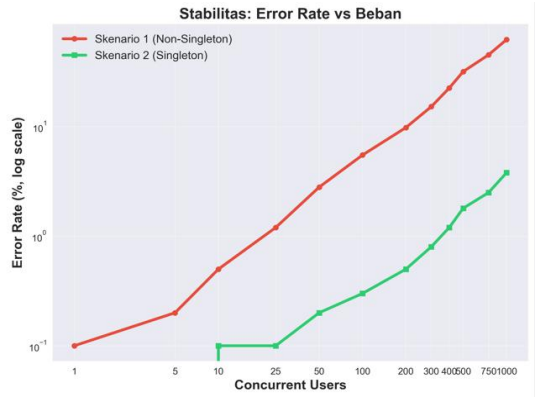
Gambar 8 memperlihatkan pertumbuhan konsumsi memori sistem seiring meningkatnya jumlah concurrent users. Grafik Skenario 1 menunjukkan pertumbuhan eksponensial yang ekstrem, mencapai 1,18 GB pada 1000 concurrent users akibat ribuan objek JWTValidator dan SessionManager yang diinstansiasi secara bersamaan memenuhi heap memori. Kondisi ini sangat berbahaya karena dapat memicu OutOfMemoryError pada sistem produksi. Skenario 2 menunjukkan pertumbuhan yang hampir linier dan terkendali, hanya mencapai 88 MB pada beban yang sama, menegaskan efektivitas Singleton Pattern dalam mencegah memory explosion.



Gambar 9. Waktu respons di bawah beban

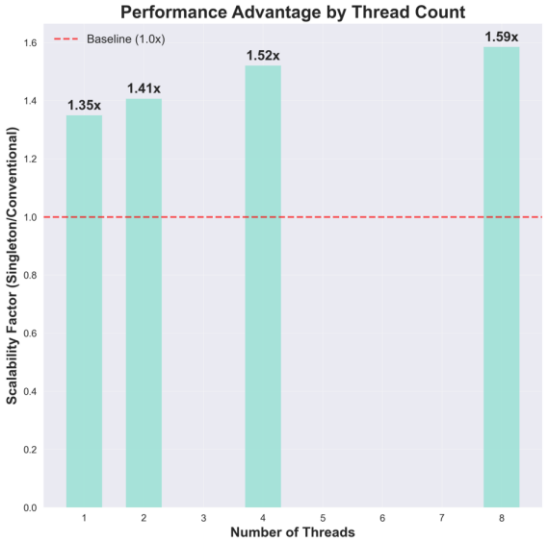
Gambar 9 menggambarkan perubahan waktu respon sistem (ms) ketika beban concurrent users meningkat secara bertahap. Pada Skenario 1, waktu respon meningkat secara dramatis dan non-linier: mulai dari nilai baseline yang sudah lebih tinggi, kemudian melonjak signifikan saat CPU mendekati saturasi karena thread-thread saling berkompetisi untuk mengalokasikan memori dan menjalankan GC. Pada Skenario 2, waktu respon bertambah secara lebih gradual dan proporsional terhadap peningkatan beban, karena setiap thread langsung mengakses instance yang sudah tersedia tanpa overhead instansiasi.

Saat beban mencapai 1000 pengguna, retensi performa turun hingga 15%, CPU mencapai titik jenuh 99%, dan terjadi memory explosion dengan konsumsi 1,18 GB. Pada Skenario 2 dengan Singleton Pattern, retensi performa mencapai 65% pada 1000 concurrent users. CPU tetap efisien di sekitar 62% dan pertumbuhan memori terkendali pada 88 MB.



Gambar 10. Error rate vs beban

Gambar 10 menampilkan tingkat error (error rate %) pada berbagai tingkat concurrent users. Pada Skenario 1, error rate meningkat tajam seiring bertambahnya beban, terutama karena request timeout yang dipicu oleh waktu respon yang sangat tinggi dan out-of-memory errors ketika heap penuh. Sebaliknya, Skenario 2 mempertahankan error rate yang sangat rendah bahkan pada 1000 concurrent users, mengindikasikan bahwa sistem dengan Singleton Pattern mampu menangani seluruh permintaan secara andal tanpa kegagalan signifikan, menjadikannya pilihan yang jauh lebih handal untuk lingkungan produksi dengan trafik tinggi.



Gambar 11. Keunggulan kinerja berdasarkan jumlah thread

Gambar 11 memvisualisasikan scalability factor, yaitu rasio throughput Skenario 2 terhadap Skenario 1, pada berbagai konfigurasi jumlah thread. Grafik menunjukkan bahwa keunggulan Singleton Pattern semakin menonjol seiring meningkatnya jumlah thread, dengan scalability factor mencapai 1,59x. Pola linear dari grafik ini mengindikasikan bahwa Singleton Pattern tidak hanya memberikan keunggulan performa absolut, tetapi juga meningkatkan efisiensi relatif secara proporsional terhadap pemanfaatan paralelisme CPU — properti yang sangat penting bagi sistem autentikasi modern yang beroperasi pada infrastruktur multi-core dan multi-thread.

Error rate tetap rendah meski beban meningkat drastis, mengindikasikan arsitektur ini mampu menangani skalabilitas horizontal tanpa degradasi signifikan. Analisis dengan variasi thread menunjukkan peningkatan linear throughput untuk Singleton Pattern, dengan scalability factor 1,59x dibanding pendekatan konvensional.

Analisis skalabilitas mengungkap perbedaan paling dramatis antar skenario. Pada Skenario 1, degradasi performa bersifat non-linear dan tajam saat concurrent users melampaui 100. Pada 500 users, CPU mencapai 85% utilisasi. Pada 1000 users, sistem memasuki kondisi kritis dengan CPU 99%, konsumsi memori meledak hingga 1,18 GB, dan error rate meningkat drastis akibat request timeout serta out-of-memory condition (Gambar 10).

Sebaliknya, Skenario 2 menunjukkan skalabilitas jauh lebih baik dengan degradasi yang gradual dan terkontrol (Gambar 6). Pada 1000 concurrent users, CPU hanya mencapai 62%, memberikan headroom untuk menangani burst traffic. Pertumbuhan memori terkendali pada 88 MB dibanding 1,18 GB pada Skenario 1, membuktikan Singleton Pattern mencegah memory explosion yang merupakan

ancaman utama ketersediaan layanan pada sistem autentikasi skala besar [9].

Scalability factor 1,59x (Gambar 11) menunjukkan Singleton Pattern tidak hanya meningkatkan performa pada beban saat ini, tetapi juga mempertahankan efisiensi seiring peningkatan beban. Penelitian Prasetya dan Suharjo [12] mengkonfirmasi bahwa optimasi pada lapisan manajemen token memberikan keuntungan performa lebih besar dibanding optimasi lapisan enkripsi semata, menegaskan pentingnya pemilihan pola arsitektur yang tepat dalam pengembangan sistem autentikasi yang skalabel dan efisien.

Hasil pengujian komprehensif pada kedua skenario dirangkum dalam Tabel 2 yang menyajikan perbandingan seluruh metrik performa secara berdampingan.

Metrik Performa	Skenario 1 (Tanpa SP)	Skenario 2 (Dengan SP)	Peningkatan
Waktu Respon Rata-rata	5,06 ms	1,77 ms	-65,0%
Penggunaan Memori (steady)	28,4 MB	12,7 MB	-55,3%
Throughput (req/s)	197 req/s	565 req/s	+186,8%
Retensi Performa @1000 users	15%	65%	+50 poin
CPU @1000 users	99% (jenuh)	62% (efisien)	-37 poin
Memori @1000 users	1.180 MB	88 MB	-92,5%
Scalability Factor	1,00x (baseline)	1,59x	+59%

Tabel 2. Perbandingan Metrik Performa Skenario 1 vs Skenario 2

5. KESIMPULAN

Penelitian ini menunjukkan bahwa penerapan Singleton Pattern dalam sistem autentikasi berbasis JWT memberikan peningkatan signifikan: (1) Pengurangan waktu respon 65%, dari 5,06 ms menjadi 1,77 ms; (2) Pengurangan penggunaan memori 55,3%, dari 28,4 MB menjadi 12,7 MB; (3) Peningkatan throughput 186,8%, dari 197 req/s menjadi 565 req/s; (4) Retensi performa 65% pada 1000 concurrent users.

Singleton Pattern terbukti efektif dalam mengelola sumber daya secara lebih efisien, mengurangi overhead, dan meningkatkan skalabilitas. Penelitian selanjutnya disarankan mengeksplorasi penerapannya dalam konteks autentikasi lain seperti sistem multi-faktor atau OAuth, serta perbandingan dengan pola desain lain dalam pengelolaan sesi dan token.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Universitas Singaperbangsa Karawang dan semua pihak yang telah memberikan dukungan dan kontribusi dalam pelaksanaan penelitian ini.

DAFTAR PUSTAKA

[1] C. Gemawaty and Y. Yuliani, "MANAJEMEN IDENTITAS DAN AKSES DALAM KEAMANAN SISTEM INFORMASI," J. Manajamen Inform. Jayakarta, vol. 4, no. 4, pp. 396-403, 2024, doi: 10.52362/jmijayakarta.v4i4.1527.

[2] A. Nazir, S. Sagheer, H. Naaz, Z. I. Ansari, M. H. Khan, and M. S. Umar, "Grid Based Pair Authentication with Improved Security," in 2024 4th Intl. Conf. on Technological Advancements in Computational Sciences (ICTACS), 2024, pp. 1636-1641. doi: 10.1109/ICTACS62700.2024.10840482.

[3] S. Wang, Q. Zhang, L. Hu, X. Zhang, Y. Wang, and C. Aggarwal, "Sequential/Session-based Recommendations: Challenges, Approaches, Applications and

Opportunities," in SIGIR 2022, pp. 3425-3428. doi: 10.1145/3477495.3532685.

[4] I. Shikhverdiyev, E. Babayev, C. Rahimli, N. Rahimli, and H. Aslanova, "Secure authentication in e-government 2.0: a comparative analysis of traditional session-based and modern jwt-based authentication," Int. Sci. J. Eng. Agric., vol. 3, no. 6, pp. 117-129, Dec. 2024, doi: 10.46299/j.isjea.20240306.12.

[5] R. Raymond and S. M. A. Savarimuthu, "Software Design Patterns and Architecture Patterns - A Study Explored," in 2022 5th Intl. Conf. on Contemporary Computing and Informatics (IC3I), 2022, pp. 1998-2006. doi: 10.1109/IC3I56241.2022.10073279.

[6] J. A. N. Hasim, U. Sa'adah, D. I. P. Sari, F. A. Damastuti, and F. N. Koirudin, "Developing Microframework based on Singleton and Abstract Factory Design Pattern," in 2022 Intl. Electronics Symposium (IES), 2022, pp. 676-683. doi: 10.1109/IES55876.2022.9888548.

[7] R. Sharma and R. Arora, "Access Control," in Low-Code Development with Appsmith, Berkeley, CA: Apress, 2023, pp. 179-196. doi: 10.1007/978-1-4842-9813-8_6.

[8] A. Tijms, T. Bais, and W. Keil, "MicroProfile JWT," in The Definitive Guide to Security in Jakarta EE, Berkeley, CA: Apress, 2022, pp. 475-519. doi: 10.1007/978-1-4842-7945-8_8.

[9] H. Alnuhait, W. Alzyadat, A. Althunibat, H. Kahtan, B. Zaqaibeh, and H. A. Al-Khawaja, "Web application performance assessment," International Journal of Advanced and Applied Sciences, vol. 11, no. 9, pp. 214-226, Sep. 2024, doi: 10.21833/ijaas.2024.09.023.

[10] N. Mitikov and N. Guk, "Investigation of Software Application Performance Issues," Matematichne ta Komp. Modelyuvannya, vol. 25, pp. 22-36, Sep. 2024, doi: 10.32626/2308-5916.2024-25.22-36.

[11] Z. K. Pathan, N. D. Singh, K. R. Sharma, and H. C. Vachheta, "Comprehensive Analysis of Cloud Computing Performance Factors," IJARSCT, pp. 344-353, Nov. 2024, doi: 10.48175/ijarsct-22164.

[12] A. B. Prasetya and I. Suharjo, "BACKEND API DATA PROTECTION MENGGUNAKAN JWT TOKEN DAN ALGORITMA AES 256-BIT," JITET, vol. 13, no. 1, 2025, doi: 10.23960/jitet.v13i1.5699.

[13] A. P. Rahman and N. Erzed, "SISTEM PENGESAHAN DOKUMEN DENGAN QR CODE DAN JWT," J. Mhs. Tek. Inform., vol. 8, no. 6, pp. 12143-12150, 2024.

[14] Akanksha and A. Chaturvedi, "Comparison of Different Authentication Techniques and Steps to Implement Robust JWT Authentication," in 2022 7th Intl. Conf. on Communication and Electronics Systems (ICCES), 2022, pp. 772-779. doi: 10.1109/ICCES54183.2022.9835796.

[15] S. V. Akram, S. K. Joshi, and R. Deorari, "Web Application Based Authentication System," in 2022 Intl. Interdisciplinary Humanitarian Conf. for Sustainability (IIHC), 2022, pp. 1439-1443. doi: 10.1109/IIHC55949.2022.10059984.