

PENGUJIAN PERFORMA DAN RATE LIMITING SISTEM INFORMASI AKADEMIK MENGGUNAKAN ALAT PENGUJIAN K6: STUDI KASUS POLITEKNIK XYZ

Alif Shofa Danutirta^{*1}, Mitsal Shafiq Sulasno², Dwi Anto³, Miftahunnisa' Igriza⁴

¹Program Studi Administrasi Hukum Umum, Politeknik Pengayoman Indonesia, Depok, Indonesia

²Program Studi Hukum Kekayaan Intelektual, Politeknik Pengayoman Indonesia, Depok, Indonesia

³Program Studi Sistem Informasi, Universitas Terbuka, Bandung, Indonesia

⁴Program Studi Ilmu Perpustakaan dan Sains Informasi, Universitas Terbuka, Bandung, Indonesia

Penulis korespondensi: Alif Shofa Danutirta (alifshofa.danutirta@gmail.com)

Keandalan Sistem Informasi Akademik (SIKAD) dalam menangani beban akses pengguna perlu dijamin melalui pengujian performa. Penelitian ini menganalisis response time, throughput, dan error rate pada website SIKAD Politeknik XYZ menggunakan k6, sekaligus mengidentifikasi mekanisme rate limiting yang memengaruhi validitas pengujian. Metode dilaksanakan melalui empat tahap: perencanaan lingkungan, perencanaan skenario, pelaksanaan, serta interpretasi dan analisis data. Kalibrasi bertahap diterapkan menemukan Rate Aman Maksimum (RAM) sebelum melaksanakan empat skenario beban baku (Smoke, Load, Stress, Break-point Test) proporsional terhadap RAM. Hasil menunjukkan RAM sistem 2 request/detik dengan error rate 0% dan response time median di bawah 100 milidetik. Penolakan mulai konsisten pada kisaran 3,0–3,8 request/detik. Respons seragam mengindikasikan penolakan berasal dari lapisan Web Application Firewall (WAF)/reverse proxy, bukan aplikasi backend. Kalibrasi berbasis laju permintaan lebih representatif dibandingkan skema virtual user konvensional dalam menghasilkan data pengujian valid dan dapat direplikasi.

Kata Kunci — Indobert, Simcse, Kemiripan Semantik, Cosine Similarity, Tugas Akhir.

I. PENDAHULUAN

Perkembangan teknologi informasi telah mendorong institusi pendidikan tinggi untuk mengadopsi sistem informasi berbasis web untuk mendukung layanan akademik maupun administratif secara digital. Kualitas layanan informasi akademik berbasis *website* tidak hanya dinilai dari kelengkapan fitur, tetapi juga dari sejauh mana sistem tersebut mampu mendukung kebutuhan pengguna secara efektif dan memuaskan [1]. Situs web akademik, termasuk perpustakaan digital dan portal informasi kampus, terbukti berperan penting dalam mendukung

aktivitas keilmuan mahasiswa maupun tenaga pengajar, sehingga keandalan dan interaktivitasnya menjadi aspek yang tidak dapat diabaikan [2]. Keandalan sistem informasi akademik menjadi semakin penting karena gangguan performa pada periode akses tinggi dapat menghambat proses akademik mahasiswa, dosen, dan administrator [3], hingga arsitektur pengelolaan data industri berskala besar [4], yang keduanya menegaskan bahwa keandalan dan ketahanan sistem informasi merupakan kebutuhan lintas domain, tidak terkecuali pada sistem informasi akademik.

Salah satu aspek utama dalam menjamin keandalan suatu sistem informasi berbasis web adalah kemampuannya dalam menangani beban akses pengguna, khususnya pada periode-periode puncak seperti masa pengisian Kartu Rencana Studi (KRS) atau publikasi hasil ujian. Untuk mengevaluasi aspek tersebut, pengujian performa (*performance testing*) menjadi metode yang lazim digunakan, mencakup berbagai jenis pengujian seperti pengujian beban (*load testing*), pengujian tekanan (*stress testing*), hingga pengujian skalabilitas [5].

Kajian literatur yang komprehensif menunjukkan bahwa metodologi pengujian performa terus berkembang seiring meningkatnya kompleksitas arsitektur perangkat lunak modern, sehingga pemilihan pendekatan dan praktik terbaik menjadi krusial untuk menjamin kualitas perangkat lunak secara keseluruhan [6].

Berbagai penelitian terdahulu telah menerapkan pengujian performa pada beragam jenis platform web. Pada sektor *e-commerce*, pengujian performa dilakukan untuk mengukur ketahanan *website* terhadap lonjakan akses pengguna, sebagaimana ditunjukkan pada studi kasus platform Shopee selama periode pandemi Covid-19 [7]. Pada konteks yang lain, pengujian performa turut digunakan untuk mengoptimalkan waktu muat (*load time*) suatu *website* komersial, yang menunjukkan bahwa hasil

pengujian dapat langsung dimanfaatkan sebagai dasar perbaikan teknis [8]. Pendekatan serupa juga diterapkan pada institusi pendidikan, di mana pengujian beban dan tekanan digunakan untuk mengevaluasi performa *website* resmi perguruan tinggi menggunakan perangkat *LoadImpact* [9], maupun untuk menjamin kualitas (*quality assurance*) *website* akademik secara lebih menyeluruh menggunakan berbagai perangkat pengujian performa [10].

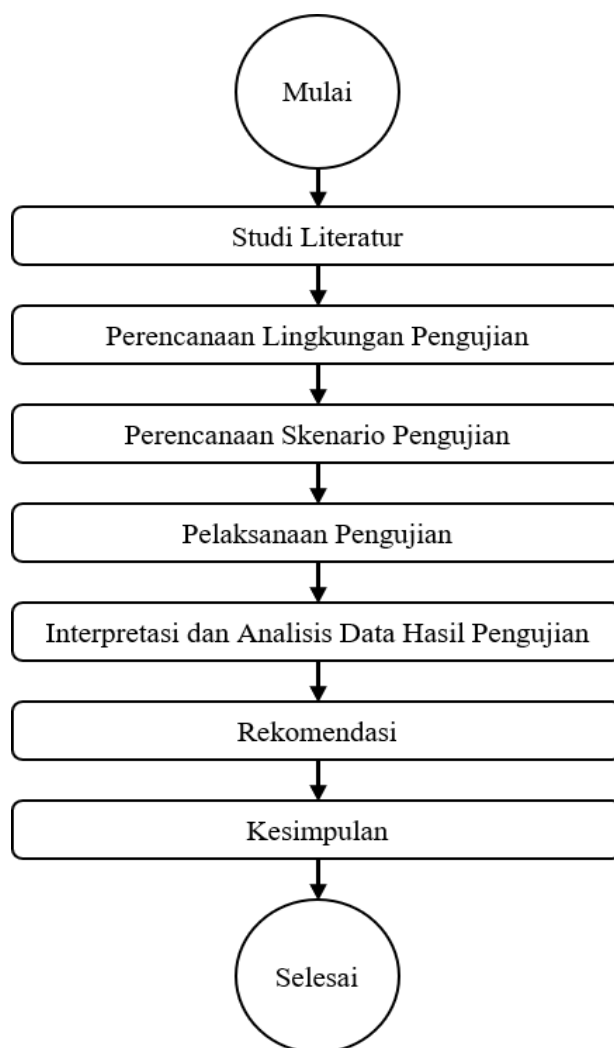
Fokus pengujian performa pada sistem informasi akademik secara khusus juga telah menjadi perhatian sejumlah peneliti. Pengujian beban terhadap *website* program studi menggunakan Apache JMeter dilakukan untuk mengevaluasi kesiapan sistem yang belum pernah diuji ketahanannya sejak pertama kali dibangun [11]. Studi lain membandingkan performa sistem informasi akademik yang dikembangkan menggunakan kerangka kerja (*framework*) PHP berbeda, yaitu *CodeIgniter* dan *Laravel*, melalui pengujian beban dan tekanan untuk menentukan kerangka kerja yang lebih unggul secara performa [12]. Untuk mendukung efisiensi proses pengujian tekanan, pendekatan berbasis model (*model-driven*) turut dikembangkan guna mengotomatisasi pembuatan skrip pengujian pada aplikasi web [13].

Dari sisi perangkat dan kerangka kerja pengujian, berbagai upaya pengembangan turut dilakukan untuk mendukung proses pengujian performa yang lebih efisien dan akurat. Kerangka kerja *E2E-Loader* dikembangkan untuk mendukung pengujian performa aplikasi web secara menyeluruh dari sisi antarmuka pengguna [14], sementara perangkat analisis pengujian beban lain turut dirancang untuk mempermudah proses evaluasi performa aplikasi berbasis web [15]. Pemanfaatan perangkat sumber terbuka (*open source*), termasuk *k6* dan *Lighthouse*, juga terbukti efektif dalam mendukung proses optimasi aplikasi web modern berbasis JavaScript [16]. Dalam praktiknya, pemilihan perangkat pengujian beban yang tepat menjadi hal penting, sebagaimana ditunjukkan oleh studi perbandingan kinerja antara *Apache JMeter*, *Gatling*, dan *k6* yang mengungkap karakteristik serta keunggulan masing-masing perangkat pada konteks pengujian yang berbeda-beda [17]. Lebih lanjut, evaluasi skalabilitas dan performa pada arsitektur sistem web modern berbasis *microservices* turut dikaji melalui pendekatan *observability*, yang menunjukkan bahwa metode pengujian performa terus beradaptasi mengikuti perkembangan arsitektur sistem [18].

Berdasarkan studi-studi terdahulu, pengujian performa pada aplikasi web dan sistem informasi akademik umumnya berfokus pada pengukuran *response time*, *throughput*, dan *error rate* sebagai indikator kapasitas aplikasi. Namun, pengaruh lapisan infrastruktur, seperti *Web Application Firewall*, *reverse proxy*, atau mekanisme *rate limiting*, terhadap hasil pengujian belum banyak dibahas secara khusus. Padahal, mekanisme tersebut dapat menyebabkan sebagian permintaan gagal sebelum mencapai aplikasi, sehingga *error rate* yang terukur berpotensi tidak merepresentasikan keterbatasan kapasitas *website* secara langsung.

Kebaruan penelitian ini terletak pada integrasi identifikasi mekanisme *rate limiting* ke dalam analisis performa *website* Sistem Informasi Akademik berbasis cloud. Berbeda dari penelitian sebelumnya yang cenderung menginterpretasikan *error rate* sebagai indikator kegagalan aplikasi, penelitian ini membedakan kemungkinan kegagalan yang berasal dari aplikasi dan dari lapisan proteksi infrastruktur. Dengan demikian, penelitian ini bertujuan menganalisis *response time*, *throughput*, dan *error rate* serta menghasilkan interpretasi performa yang lebih representatif terhadap kondisi aktual sistem.

II. METODE PENELITIAN



Gambar 1 Tahapan Penelitian

Metodologi penelitian ini dirancang untuk memberikan gambaran terstruktur mengenai tahapan evaluasi performa situs web Sistem Informasi Akademik (SIKAD) di Politeknik XYZ. Sebagaimana ditunjukkan pada gambar 1, alur penelitian ini mengikuti kerangka kerja pengujian perangkat lunak yang sistematis, mulai dari tahap perencanaan hingga tahap analisis hasil. Berdasarkan pendekatan yang dikemukakan oleh Setiawan, penelitian ini mengadopsi kerangka pengujian performa sistematis yang dibagi menjadi empat fase utama: perencanaan lingkungan, perancangan skenario beban, eksekusi teknis

menggunakan k6, serta interpretasi data hasil pengujian [11]. Alur ini dirancang untuk memetakan perilaku server secara objektif saat menghadapi fluktuasi trafik pengguna [17].

1. Perencanaan Lingkungan Pengujian

Fase ini mencakup penyiapan infrastruktur pengujian, mulai dari spesifikasi perangkat keras hingga konfigurasi alat pengujian k6 dari Grafana Lab pada mesin lokal.

Operasional k6 dilakukan sepenuhnya melalui antarmuka terminal untuk memastikan efisiensi konsumsi memori dan stabilitas generator beban selama proses simulasi berlangsung [9], [17].

2. Perencanaan Skenario Pengujian

Pengujian beban (*load testing*) terhadap sistem Sistem Informasi Akademik (SIKAD) dilakukan menggunakan pendekatan kalibrasi bertahap (*step-ladder calibration*) dengan bantuan perangkat lunak k6 dari Grafana Labs.

Pendekatan ini dipilih karena percobaan awal menggunakan skema pengujian konvensional berbasis jumlah *virtual user* (VU) menunjukkan tingkat kegagalan permintaan (*request failure rate*) yang tinggi bahkan pada beban rendah, yang setelah ditelusuri disebabkan oleh mekanisme pembatasan kecepatan permintaan (*rate limiting*) pada lapisan *Web Application Firewall* (WAF) atau proxy di depan aplikasi, bukan karena keterbatasan kapasitas aplikasi itu sendiri. Untuk memperoleh data pengujian yang representatif dan valid, metodologi pengujian kemudian disesuaikan dengan mengontrol kecepatan permintaan secara presisi (*request per second*, rps) menggunakan *executor ramping-arrival-rate* dan *constant-arrival-rate* daripada menambah jumlah VU.

3. Pelaksanaan Pengujian

Proses pengujian dilaksanakan dalam dua tahap utama. Tahap pertama (kalibrasi) bertujuan menemukan Rate Aman Maksimum (RAM), yaitu kecepatan permintaan tertinggi yang masih dapat diproses sistem tanpa memicu penolakan (HTTP 429 – *Too Many Requests*). Kalibrasi dilakukan dengan menaikkan kecepatan permintaan secara bertahap dari 0,2 *request/detik* hingga 5 *request/detik*, dengan jeda antar tahap untuk memastikan hasil tidak bias oleh akumulasi permintaan sebelumnya.

Tahap kedua (pengujian utama) terdiri atas empat skenario baku pengujian performa (*Smoke Test*, *Load Test*, *Stress Test*, dan *Break-point Test*) yang masing-masing target kecepatan permintaannya ditentukan secara proporsional terhadap RAM yang telah ditemukan pada tahap kalibrasi, guna memastikan setiap skenario memiliki dasar kuantitatif yang jelas dan dapat dipertanggungjawabkan secara metodologis.

4. Interpretasi dan Analisis Data Hasil Pengujian

Setelah pengujian selesai, data tiap skenario diinterpretasikan dan dianalisis menggunakan empat pendekatan.

Pertama, analisis deskriptif-kuantitatif dengan tiga metrik utama: *response time* (rata-rata, median, persentil 90 & 95, maksimum), *throughput*, dan *error rate* untuk menggambarkan performa pada berbagai beban.

Kedua, analisis pola kegagalan permintaan untuk membedakan kegagalan akibat kapasitas aplikasi dengan kegagalan dari proteksi infrastruktur (*rate limiting* pada WAF), melalui pemeriksaan status HTTP (khususnya 429) dan isi respons.

Ketiga, triangulasi antar *Load*, *Stress*, dan *Break-point Test* untuk membandingkan titik awal kegagalan dan menguji konsistensi ambang batas sistem.

Keempat, sintesis hasil menjadi kesimpulan tentang kapasitas dan ambang aman SIKAD serta rekomendasi pengelolaan kapasitas dan konfigurasi pembatasan kecepatan.

III. HASIL DAN PEMBAHASAN

Temuan deskriptif kuantitatif ini menjadi rujukan utama dalam melakukan optimasi infrastruktur agar layanan akademik digital tetap handal dan responsif bagi pengguna.

Perencanaan Lingkungan Pengujian. Tahap perencanaan lingkungan pengujian bertujuan untuk mendefinisikan seluruh infrastruktur teknis yang digunakan selama proses evaluasi performa berlangsung. Dokumentasi spesifikasi perangkat keras dan lunak sangat krusial untuk memastikan bahwa hasil pengujian bersifat akurat dan dapat direplikasi oleh peneliti yang lain. Selain itu, pemilihan perangkat dengan performa tinggi diperlukan untuk meminimalkan risiko kegagalan generator beban (*producer crash*) saat melakukan simulasi beban tinggi atau *stress testing*.

Dalam penelitian ini, pengujian dilakukan menggunakan mesin lokal (*workstation*) dengan spesifikasi seperti pada tabel 1 berikut:

Tabel 1. Spesifikasi Lingkungan Pengujian

Komponen	Spesifikasi Teknis	Relevansi dan Dasar Penentuan
Model Perangkat	Laptop X (System Model 83DA)	Penggunaan mesin lokal (<i>workstation</i>) yang stabil meminimalkan bias eksternal selama simulasi beban.
Prosesor (CPU)	Intel (R) Core (TM) Ultra 7 155H (22 CPUs), ~1.4GHz	Jumlah core yang besar (22 CPUs) sangat mendukung koneksi independen yang konkuren pada alat k6 secara efisien.
Memori (RAM)	16.384 MB (16 GB)	Memori yang besar memastikan generator beban tidak mengalami bottleneck internal saat mensimulasikan ribuan pengguna virtual.
Sistem Operasi	Windows 11 Home Single Language 64-bit	Arsitektur 64-bit memungkinkan pemanfaatan sumber daya sistem secara maksimal untuk eksekusi skrip pengujian beban.
Alat Pengujian	k6 oleh Grafana Lab	Dipilih karena sifatnya yang ringan (<i>lightweight</i>) dan efisiensi generator bebannya yang presisi untuk beban tinggi.
Metode Operasi	Terminal (Command Line Interface)	Operasi melalui terminal (Powershell) untuk menjaga efisiensi sumber daya CPU dan memori pada <i>workstation</i> penguji.
Bahasa Skrip	JavaScript	Basis utama skrip k6 yang memberikan fleksibilitas tinggi dalam merancang skenario transaksi pengguna yang

Komponen	Spesifikasi Teknis	Relevansi dan Dasar Penentuan
Grafis	DirectX 12	kompleks. Mendukung akselerasi performa sistem secara keseluruhan selama proses monitoring pengujian.

Perencanaan Skenario Pengujian

Perencanaan pengujian pada awalnya disusun menggunakan skema klasik berbasis jumlah *virtual user* (VU), namun kemudian diubah menjadi skema berbasis kecepatan permintaan (*request per second*, rps) setelah tahap kalibrasi menunjukkan bahwa parameter VU tidak merepresentasikan beban aktual secara akurat akibat adanya mekanisme *rate limiting* pada lapisan WAF/reverse proxy.

Pada tabel 2, disajikan rencana kalibrasi dan rencana skenario pengujian beban bertahap untuk situs web

Tabel 3. Rencana Skenario Pengujian

Skenario	Tujuan Pengujian	Parameter Rancangan Awal	Parameter Rancangan Final	Durasi Ramp-Up	Durasi Hold
1. Smoke Test	Memvalidasi sistem berjalan normal pada beban minimal	50 VU	2 request/detik (1× RAM)	1 menit	1 menit
2. Load Test	Menguji performa pada beban traffic normal-tinggi	500 VU	3 request/detik (1,5× RAM)	3 menit	3 menit
3. Stress Test	Menguji respons sistem saat beban melebihi kapasitas amannya	1000 VU	8 request/detik (4× RAM)	5 menit	5 menit
4. Break-point Test	Menentukan titik jenuh (breaking point) sistem secara presisi	1500 VU	6 request/detik (bertahap, naik dari RAM)	7 menit	5 menit

Perubahan dari parameter berbasis VU ke parameter berbasis rps dilakukan karena satu VU tidak selalu menghasilkan satu permintaan per detik secara konsisten jumlah permintaan aktual yang dikirim sangat bergantung pada waktu respon server dan jeda antar iterasi yang

SIAKAD Politeknik XYZ yang diimplementasikan menggunakan fitur *stages* pada script k6:

Tabel 2. Rencana Kalibrasi

Step	Rate
A1	1 request/10 detik (0.1 rps)
A2	1 request/5 detik (0.2 rps)
A3	1 request/2 detik (0.5 rps)
A4	1 request/detik (1 rps)
A5	2 request/detik (2 rps)
A6	5 request/detik (5 rps)

Rencana kalibrasi dilakukan melalui 6 tahapan untuk mengetahui rate optimal yang bisa ditetapkan dalam untuk skenario pengujian. Pada tabel 3, ditunjukkan rencana skenario pengujian dan perbandingan antara parameter rancangan awal dan parameter rancangan final yang digunakan dalam pelaksanaan pengujian.

diterapkan pada skrip pengujian. Parameter berbasis rps dipilih agar kecepatan permintaan ke server dapat dikendalikan secara presisi dan hasil pengujian dapat direplikasi secara konsisten.

```

Administrator: Windows PowerShell
PS C:\WINDOWS\system32> Set-ExecutionPolicy Bypass -Scope Process -Force
>> [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocol -bor 3072
>> iex ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1')) Forcing web requests to allow TLS v1.2 (
Required for requests to Chocolatey.org) Getting latest version of the Chocolatey package for download.
Not using proxy.
Getting Chocolatey from https://community.chocolatey.org/api/v2/package/chocolatey/2.7.4. Downloading https://community.chocolatey
.org/api/v2/package/chocolatey/2.7.4 to C:\Users\alifs\AppData\Local\Temp\chocolatey\chocoInstall\chocolatey.zip
Not using proxy.
Extracting C:\Users\alifs\AppData\Local\Temp\chocolatey\chocoInstall\chocolatey.zip to C:\Users\alifs\AppData\Local\Temp\chocolatey\chocoInstall
Installing Chocolatey on the local machine
Creating ChocolateyInstall as an environment variable (targeting 'Machine')
Setting ChocolateyInstall to 'C:\ProgramData\chocolatey'
WARNING: It's very likely you will need to close and reopen your shell
before you can use choco.
Restricting write permissions to Administrators
We are setting up the Chocolatey package repository.
The packages themselves go to 'C:\ProgramData\chocolatey\lib'
(i.e. C:\ProgramData\chocolatey\lib\yourPackageName).
A shim file for the command line goes to 'C:\ProgramData\chocolatey\bin'
and points to an executable in 'C:\ProgramData\chocolatey\lib\yourPackageName'.

Creating Chocolatey CLI folders if they do not already exist.

chocolatey.nupkg file not installed in lib.
Attempting to locate it from bootstrapper.
PATH environment variable does not have C:\ProgramData\chocolatey\bin in it. Adding...
WARNING: Not setting tab completion: Profile file does not exist at
'C:\Users\alifs\OneDrive\Documents\WindowsPowerShell\Microsoft.PowerShell_profile.ps1'.
Chocolatey CLI (choco.exe) is now ready.
You can call choco from anywhere, command line or PowerShell by typing choco.
Run choco ?? for a list of functions.
You may need to shut down and restart PowerShell and/or consoles
first prior to using choco.
Ensuring Chocolatey commands are on the path
Ensuring chocolatey.nupkg is in the lib folder
PS C:\WINDOWS\system32> choco -v
2.7.4
PS C:\WINDOWS\system32>

```

Gambar 2. Instalasi Package Manag

Pelaksanaan Pengujian

Pelaksanaan pengujian dilaksanakan menjadi 3 sub tahapan, diantaranya adalah Persiapan Pengujian, Pengujian Tahap 1 (Kalibrasi), dan Pengujian Tahap 2 (Skenario).

1. Persiapan Pengujian

Terdapat 2 persiapan yang perlu dilakukan saat akan melakukan pengujian, diantaranya adalah instalasi *package manager* chocolatey dan instalasi alat pengujian k6 dari Grafana Labs.

a. Instalasi *package manager*

Pada pengujian ini, peneliti menggunakan chocolatey sebagai *package manager* untuk mempersiapkan tools k6 yang dibuat oleh Grafana Labs. Gambar 2 menunjukan proses instalasi dari *package manager* chocolatey sebagai langkah persiapan untuk melakukan instalasi alat pengujian “k6”.

Pemilihan Chocolatey bertujuan untuk memastikan konfigurasi lingkungan yang konsisten dan *reproducible* guna meminimalkan bias eksternal agar hasil metrik *response time* serta *throughput* tetap akurat [4]. Selain itu, dukungan antarmuka terminal (CLI) pada alat ini selaras dengan model operasional k6, sehingga meningkatkan efisiensi persiapan alat serta mendukung integrasi otomatisasi yang lebih baik dalam siklus pengujian performa [16].

b. Instalasi alat pengujian “k6”

Alat pengujian yang digunakan adalah k6 dari Grafana Labs. Gambar 3 menunjukkan proses instalasi alat pengujian “k6” yang dilakukan pada *command line interface* (CLI). Pemilihan k6 didasarkan pada keunggulan fitur dan efisiensi generator beban yang presisi untuk simulasi trafik tinggi [17].

Melalui pendekatan skrip *JavaScript*, k6 menawarkan fleksibilitas otomatisasi serta kemampuan menjalankan berbagai skenario pengujian seperti load dan stress testing secara komprehensif. Selain itu, k6 sangat efektif dalam mensimulasikan beban trafik yang realistis untuk memvalidasi ketahanan dan performa sistem di bawah tekanan [18].

```

Administrator: Windows PowerShell
PS C:\WINDOWS\system32> choco install k6
Chocolatey v2.7.3
Installing the following packages:
k6
By installing, you accept licenses for the packages.
Downloading package from source 'https://community.chocolatey.org/api/v2/'
Progress: Downloading k6 2.1.0... 100%

k6 v2.1.0 [Approved]
k6 package files install completed. Performing other installation steps.
The package k6 wants to run 'chocolateyInstall.ps1'.
Note: If you don't run this script, the installation will fail.
Note: To confirm automatically next time, use '-y' or consider:
choco feature enable -n allowGlobalConfirmation
Do you want to run the script?([Y]es/[A]ll scripts/[N]o/[P]rint): Y

Extracting 64-bit C:\ProgramData\chocolatey\lib\k6\tools\k6-v2.1.0-windows-amd64.zip to C:\ProgramData\chocolatey\lib\k6\tools...
C:\ProgramData\chocolatey\lib\k6\tools
ShimGen has successfully created a shim for k6.exe
The install of k6 was successful.
Deployed to 'C:\ProgramData\chocolatey\lib\k6\tools'

Chocolatey installed 1/1 packages.
See the log for details (C:\ProgramData\chocolatey\logs\chocolatey.log).
PS C:\WINDOWS\system32> Choco --version
2.7.3
PS C:\WINDOWS\system32> k6 --version
k6.exe v2.1.0 (commit/83a87a41e2, go1.26.4, windows/amd64)
PS C:\WINDOWS\system32>

```

Gambar 3 Instalasi Alat Pengujian “k6”

2. Pengujian Tahap 1 (Kalibrasi)

Tabel 4. Hasil Tahap 1 (Kalibrasi)

Langkah	Target kecepatan Permintaan	Durasi (menit)	Hasil
A1	0,1 req/detik (1 req/10 detik)	2	Berhasil 100% permintaan sukses
A2	0,2 req/detik (1 req/5 detik)	2	Berhasil 100% permintaan sukses
A3	0,5 req/detik (1 req/2 detik)	2	Berhasil 100% permintaan sukses
A4	1 req/detik	2	Berhasil 100% permintaan sukses
A5	2 req/detik	2	Berhasil 100% permintaan sukses (RAM)
A6	5 req/detik	2	Gagal, Error (429) mulai muncul pada permintaan ke-47

Pada tabel 4 ditunjukkan hasil tahap 1 kalibrasi dimana ditemukan pola kegagalan yang tidak bertahap (*non-gradual*), melainkan bersifat ambang (*threshold-based*) pada seluruh langkah pengujian dari A1 hingga A5 (0,1 hingga 2 *request/detik*) menghasilkan tingkat

keberhasilan 100%, sementara pada langkah A6 (5 *request/detik*) kegagalan langsung muncul secara signifikan mulai permintaan ke-47. Pola ini mengindikasikan keberadaan mekanisme pembatasan laju permintaan yang bekerja berdasarkan ambang batas tetap, bukan penurunan performa secara gradual sebagaimana lazim ditemukan pada keterbatasan kapasitas *server* murni. Berdasarkan hasil ini, nilai 2 *request/detik* ditetapkan sebagai Rate Aman Maksimum (RAM) yang menjadi acuan penentuan parameter pada tahap pengujian utama.

3. Pengujian Tahap 2 (Skenario)

Pengujian Skenario 1

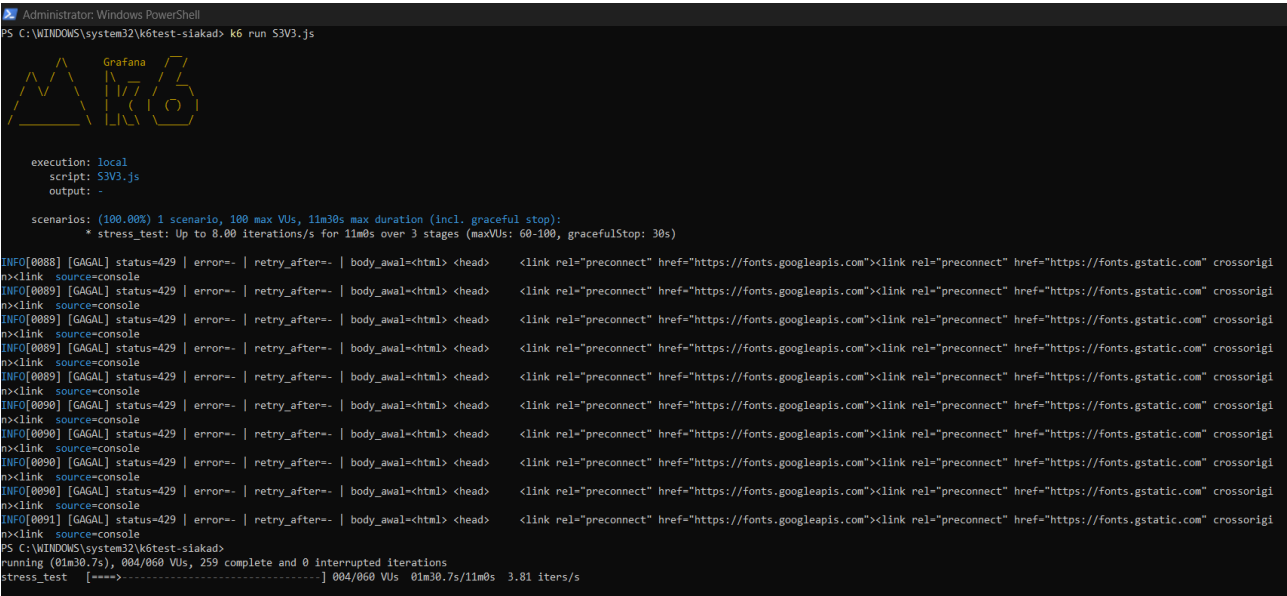
Pada pengujian skenario 1 (*Smoke Test*), peneliti menggunakan parameter berbasis rps hasil kalibrasi sebesar 2 *request/detik* ($1 \times \text{RAM}$) dengan durasi *ramp-up* selama 1 menit dan ditahan selama 1 menit.

Testing Result

Pada gambar 4 menunjukkan bahwa pengujian skenario 1 berhasil 100% permintaan sukses atau 0% gagal.

Pengujian Skenario 3

Pada pengujian skenario 3 (*Stress Test*), peneliti menggunakan parameter berbasis rps hasil kalibrasi sebesar 8 *request/detik* (4× RAM) dengan durasi *ramp-up* selama 5 menit dan ditahan selama 5 menit.



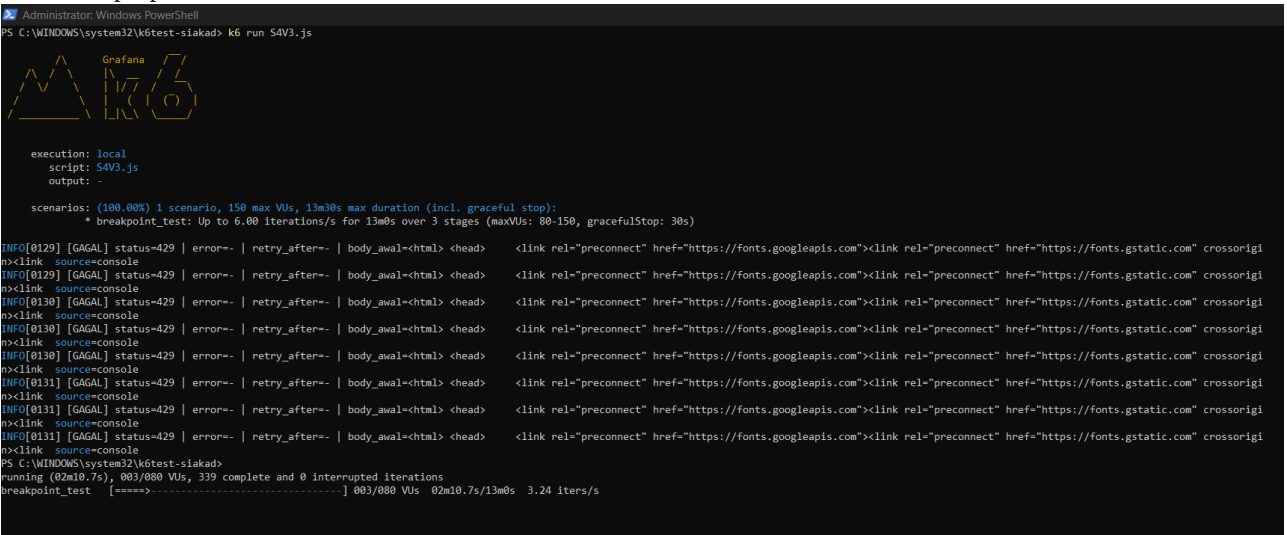
Gambar 6 Testing Result Skenario 3

Testing Result

Pada gambar 6 menunjukkan bahwa penolakan permintaan mulai konsisten terjadi pada kecepatan $\pm 3,8$ req/detik (dalam fase *ramp-up*).

Pengujian Skenario 4

Pada pengujian skenario 4 (*Breakpoint Test*), peneliti menggunakan parameter berbasis rps hasil kalibrasi sebesar 6 *request/detik* (bertahap, naik dari RAM) dengan durasi *ramp-up* selama 7 menit dan ditahan



Gambar 7 Testing Result Skenario 4

Testing Result

Pada gambar 7 menunjukkan bahwa penolakan permintaan mulai konsisten terjadi pada kecepatan $\pm 3,23$ req/detik (dalam fase *ramp-up*).

Interpretasi dan Analisis Data Hasil Pengujian

Berdasarkan dari hasil pengujian menggunakan 4 skenario, berikut merupakan data agregasi hasil pengujian:

Tabel 5. Data Hasil Skenario Pengujian

Skenario	Target kecepatan Permintaan	Durasi Ramp-Up	Durasi Hold (menit)	Hasil	
Smoke Test	2 req/detik (1× RAM)	1	1	Berhasil permintaan (0% gagal)	100% sukses
Load Test	3 req/detik (1,5× RAM)	3	3	Penolakan konsisten pada kecepatan awal fase	mulai pada ± 3 req/detik

Stress Test	8 req/detik (4× RAM)	5	5	Penolakan konsisten pada kecepatan $\pm 3,8$ req/detik (dalam fase <i>ramp-up</i>)	mulai pada $\pm 3,8$ req/detik (dalam fase <i>ramp-up</i>)
Break-point Test	6 req/detik (bertahap)	7	5	Penolakan konsisten pada kecepatan $\pm 3,23$ req/detik (dalam fase <i>ramp-up</i>)	mulai pada $\pm 3,23$ req/detik (dalam fase <i>ramp-up</i>)

Hasil pengujian utama yang disajikan pada tabel 5 memperlihatkan bahwa *Smoke Test* pada laju 1× RAM (2 *request/detik*) berhasil dilalui tanpa satu pun penolakan permintaan, mengonfirmasi validitas RAM yang

ditemukan pada tahap kalibrasi. Sebaliknya, ketiga skenario berikutnya (*Load*, *Stress*, dan *Break-point Test*), yang seluruhnya menetapkan target laju di atas RAM, secara konsisten menunjukkan munculnya penolakan permintaan sebelum mencapai target kecepatan maksimumnya masing-masing. Temuan ini menunjukkan bahwa ambang batas efektif sistem berada tidak jauh di atas RAM hasil kalibrasi, yaitu pada kisaran laju permintaan yang akan diuraikan lebih lanjut pada tabel 6.

Tabel 6. Konvergensi Titik Jenuh Antar Skenario

Skenario	Kecepatan Saat Penolakan Mulai Konsisten
<i>Load Test</i>	$\pm 3,00$ request/detik
<i>Stress Test</i>	$\pm 3,80$ request/detik
<i>Break-point Test</i>	$\pm 3,23$ request/detik
Rata-rata (estimasi titik jenuh)	$\pm 3,34$ request/detik

Ketiga skenario pengujian independen yang disajikan pada tabel 6 menunjukkan hasil yang saling konvergen, dengan titik jenuh sistem secara konsisten teramati pada rentang 3,0–3,8 request/detik, meskipun ketiganya menerapkan pola kenaikan beban (durasi *ramp-up* dan target kecepatan) yang berbeda-beda.

Konvergensi hasil dari tiga metode pengujian yang berbeda ini memperkuat validitas temuan melalui triangulasi data, sebagaimana dijelaskan pada pendekatan analisis ketiga di atas, dan mengindikasikan bahwa titik jenuh sistem SIAKAD bersifat konsisten dan dapat direplikasi, bukan merupakan anomali dari satu skenario pengujian tunggal.

Tabel 7. Ringkasan Ambang Batas Sistem

Parameter	Nilai
Rate Aman Maksimum (RAM)	2 request/detik
Ambang mulai penolakan konsisten	$\pm 3 - 3,8$ request/detik
Estimasi titik jenuh (rata-rata)	$\pm 3,34$ request/detik
Sumber penolakan	Lapisan WAF/reverse proxy (bukan aplikasi backend)
Indikator penolakan	HTTP 429 (<i>Too Many Requests</i>), tanpa header <i>Retry-After</i>

Secara keseluruhan, pada tabel 7 disajikan hasil analisis penelitian yang menunjukkan bahwa sistem SIAKAD Politeknik XYZ memiliki kapasitas aman sebesar 2 request/detik, dengan ambang penolakan mulai terjadi secara konsisten pada kisaran 3–3,8 request/detik.

Karakteristik respons penolakan yang seragam pada setiap skenario pengujian berupa halaman generik tanpa header *Retry-After* memperkuat indikasi bahwa mekanisme pembatasan ini diterapkan pada lapisan infrastruktur (WAF/reverse proxy), bukan pada logika aplikasi backend SIAKAD itu sendiri.

Analisis Ambang Batas

Temuan ambang batas 3,0–3,8 request/detik yang dalam penelitian ini tergolong sangat rendah untuk ukuran aplikasi layanan publik pada tingkat perguruan tinggi, yang secara umum diharapkan mampu melayani ratusan hingga ribuan pengguna secara bersamaan pada periode akses puncak seperti pengisian KRS. Terdapat beberapa penjelasan analitis potensi penyebab atas temuan ini.

Pertama, ambang batas tersebut kemungkinan besar diterapkan per alamat IP tunggal, bukan merupakan

kapasitas total sistem secara keseluruhan. Dalam kondisi penggunaan normal, setiap mahasiswa umumnya mengakses SIAKAD dari perangkat dan alamat IP yang berbeda-beda, sehingga jumlah permintaan yang dikirimkan oleh masing-masing pengguna secara individual jarang melampaui beberapa request per detik. Ambang batas rendah yang ditemukan dalam penelitian ini kemungkinan besar tidak mencerminkan kapasitas layanan sistem secara agregat, melainkan mekanisme proteksi terhadap pola akses tidak wajar dari satu sumber tunggal, sebagaimana lazim diterapkan untuk mencegah serangan *brute-force*, *credential stuffing*, maupun aktivitas *scraping* otomatis. Meskipun demikian, ambang batas per-IP yang terlalu ketat tetap berpotensi menimbulkan dampak negatif pada skenario akses dari jaringan dengan banyak pengguna di balik satu alamat IP publik yang sama (*Network Address Translation/NAT*).

Kedua, SIAKAD Politeknik XYZ teridentifikasi menggunakan infrastruktur *hosting* berbasis platform pihak ketiga (*Software-as-a-Service*), yang umumnya melayani banyak institusi pendidikan lain secara bersamaan dalam satu infrastruktur bersama (*multi-tenant*). Pada model layanan semacam ini, konfigurasi *rate limiting* sering kali diterapkan secara seragam (*generic default configuration*) oleh penyedia layanan untuk seluruh *tenant*/institusi tanpa penyesuaian khusus terhadap karakteristik dan skala kebutuhan masing-masing institusi, sehingga ambang batas yang diterapkan cenderung bersifat konservatif dan tidak selalu merepresentasikan kapasitas riil yang dibutuhkan oleh institusi dengan jumlah mahasiswa yang lebih besar.

Penelitian ini tidak dapat memastikan secara pasti penyebab spesifik dari rendahnya ambang batas tersebut, karena konfigurasi WAF/reverse proxy berada di luar kendali dan akses langsung peneliti. Diperlukan konfirmasi langsung dari pihak penyedia infrastruktur atau tim pengelola SIAKAD Politeknik XYZ untuk memastikan penyebab pastinya.

IV. KESIMPULAN DAN SARAN

Penelitian ini bertujuan untuk menganalisis *response time*, *throughput*, dan *error rate* pada website Sistem Informasi Akademik (SIAKAD) Politeknik XYZ menggunakan perangkat *load testing* k6, sekaligus mengidentifikasi keberadaan mekanisme *rate limiting* yang memengaruhi validitas hasil pengujian. Berdasarkan proses pengujian yang telah dilaksanakan melalui pendekatan kalibrasi bertahap, dapat disimpulkan beberapa hal sebagai berikut.

Pertama, sistem SIAKAD Politeknik XYZ memiliki Rate Aman Maksimum (RAM) sebesar 2 request/detik, yaitu kecepatan permintaan tertinggi yang masih dapat diproses sistem tanpa memicu penolakan permintaan (HTTP 429). Pada kecepatan ini, sistem menunjukkan kinerja yang stabil dengan *error rate* sebesar 0% dan *response time* median di bawah 100 milidetik.

Kedua, ambang batas mulai terjadinya penolakan permintaan secara konsisten ditemukan berada pada kisaran 3,0–3,8 request/detik, dengan estimasi titik jenuh rata-rata sebesar 3,34 request/detik. Temuan ini diperoleh

melalui triangulasi tiga skenario pengujian independen (*Load Test*, *Stress Test*, dan *Break-point Test*) dengan pola kenaikan beban yang berbeda-beda, sehingga hasil yang konvergen tersebut dapat dianggap valid dan dapat direplikasi.

Ketiga, karakteristik respons penolakan permintaan berupa halaman generik yang seragam pada setiap kejadian tanpa disertai *header Retry-After* mengindikasikan bahwa mekanisme pembatasan laju permintaan diterapkan pada lapisan infrastruktur di luar aplikasi utama, yaitu *Web Application Firewall (WAF)* atau *reverse proxy*, bukan pada logika aplikasi backend SIAKAD itu sendiri. Temuan ini menegaskan pentingnya mengidentifikasi sumber penolakan permintaan secara tepat sebelum menginterpretasikan matrik *error rate* pada pengujian performa sistem yang dilindungi lapisan proteksi tambahan, karena kegagalan yang teramati tidak selalu mencerminkan keterbatasan kapasitas aplikasi yang sesungguhnya.

Keempat, penelitian ini menunjukkan bahwa pendekatan pengujian beban konvensional berbasis jumlah *virtual user (VU)* tidak selalu representatif pada sistem yang dilengkapi mekanisme *rate limiting*, karena jumlah permintaan aktual yang dihasilkan oleh satu VU dapat bervariasi tergantung waktu respons server. Pendekatan kalibrasi bertahap berbasis kecepatan permintaan (*request per second*) yang diterapkan dalam penelitian ini terbukti mampu menghasilkan data pengujian yang lebih presisi, valid, dan dapat dipertanggungjawabkan, sehingga dapat menjadi rujukan metodologis bagi penelitian atau praktik pengujian performa pada sistem informasi sejenis di masa mendatang.

Berdasarkan temuan-temuan tersebut, disusun beberapa saran sebagai berikut.

Bagi pengelola/pihak institusi Politeknik XYZ, disarankan untuk melakukan evaluasi ulang terhadap konfigurasi mekanisme *rate limiting* yang diterapkan pada lapisan *WAF/reverse proxy* saat ini, mengingat ambang batas 2–3 *request/detik* per alamat IP berpotensi terlalu ketat untuk mengakomodasi skenario akses bersamaan dari banyak pengguna pada satu jaringan yang sama (misalnya jaringan kampus dengan konfigurasi NAT), khususnya pada periode akses puncak seperti pengisian Kartu Rencana Studi (KRS) atau publikasi hasil ujian. Selain itu, pengelola sistem disarankan menambahkan *header Retry-After* pada respons penolakan permintaan, agar klien (baik pengguna maupun sistem pemantauan) dapat mengetahui durasi tunggu yang sesuai sebelum mengirimkan permintaan berikutnya, sehingga dapat meningkatkan pengalaman pengguna sekaligus efisiensi trafik.

Bagi penelitian selanjutnya, disarankan untuk melakukan pengujian lanjutan dengan melibatkan koordinasi langsung bersama pihak penyedia infrastruktur (*WAF/CDN*) guna memperoleh konfirmasi resmi mengenai parameter *rate limiting* yang diterapkan, sehingga hasil pengujian dapat divalidasi secara lebih akurat. Penelitian selanjutnya juga disarankan untuk memperluas cakupan pengujian pada *endpoint* yang bersifat transaksional (seperti proses login, pengisian KRS, atau unggah dokumen), tidak terbatas pada halaman

utama saja, mengingat *endpoint* tersebut umumnya melibatkan proses komputasi dan akses basis data yang lebih kompleks sehingga berpotensi memiliki karakteristik performa yang berbeda. Selain itu, penelitian lanjutan dapat mempertimbangkan penggunaan pendekatan pengujian terdistribusi dari beberapa alamat IP atau lokasi geografis yang berbeda untuk mensimulasikan pola akses pengguna yang lebih representatif terhadap kondisi penggunaan sesungguhnya.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih sebesar-besarnya kepada seluruh civitas akademika Politeknik XYZ atas bantuannya dalam penyusunan makalah ini. Ucapan terima kasih juga ditujukan kepada orang tua atas dukungan moral dan dorongan yang terus diberikan hingga artikel ini selesai. Terima kasih khusus disampaikan kepada tim editor SIMTEK atas bantuannya dalam penyuntingan dan penyelesaian paper ini.

DAFTAR PUSTAKA

- [1] E. D. Utari, F. S. Atmadja, R. R. Nida, and L. Rabbaniyah, "Evaluation of the Quality of Academic Information Services on the FEB UNJ Website Based on the WebQual and EUCS Models," *J. Comput. Networks, Archit. High Perform. Comput.*, vol. 7, no. 3, pp. 959–969, 2025, doi: <https://doi.org/10.47709/cnaphc.v7i3.6559>.
- [2] O. Onunka, M. O. Eigbadon, T. Onunka, A. A. Fawole, G. C. Uzougbo, and O. O. Ayeni, "Investigating The Effectiveness Of A University'S Library Website In Supporting Scholarly Pursuits," *Educ. Learn. Dev. Nations*, vol. 1, no. 2, pp. 89–92, 2023, doi: [10.26480/eldn.02.2023](https://doi.org/10.26480/eldn.02.2023).
- [3] P. Matthews, "A Path-Dependent , Multi-Layered Architecture for Runtime Governance and Post-Quantum Cryptographic Assurance of Autonomous AI Agents," *Path-Dependent Runtime AI Gov.*, no. June, pp. 1–14, 2026, doi: <https://ssrn.com/abstract=6898119>.
- [4] A. R. Kretzer and F. Siqueira, "Empirical Evaluation of a Data Lakehouse Architecture for Streaming Industrial Data," *J. Ind. Inf. Integr.*, 2026, doi: <http://dx.doi.org/10.2139/ssrn.6986769>.
- [5] N. Srivastava, U. Kumar, and P. Singh, "Software and Performance Testing Tools," *J. Informatics Electr. Electron. Eng.*, vol. 02, no. 001, pp. 1–12, 2021, doi: <https://doi.org/10.54060/JIEEE/002.01.001>.
- [6] S. Pargaonkar, "A Comprehensive Review of Performance Testing Methodologies and Best Practices : Software Quality Engineering," in *International Journal of Science and Research (IJSR)*, 2023, pp. 2008–2014. doi: [10.21275/SR23822111402](https://doi.org/10.21275/SR23822111402).
- [7] A. Z. Musthafawi and A. Mas, "Performance Testing On The Shopee Website In The Pandemic Period Of Covid-19," in *International Conference on Sustainable Information Engineering and Technology*, 2020, pp. 195–199. doi: <https://doi.org/10.1145/3427423.3427457>.
- [8] A. C. Barus, E. S. Sinambela, I. Purba, J. Simatupang, M. Marpaung, and N. Pandjaitan, "Performance Testing and Optimizaton of DiTenun Website," *J. Appl. Sci. Eng. Technol. Educ.*, vol. 4, no. 1, 2022, doi: <https://doi.org/10.35877/454RI.asci841>.
- [9] A. Suprpto and D. Sasongko, "Evaluasi Performa Website Berdasarkan Pengujian Beban Dan Stress Menggunakan Loadimpact (Studi Kasus Website Iain Salatiga)," *J. Ilm. NERO*, vol. 6, no. 1, pp. 31–37, 2021, doi: <https://doi.org/10.21107/nero.v6i1.198>.
- [10] F. A. Mufarroha, H. A. Farisul, M. Arifatul, D. R. Anamisa, A. A. Supianto, and A. Jauhari, "Quality Assurance of Academic Websites using Performance Testing Tools," in *The International Conference on Science Technology*, 2023. doi: <https://doi.org/10.47577/technium.v16i.9985>.
- [11] H. Setiawan and D. Enda, "Pengujian Load Testing Website Jurusan Teknik Informatika Politeknik Negeri Bengkalis," *J. Tek.*

- Inform., vol. 5, no. 1, pp. 1–13, 2024, doi: <https://doi.org/10.58794/jekin.v5i1.820> Pengujian.
- [12] A. Niarman, “Comparative Analysis of PHP Frameworks for Development of Academic Information System Using Load and Stress Testing,” in *International Journal Software Engineering and Computer Science (IJSECS)*, 2023, pp. 424–436. doi: <https://doi.org/10.35870/ijsecs.v3i3.1850>.
 - [13] F. Waheed, F. Azam, M. W. Anwar, and Y. Rasheed, “Model Driven Approach for Automatic Script Generation in Stress Testing of Web Applications,” in *International Conference on Computer and Technology Applications*, 2020, pp. 46–50. doi: <https://doi.org/10.1145/3397125.3397137>.
 - [14] E. Battista, S. Di Martino, S. Di Meglio, F. Scippacercola, L. Libero, and L. Starace, “E2E-Loader: A Framework to Support Performance Testing of Web Applications,” in *2023 IEEE Conference on Software Testing, Verification and Validation (ICST)*, IEEE, 2023, pp. 351–361. doi: [10.1109/ICST57152.2023.00040](https://doi.org/10.1109/ICST57152.2023.00040).
 - [15] S. Kaushal, “Load Testing Analyzer for Web Application,” in *International Journal of Innovative Research in Computer Science & Technology (IJIRCST)*, 2020, pp. 171–173. doi: <https://doi.org/10.21276/ijircst.2020.8.3.25>.
 - [16] Y. Kravchenko, O. Leshchenko, O. Trush, N. Dakhno, and P. Krasnopyorov, “Optimizing and Improvement a Web Application Using Open Source Tools,” *Inf. Technol. Implement.*, pp. 368–379, 2023.
 - [17] Q. M. Rosul and E. Sulistiyani, “Analisis Perbandingan Kinerja Alat Pengujian Beban Perangkat Lunak: Apache Jmeter, Gatling, Dan K6,” *J. Inform. Polinema*, pp. 393–400, 2024, doi: <https://doi.org/10.33795/jip.v12i2.8693>.