Journal of Digital Business and Technology Innovation (DBESTI)

https://journal.nurulfikri.ac.id/index.php/DBESTI
P-ISSN: 3047-1028 E-ISSN: 3032-775X

EVALUASI PENGUJIAN KINERJA MENGGUNAKAN JMETER UNTUK MENUNJANG STABILITAS APLIKASI LAYANAN PERBANKAN PADA PT BANK RAKYAT INDONESIA TBK

Ida Hamidah[1], Imam Haromain[2], Ishom Muhammad Drehem[3]

[1,2,3]Teknik Informatika, Sekolah Tinggi Teknologi Terpadu Nurul Fikri
Jakarta Selatan, DKI Jakarta, Indonesia 12640
ini.idahamidah@gmail.com, haromain@nurulfikri.ac.id, ishom.drehem@nurulfikri.ac.id

Abstract

The advancement of information technology has had a significant impact on the banking sector, including PT Bank Rakyat Indonesia, which relies on the New Delivery System (NDS) application to support its digital services. NDS is designed to ensure that banking services remain stable, scalable, and fast, even in the face of significant spikes in transaction volumes. As one of BRI's flagship applications, NDS has proven reliable in supporting banking operations. However, performance evaluation is still necessary to ensure the system's readiness to handle growing demands in the digital era. This study uses Performance Testing methods to analyze the performance of NDS using the Apache JMeter tool. Testing is carried out through Load Testing and Stress Testing to measure the application's response under various workloads. Evaluation parameters include response time, throughput, and transaction failure rate. The testing simulates different usage scenarios, from normal to extreme load conditions, to ensure system reliability in various operational situations. The results show that under normal load, NDS has an average response time ranging from 244 ms to 739 ms, throughput of up to 22.26 samples/second, and a transaction failure rate of 0%. However, in high-load scenarios (Stress Testing) with 500 users, the response time increases to 12,691 ms, throughput reaches 36.03 samples/second, and the transaction failure rate is 73.10%. The tests also reveal that CPU usage is close to its maximum capacity (85%), while memory usage remains stable. The analysis shows that NDS requires further optimization in middleware and load distribution to improve application performance and stability. This study provides valuable insights for the development of similar applications in the banking sector to support more stable, efficient, and responsive services.

Keywords: *Banking*, JMeter, *Load Testing, Stress Testing, Performance Testing*

Abstrak

Perkembangan teknologi informasi memberikan dampak signifikan terhadap sektor perbankan, termasuk PT Bank Rakyat Indonesia yang mengandalkan aplikasi New Delivery System (NDS) untuk mendukung layanan digital. NDS dirancang untuk memastikan layanan perbankan tetap stabil, skalabel, dan cepat, meskipun menghadapi lonjakan *volume* transaksi yang signifikan. Sebagai salah satu aplikasi unggulan BRI, NDS telah terbukti andal dalam mendukung operasional perbankan. Namun, evaluasi performa tetap diperlukan untuk memastikan kesiapan sistem dalam menangani kebutuhan yang terus berkembang di era digital. Penelitian menggunakan metode *Performance Testing* untuk menganalisis kinerja NDS dengan alat Apache JMeter. Pengujian dilakukan melalui *Load Testing* dan *Stress Testing* guna mengukur respons aplikasi terhadap berbagai beban kerja. Hasil penelitian menunjukkan bahwa pada beban normal, NDS memiliki waktu *response* rata-rata 244 ms hingga 739 ms, *throughput* hingga 22,26 sampel/detik, dan tingkat kegagalan transaksi 0%. Namun, pada skenario beban tinggi (*Stress Testing*) dengan 500 pengguna, waktu *response* meningkat menjadi 12.691 ms, *throughput* mencapai 36,03 sampel/detik, dan tingkat kegagalan transaksi sebesar 73,10%. Pengujian menunjukkan bahwa penggunaan CPU mendekati kapasitas maksimum (85%), sedangkan penggunaan memori stabil. Hasil analisis menunjukkan bahwa NDS membutuhkan optimasi lebih lanjut pada *middleware* dan distribusi beban untuk meningkatkan performa dan stabilitas aplikasi. Penelitian ini memberikan referensi bagi pengembangan aplikasi serupa di sektor perbankan.

Kata kunci: Bank, JMeter, *Load Testing, Stress Testing, Performance Testing*

Diterima 28 Januari 2025, Direvisi 3 Juni 2025, Diterima untuk publikasi 4 Juni 2025

1. PENDAHULUAN

Perkembangan teknologi informasi telah membawa dampak besar pada berbagai sektor, termasuk perbankan. Digitalisasi layanan menjadi kebutuhan mendesak bagi bank untuk memberikan layanan yang lebih cepat, efisien, dan andal [1]. PT Bank Rakyat Indonesia (Persero) Tbk (BRI) sebagai salah satu bank terbesar di Indonesia, terus berinovasi dalam layanan digital, terutama untuk mendukung segmen Usaha Mikro, Kecil, dan Menengah (UMKM). Dengan semakin meningkatnya *volume* transaksi digital, BRI membutuhkan aplikasi yang stabil, skalabel, dan memiliki kecepatan respons tinggi untuk mendukung kelancaran operasional. Salah satu aplikasi penting yang digunakan BRI adalah New Delivery System (NDS) yang dirancang untuk mendukung kebutuhan seluruh unit kerja BRI, termasuk di daerah terpencil [2]. NDS diandalkan untuk menjaga konsistensi layanan, meskipun menghadapi tantangan berupa peningkatan jumlah transaksi yang signifikan. Data menunjukkan bahwa sekitar 98,9% transaksi BRI dilakukan melalui kanal digital [3] dengan nilai transaksi yang mencapai triliunan rupiah setiap tahunnya. Oleh karena itu, performa NDS menjadi sangat krusial dalam menjaga kepuasan nasabah serta reputasi bank. Namun, peningkatan jumlah transaksi digital tersebut menghadirkan tantangan besar dalam menjaga stabilitas dan performa NDS. Pada tahun 2023, total aset BRI mencapai Rp1.965 triliun, dengan transaksi finansial yang meningkat lebih dari dua kali lipat menjadi Rp2.669 triliun [3]. Jika performa sistem tidak dikelola dengan baik, ketidakstabilan dapat menyebabkan penurunan kualitas layanan, lambatnya waktu *response*, dan kegagalan transaksi, yang pada akhirnya merugikan bank secara finansial. Untuk mengatasi hal ini, pengujian performa yang komprehensif yaitu melalui *Performance Testing*. *Performance Testing* bertujuan untuk mengevaluasi stabilitas, skalabilitas, dan kecepatan aplikasi di bawah beban kerja tertentu [4].

Pengujian ini membantu memahami seberapa baik aplikasi dapat beroperasi di bawah tekanan beban yang tinggi, serta memastikan bahwa sistem mampu menangani peningkatan volume transaksi sehingga BRI dapat mengidentifikasi potensi permasalahan performa sebelum terjadi gangguan dalam operasional. Penelitian ini bertujuan menguji performa NDS menggunakan Apache JMeter dengan fokus pada skenario *Load Testing* dan *Stress Testing*. Untuk melakukan *Performance Testing*, Apache JMeter sering digunakan sebagai alat pengujian yang dapat mensimulasikan ribuan pengguna atau transaksi secara bersamaan. JMeter mampu mengukur waktu *response*, *throughput, error rate,* serta penggunaan sumber daya aplikasi [5].

Pengujian yang komprehensif diharapkan mampu memberikan rekomendasi yang signifikan bagi BRI dalam menjaga stabilitas aplikasi serta meningkatkan pengalaman nasabah di era digitalisasi perbankan. Hasil dari penelitian ini juga diharapkan dapat memberikan wawasan praktis dalam mendukung kelangsungan dan keandalan layanan perbankan digital, khususnya di tengah lonjakan transaksi digital yang terus meningkat.

2. METODE PENELITIAN

Penelitian ini menggunakan penelitian deskriptif, yang bertujuan untuk menggambarkan dan menganalisis performa aplikasi New Delivery System (NDS) milik PT Bank Rakyat Indonesia dibawah kondisi beban transaksi yang berbeda. Pendekatan deskriptif digunakan untuk memberikan gambaran yang jelas mengenai stabilitas, kecepatan, dan skalabilitas aplikasi berdasarkan hasil pengujian performa [6]. Hasil dari pengujian ini akan dianalisis dan disajikan secara deskriptif untuk memberikan wawasan tentang area yang perlu dioptimalkan, guna menjaga stabilitas aplikasi NDS BRI. Lalu, metode analisis data yang digunakan adalah metode kuantitatif karena data yang dihasilkan dari pengujian performa aplikasi NDS berupa angka dan parameter teknis, seperti waktu *response, throughput*, dan *error rate*.

2.1 Metode Pengumpulan Data

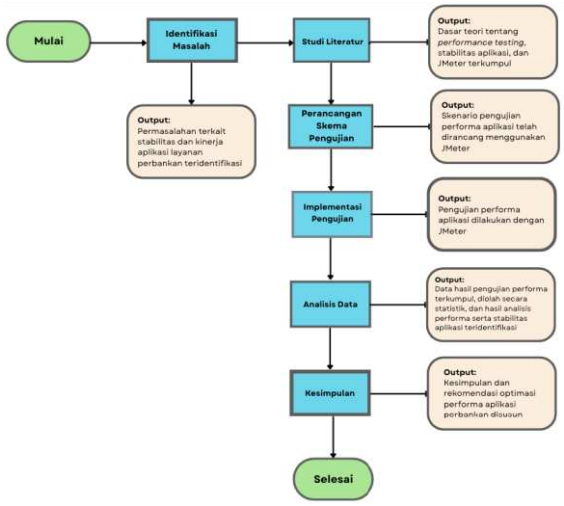
Metode pengumpulan data dalam penelitian ini melibatkan pendekatan eksperimen, studi literatur, observasi, dan wawancara. Eksperimen dilakukan menggunakan Apache JMeter dengan dua jenis pengujian utama, yaitu *Load Testing* dan *Stress Testing*. Data yang dikumpulkan dari eksperimen ini mencakup waktu *response*, *throughput*, penggunaan CPU dan memori, serta tingkat kegagalan transaksi. Selain itu, studi literatur dilakukan untuk meninjau teori dan penelitian sebelumnya yang relevan, mencakup topik *Performance Testing, Load Testing, Stress Testing*, serta stabilitas aplikasi. Observasi juga dilakukan selama eksperimen untuk mengamati *response* aplikasi secara langsung dibawah kondisi beban tinggi, dengan fokus pada stabilitas sistem dan masalah teknis seperti *bottleneck* atau *timeout*. Wawancara dengan tim pengembang IT BRI dilaksanakan untuk memperoleh wawasan mengenai operasional aplikasi NDS, kendala teknis, serta masukan terkait performa sistem.

Penelitian ini menggunakan metode pengujian *Load Testing* dan *Stress Testing* dengan bantuan Apache JMeter. *Load Testing* bertujuan untuk memastikan aplikasi NDS mampu menangani beban transaksi harian normal tanpa penurunan performa, dengan mensimulasikan jumlah pengguna sesuai dengan *volume* transaksi aktual di BRI. Sementara itu, *Stress Testing* dilakukan dengan meningkatkan beban secara bertahap hingga aplikasi mencapai titik kegagalan atau mengalami penurunan performa signifikan. Metode pengujian ini bertujuan untuk mengevaluasi stabilitas, skalabilitas, dan kecepatan aplikasi dalam mendukung layanan digital BRI.

2.2 Tahapan penelitian

Penelitian ini terdiri dari beberapa tahapan sistematis yang dirancang untuk mengevaluasi kinerja NDS yang digunakan

oleh PT Bank Rakyat Indonesia (Persero) Tbk melalui pengujian performa.



Gambar 1. *Flowchart* Tahapan Penelitian

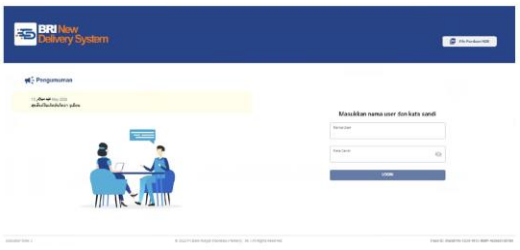
Pada gambar 1 tahapan penelitian akan mengidentifikasi masalah performa aplikasi NDS BRI, mengumpulkan teori melalui studi literatur, dan merancang pengujian menggunakan JMeter. Setelah pengujian dilakukan, data dianalisis untuk mengevaluasi stabilitas dan performa aplikasi. Hasilnya dirangkum dalam kesimpulan dan rekomendasi untuk optimasi aplikasi.

3. HASIL DAN PEMBAHASAN

Bagian ini memaparkan hasil penelitian terkait pengujian performa aplikasi New Delivery System (NDS) di PT Bank Rakyat Indonesia menggunakan Apache JMeter. Hasil penelitian difokuskan pada analisis parameter performa seperti waktu *response*, *throughput,* dan tingkat kegagalan transaksi pada berbagai skenario pengujian, yaitu *Load Testing* dan *Stress Testing*.

3.1 Aplikasi Pengujian

New Delivery System (NDS) merupakan aplikasi yang dirancang dan diimplementasikan di cabang-cabang Bank BRI, mencakup seluruh level organisasi, mulai dari Unit hingga Kantor Wilayah (Kanwil). Pada gambar 2 di bawah terdapat menu *Login* NDS.

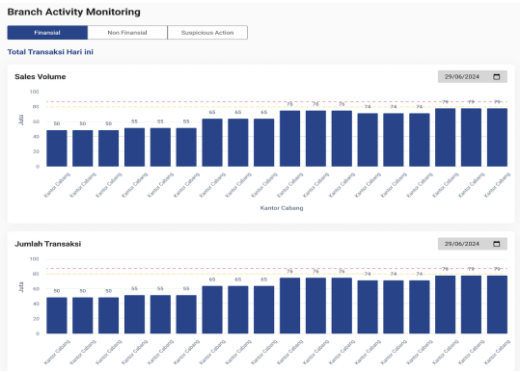


Gambar 2. *Login* NDS

Sebagai salah satu lembaga keuangan terbesar di Indonesia dengan jaringan cabang yang luas, efektivitas pengawasan dan *mitigasi* risiko operasional menjadi elemen krusial dalam mendukung kelancaran dan stabilitas layanan perbankan. Untuk menjawab kebutuhan tersebut, NDS dirancang untuk meningkatkan efisiensi dan akurasi sistem transaksi yang melibatkan berbagai jenis layanan perbankan. Salah satu fitur unggulan yang dikembangkan dalam aplikasi ini adalah *Risk Monitoring Management,* pada gambar 3 terdapat menu *Risk Monitoring Management.*

Gambar 3. Menu *Risk Monitoring Management*

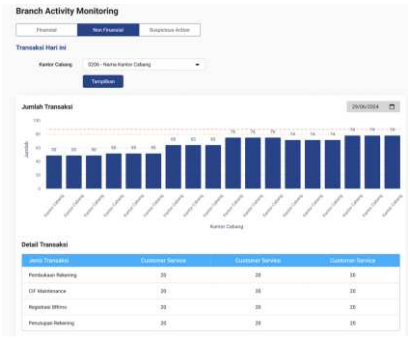
Menu yang berfungsi untuk mengawasi seluruh transaksi yang terjadi di BRI, termasuk transaksi finansial, non-finansial, hingga aktivitas mencurigakan (*suspicious activity*) yang terindikasi tidak sesuai dengan kebijakan atau keputusan bisnis yang ditetapkan oleh BRI. Salah satu fitur yang berada dalam menu *Risk Monitoring Management* adalah *Monitoring Risk Finansial.*



Gambar 4. Fitur *Monitoring Risk Finansial*

Pada fitur tersebut nantinya akan mencatat seluruh transaksi finansial yang lakukan oleh unit kerja BRI. Transaksi finansial mencakup semua aktivitas yang melibatkan perpindahan dana atau pengelolaan keuangan nasabah, baik secara individu maupun bisnis. Contohnya meliputi transfer antar rekening. Transaksi yang sudah dilakukan akan menjadi data *bar chart sales volume* dan jumlah transaksi, agar dapat di *monitoring* setiap harinya untuk kegiatan unit kerja dalam transaksi finansial.

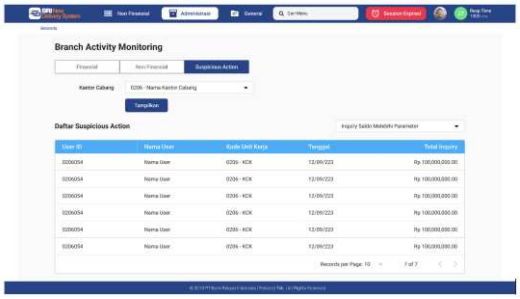
Selain memonitori transaksi finansial, fitur lain *Risk Monitoring Management* adalah *Risk Monitoring Non Finansial.*



Gambar 5. Fitur *Risk Monitoring Non Finansial*

Pada gambar 5 digunakan untuk memantau transaksi non-finansial BRI, mencakup semua aktivitas yang tidak melibatkan perpindahan dana, tetapi tetap terkait dengan layanan perbankan. Contoh pembukaan rekening, CIF *Maintenance*, Registrasi Brimo dan penutupan rekening.

Fitur terakhir dari gambar 6 di bawah yaitu *Risk Management Monitoring* adalah *Suspicious activity.*



Gambar 6. Fitur *Risk Monitoring Suspicious Activity*

Suspicious activity dalam perbankan merujuk pada aktivitas transaksi yang dianggap mencurigakan karena tidak sesuai dengan pola transaksi nasabah atau melanggar kebijakan dan regulasi yang berlaku. Aktivitas ini sering kali menjadi indikator awal dari potensi tindak kejahatan seperti pencucian uang (*money laundering*), pendanaan teroris, atau penipuan. Pengawasan terhadap seluruh transaksi BRI membutuhkan kestabilan fitur dan performa yang optimal agar dapat menampilkan data transaksi dalam jumlah besar secara akurat dan dalam waktu nyata. Dalam hal ini, performa sistem sangat berpengaruh terhadap kecepatan akses, validitas data, dan kemampuan analisis berbasis data operasional. Dengan memastikan kestabilan sistem, proses monitoring dapat dilakukan secara efektif untuk mendukung pengambilan keputusan berdasarkan data serta meningkatkan keamanan dan efisiensi operasional. Fitur monitoring ini dirancang untuk memberikan kemampuan analitik data operasional secara *real-time*.

3.2 *Performance Testing*

Pengujian perangkat lunak atau *Software Testing* adalah proses mengevaluasi sebuah aplikasi perangkat lunak untuk memastikan bahwa perangkat lunak tersebut berfungsi sesuai dengan yang diharapkan, memenuhi persyaratan yang telah ditentukan, dan bebas dari cacat atau *bug*. Ini merupakan bagian penting dari *software development life cycle* yang bertujuan untuk menjamin kualitas, keamanan, dan kinerja produk [7].

Tujuannya adalah untuk mengidentifikasi perbedaan antara perilaku yang sebenarnya dengan yang diharapkan, sehingga pengembang dapat memperbaiki kesalahan sebelum perangkat lunak diimplementasikan. Terdapat dua kategori utama dalam pengujian perangkat lunak, yaitu *Functional Testing* dan *Non-Functional Testing* (atau sering disebut juga *Performance Testing*) [7].

Functional Testing berfokus pada verifikasi bahwa perangkat lunak berperilaku sesuai dengan persyaratan yang telah ditetapkan, memastikan bahwa fitur-fitur berjalan sesuai yang diinginkan. Sementara itu, *Non-Functional Testing* menilai kinerja, keamanan, dan keandalan perangkat lunak dalam berbagai kondisi, memastikan bahwa sistem dapat beroperasi secara optimal di bawah beban dan tekanan.

Dalam konteks NDS, pengujian juga harus memastikan bahwa sistem memiliki stabilitas, skalabilitas, dan kecepatan yang diperlukan untuk menunjang transaksi dalam jumlah besar. NDS harus mampu menangani beban transaksi yang tinggi secara konsisten, terutama di lingkungan perbankan yang melibatkan jutaan transaksi setiap harinya.

Oleh karena itu, stabilitas sistem dalam berbagai kondisi operasional, kemampuan sistem untuk bereskalasi sesuai dengan peningkatan jumlah pengguna, serta kecepatan dalam memproses transaksi menjadi fokus penting dalam pengujian. Hal ini sejalan dengan penelitian yang akan membahas *Performance Testing* untuk memastikan bahwa NDS dapat memenuhi tuntutan tersebut, sehingga layanan perbankan tetap berjalan lancar dan efisien tanpa gangguan.

Peneliti Shravan Pargaonkar dalam penelitiannya yang berjudul *"A Comprehensive Review of Performance Testing Methodologies and Best Practices: Software Quality Engineering"* menekankan pentingnya pengujian kinerja dalam memastikan keandalan, skalabilitas, dan responsivitas aplikasi perangkat lunak [8].

Pargaonkar menyoroti bahwa pengujian kinerja berperan penting dalam mendeteksi *bottleneck*, mengoptimalkan pemanfaatan sumber daya, serta meningkatkan pengalaman pengguna. Pengujian perangkat lunak adalah proses yang tidak bisa diabaikan dalam siklus pengembangan perangkat lunak untuk menjamin produk yang andal dan berkualitas tinggi [8]. Pentingnya merancang kasus uji yang relevan dan realistis, dengan data pengguna sintetik maupun nyata, ditekankan untuk mendapatkan evaluasi kinerja yang akurat.

Metodologi ini memberikan perspektif yang berbeda tentang perilaku aplikasi, sehingga memungkinkan evaluasi yang komprehensif terhadap karakteristik kinerja aplikasi.

Dibawah ini adalah beberapa metodologi utama dalam pengujian performa:

1. Pengujian Beban (*Load Testing*): Pengujian beban melibatkan pemberian berbagai tingkat beban pengguna pada sistem perangkat lunak untuk mengevaluasi waktu *response*, penggunaan sumber daya, dan kinerja keseluruhan dibawah skenario penggunaan yang berbeda. Metodologi ini membantu mengidentifikasi hambatan kinerja, seperti waktu *response* yang lambat atau kegagalan server, yang mungkin muncul seiring dengan peningkatan lalu lintas pengguna [8].

2. Pengujian Stres (*Stress Testing*): Pengujian stres membawa evaluasi kinerja lebih jauh dengan mendorong aplikasi ke batas maksimal atau bahkan melebihi kapasitasnya. Dengan menerapkan beban ekstrem, pengujian stres menunjukkan bagaimana sistem berperilaku ketika sumber daya terbatas, koneksi database jenuh, atau komponen perangkat keras berada dibawah tekanan. Metodologi ini mengungkap titik-titik kegagalan potensial dan membantu menilai mekanisme pemulihan sistem [8].

Untuk menjalankan kedua metode tersebut, JMeter digunakan sebagai tools dalam *Performance Testing*. JMeter memungkinkan simulasi jumlah pengguna yang besar untuk *Load Testing* dan pengujian di luar kapasitas normal untuk *Stress Testing*. JMeter mengukur waktu *response*, *throughput,* dan stabilitas sistem selama pengujian, sehingga dapat membantu mengidentifikasi batasan kinerja dan potensi masalah yang mungkin terjadi ketika sistem NDS dihadapkan pada beban kerja yang berat.

3.3 Apache JMeter

JMeter atau Apache JMeter adalah perangkat lunak open-source yang dikembangkan oleh Apache Software Foundation, salah satu alat *open-source* yang sering digunakan untuk melakukan pengujian performa [9].

JMeter mendukung berbagai jenis pengujian kinerja, termasuk *Load testing*, *Stress Testing*. JMeter digunakan untuk melakukan simulasi beban kerja dengan jumlah pengguna virtual yang tinggi guna menguji seberapa baik suatu aplikasi dapat menangani beban tersebut. Pengujian ini membantu mengidentifikasi apakah aplikasi memiliki masalah seperti waktu respon yang lambat, performa yang tidak stabil, atau gagal ketika beban meningkat. JMeter mendukung berbagai protokol seperti HTTP, HTTPS, SOAP, FTP, dan database melalui JDBC. JMeter memiliki beberapa komponen kunci yang mendukung pengujian kinerja, di antaranya:

1. *Thread Group*
Mewakili sejumlah pengguna virtual (threads) yang mengirimkan permintaan ke server secara bersamaan. Pengaturan Thread Group di JMeter memungkinkan simulasi skenario pengujian nyata, seperti berapa banyak pengguna yang mengakses aplikasi dan dalam periode waktu apa.

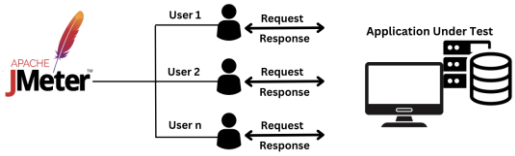
2. *Samplers*
Digunakan untuk mengirim permintaan ke server. JMeter mendukung berbagai jenis *samplers*, termasuk HTTP Request, JDBC *Request* (untuk pengujian basis data), dan FTP *Request*.

3. *Listeners*
Bertanggung jawab untuk menampilkan hasil pengujian dalam berbagai format, seperti grafik, tabel, atau file log. Hasil ini membantu menganalisis kinerja aplikasi, termasuk waktu respon, jumlah transaksi per detik, dan tingkat *error*.

Dapat dipahami bahwa pengujian performa menggunakan *Load Testing* dan *Stress Testing* dan alat seperti Apache JMeter sangat berguna untuk mengidentifikasi kemampuan sistem dalam menghadapi beban kerja yang tinggi dan memastikan stabilitas serta kecepatan *response* dalam kondisi operasional nyata. Stabilitas sistem merujuk pada kemampuan sistem untuk tetap beroperasi dengan lancar dan konsisten meskipun dihadapkan pada beban kerja yang terus meningkat atau perubahan kondisi yang signifikan.

3.4 Perancangan Sistem Pengujian

Perancangan sistem pengujian performa bertujuan untuk menciptakan lingkungan pengujian yang mendekati kondisi nyata [10].



Gambar 7. Perancangan Sistem Pengujian

Diagram pada gambar 7 di atas tersebut menggambarkan arsitektur pengujian performa dengan menggunakan Apache JMeter. Berikut adalah penjelasan dari setiap komponen yang terdapat dalam diagram:

3.4.1 Apache JMeter
Apache JMeter merupakan perangkat lunak yang digunakan untuk melakukan pengujian performa terhadap aplikasi berbasis web atau sistem *backend* lainnya. Dalam konteks diagram ini, JMeter berfungsi sebagai alat simulasi untuk menciptakan sejumlah pengguna virtual (*virtual users*) yang mengakses aplikasi secara bersamaan.

3.4.2 *Virtual Users* (*User* 1, *User* 2, hingga User n)
Virtual users adalah entitas yang disimulasikan oleh JMeter untuk merepresentasikan pengguna nyata yang melakukan interaksi dengan aplikasi yang diuji. Setiap *virtual user* bertugas mengirimkan permintaan (*request*) ke aplikasi dan menerima tanggapan (*response*) dari sistem.

3.4.3 *Request* Komponen
Merepresentasikan aktivitas pengiriman data atau permintaan dari *virtual users* ke aplikasi yang diuji. Permintaan ini dapat berupa permintaan HTTP, panggilan API, atau jenis permintaan lainnya yang relevan dengan sistem yang diuji.

3.4.4 *Response*

Response adalah data yang dikirimkan kembali oleh aplikasi sebagai tanggapan atas request yang diterima. Parameter performa seperti waktu respon (response time) dan tingkat keberhasilan tanggapan (success rate) dianalisis berdasarkan respons ini.

3.4.5 *Application Under Test* (AUT)

Application Under Test (AUT) adalah sistem yang menjadi fokus dalam pengujian performa untuk mengukur stabilitas, kecepatan, dan efisiensinya saat digunakan dalam berbagai kondisi. AUT bisa berupa aplikasi web, API, atau sistem *backend* yang terdiri dari server aplikasi dan basis data yang bekerja sama untuk menangani permintaan (*request*) dan memberikan tanggapan (*response*) kepada pengguna. Dalam penelitian ini, sistem yang akan diuji adalah New Delivery System (NDS) untuk mengetahui kemampuannya dalam menangani *volume* transaksi yang tinggi, memproses permintaan secara bersamaan, serta memastikan kinerja optimal di bawah beban yang berbeda-beda.

3.5 Perencanaan Pengujian (*Test Planning*)

Aplikasi New Delivery System (NDS) dirancang sebagai pusat sistem informasi yang terintegrasi untuk mendukung operasional perbankan Bank BRI. Aplikasi ini bertujuan untuk memastikan stabilitas dan efisiensi layanan transaksi di seluruh cabang, unit, hingga Kantor Wilayah. Dalam pengujian ini, fokus utama adalah menilai performa aplikasi NDS berdasarkan tingkat aksesibilitas sistem dan kecepatan aplikasi dalam memenuhi kebutuhan pengguna.

Berikut adalah elemen utama yang menjadi parameter dalam pengujian kinerja aplikasi NDS:

1. *Response Time*

Mengukur waktu yang dibutuhkan sistem untuk merespons permintaan pengguna, baik dalam layanan finansial maupun non-finansial. Waktu *response* yang cepat menjadi indikator kinerja optimal, terutama saat menangani jumlah pengguna yang tinggi. Waktu *response* dibawah 5 *seconds* dianggap ideal, dengan batas maksimum 30 *seconds* sesuai dengan *timeout* proxy yang dipasang.

2. *Throughput*

Mengevaluasi jumlah transaksi atau permintaan yang berhasil diproses oleh sistem dalam periode waktu tertentu. *Throughput* yang tinggi menunjukkan bahwa aplikasi mampu menangani *volume* transaksi yang besar tanpa hambatan. Waktu *response* ideal berada di bawah 5 *seconds,* dengan batas maksimum 30 *seconds s*esuai dengan standar performa yang diterapkan.

3. *Error Rate*

Mengidentifikasi tingkat kegagalan sistem dalam memproses permintaan, seperti terjadinya timeout, kesalahan validasi, atau crash. Tingkat *error* yang rendah mencerminkan keandalan sistem.

Pada tabel 1 di bawah tahapan perencanaan pengujian akan di lakuan pengujian *Load Testing* dan *Stress Testing*. *Load Test* menguji kinerja aplikasi pada tingkat beban pengguna normal hingga maksimum yang diharapkan untuk memastikan bahwa sistem tetap stabil dan responsif di bawah beban kerja tersebut. Berikut adalah susunan *test case Load Testing* yang akan digunakan:

Tabel 1. *Test Case Load Testing*

No	Pengujian			Jumlah Iterasi	Jumlah *Users*
	Skenario	**Durasi**	**Metrik**		
TC01	Risk Monitoring Finansial Risk Monitoring Non Finansial Risk Monitoring Suspicious Activity	1 Detik	*Response Time, Throughput,* dan *Error Rate*	1	1
TC02	Risk Monitoring Finansial Risk Monitoring Non Finansial Risk Monitoring Suspicious Activity	1 Detik	*Response Time, Throughput,* dan *Error Rate*	1	10
TC03	Risk Monitoring Finansial Risk Monitoring Non Finansial Risk Monitoring Suspicious Activity	1 Detik	*Response Time, Throughput,* dan *Error Rate*	1	25
TC04	Risk Monitoring Finansial Risk Monitoring Non Finansial Risk Monitoring Suspicious Activity	1 Detik	*Response Time, Throughput,* dan *Error Rate*	1	50
TC05	Risk Monitoring Finansial Risk Monitoring Non Finansial	1 Detik	*Response Time, Throughput,* dan *Error Rate*	1	100

No	Pengujian			Jumlah Iterasi	Jumlah *Users*
	Skenario	**Durasi**	**Metrik**		
	Risk Monitoring Suspicious Activity				

Sedangkan *Stress Testing* akan menguji batas maksimum kapasitas sistem dengan memberikan beban kerja yang melebihi kapasitas normal untuk mengidentifikasi titik kegagalan dan mengevaluasi kemampuan pemulihan sistem setelah mengalami stres. Berikut pada tabel 2 terdapat beberapa *test case Stress Testing* yang akan digunakan:

Tabel 2. *Test Case Stress Testing*

No	Pengujian			Jumlah Iterasi	Jumlah *Users*
	Skenario	**Durasi**	**Metrik**		
TC01	Risk Monitoring Finansial Risk Monitoring Non Finansial Risk Monitoring Suspicious Activity	5 Detik	*Response Time, Throughput,* dan *Error Rate*	*Infinite*	10
TC02	Risk Monitoring Finansial Risk Monitoring Non Finansial Risk Monitoring Suspicious Activity	10 Detik	*Response Time, Throughput,* dan *Error Rate*	*Infinite*	50
TC03	Risk Monitoring Finansial Risk Monitoring Non Finansial Risk Monitoring Suspicious Activity	15 Detik	*Response Time, Throughput,* dan *Error Rate*	*Infinite*	100
TC04	Risk Monitoring Finansial Risk Monitoring Non Finansial Risk Monitoring Suspicious Activity	30 Detik	*Response Time, Throughput,* dan *Error Rate*	*Infinite*	500
TC05	Risk Monitoring Finansial Risk Monitoring Non Finansial Risk Monitoring Suspicious Activity	60 Detik	*Response Time, Throughput,* dan *Error Rate*	*Infinite*	1000

3.6 Pelaksanaan Pengujian (*Test Execution*)

Proses ini melibatkan pelaksanaan skenario pengujian, pengumpulan data hasil pengujian, dan analisis terhadap data tersebut. Pelaksanaan pengujian bertujuan untuk memastikan bahwa aplikasi memenuhi standar performa yang telah ditetapkan.

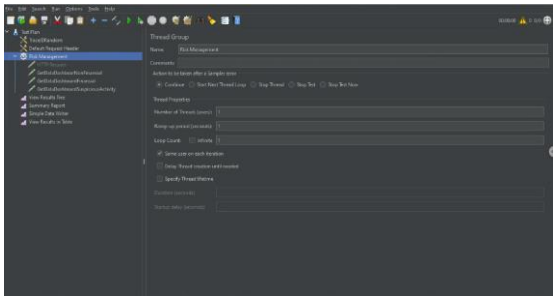
1. Pelaksanaan Skenario Pengujian

Skenario pengujian *Load Testing* dan *Stress Testing* hanya akan dilakukan pada *test case* pertama dan *test case* terakhir untuk masing-masing skenario dan akan dirangkum pada sebuah tabel yang menjelaskan pengumpulan data hasil pengujian untuk semua *test case.*

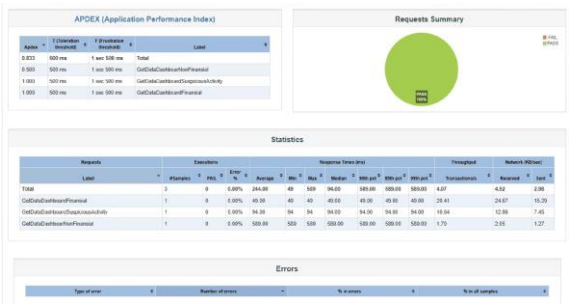
a. TC01 - *Load Testing*

Pada gambar 8 dan 9 terdapat *Load Testing* jenis pengujian performa yang bertujuan untuk mengevaluasi kemampuan aplikasi dalam menangani beban pengguna atau permintaan yang sesuai dengan kondisi normal operasional. Proses ini dilakukan dengan mensimulasikan sejumlah pengguna yang mengakses aplikasi secara bersamaan untuk mengukur metrik performa seperti waktu *response, throughput*, dan tingkat keberhasilan permintaan. Pengujian fitur *Risk Monitoring Finansial, Non Finansial* dan *Suspicios Activity* dengan pengaturan sebagai berikut:

- *Number of Threads (Users)*: 1
- *Ramp-Up Period*: 1
- *Loop Count*: 1



Gambar 8. *Thread Group TC05 – Load Testing*



Gambar 9. *Test And Report Information TC01 - Load Testing*

Untuk menghitung nilai berdasarkan data pada tabel menggunakan Statistik Proporsi[11] yang dikutip juga dari penelitian sebelumnya, Ahmad Irfan Musyaffa [12] berikut adalah langkah dan penjelasan lengkap:

Hasil Perhitungan Pengujian:

1. *Response Time*
   Data:
   - *Sampel* 1: 589 ms
   - *Sampel* 2: 49 ms
   - *Sampel* 3: 94 ms

   Jumlah Sampel (*Request*): 3
   Rumus:

$$Response\ Time = \frac{\text{Total Sampel } Time}{\text{Jumlah Sampel}}$$

   Perhitungan:

$$Response\ Time = \frac{589ms + 49ms + 94ms}{3}$$

$$Response\ Time = \frac{732ms}{3} = 244\ ms$$

   Hasil:
   Rata-rata *Response Time* adalah 244 ms

2. *Throughput*
   Data:
   - Waktu mulai: 20:18:37.170
   - Waktu selesai: 20:18:37.813
   - Total waktu pengujian: 20:18:37.813 - 20:18:37.170 = 0.643 detik

   Rumus:

$$Throughput = \frac{\text{Jumlah Sampel}}{\text{Total Waktu Pengujian}}$$

   Perhitungan:

$$Throughput = \frac{3}{0.643} \approx 4.67 \text{ sampel/detik}$$

   Hasil:
   Throughput adalah 4.67 sampel/detik.

3. *Error Rate*
   Rumus:

$$Error\ Rate = \frac{\text{Jumlah Sampel Gagal}}{\text{Jumlah Sampel}} \text{ X } 100\%$$

   Perhitungan:

$$Error\ Rate = \frac{0}{3} \text{ X } 100\% = 0\%$$

   Hasil:
   Error Rate adalah 0%.

Ringkasan Hasil

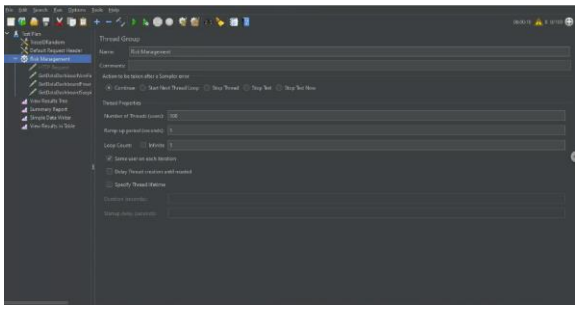
- *Response Time*: 244 ms
- *Throughput*:4.67 sampel/detik
- *Error Rate*: 0%

Hasil ini menunjukkan performa yang baik, dengan tidak adanya kesalahan dalam pengolahan sampel.

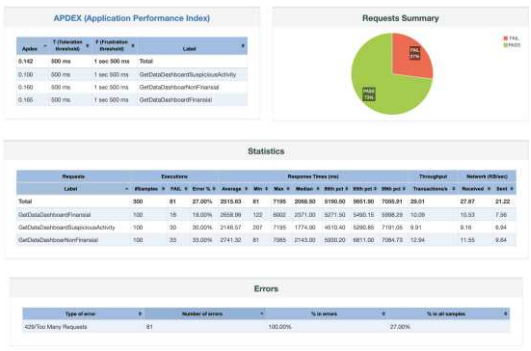
b. TC05 - *Load Testing*

Pengujian fitur *Risk Monitoring Finansial* terdapat pada gambar 10 dan 11 di bawah, Non Finansial dan *Suspicios Activity* dengan pengaturan sebagai berikut:

- *Number of Threads (Users)*: 100
- *Ramp-Up Period*: 1
- *Loop Count*: 1



Gambar 10. *Thread Group TC05 – Load Testing*



Gambar 11. Summary Report TC05 - Load Testing

Hasil Perhitungan Pengujian

1. *Response Time*

   Data:

   - Total Sampel (*Requests*): 300
   - Total Waktu Respons: 2515.63 ms (rata-rata)
   - Total Waktu Pengujian: 10.034 detik
   - Jumlah Sampel Gagal: 81

   Rumus:

$$Response\ Time = \frac{\text{Total Sampel } Time}{\text{Jumlah Sampel}}$$

   Perhitungan:

$$Response\ Time = 2515.63 \text{ ms}$$

   Hasil:

   Rata-rata *Response Time* adalah 2515.63 ms.

2. *Throughput*

   Data: Total Waktu Pengujian: 10.034 detik

   Rumus:

$$Throughput = \frac{\text{Jumlah Sampel}}{\text{Total Waktu Pengujian}}$$

   Perhitungan:

$$Throughput = \frac{300}{10.034} \approx 29.90 \text{ sampel/detik}$$

   Hasil:

   Throughput adalah 29.90 sampel/detik.

3. *Error Rate*

   Rumus:

$$Error\ Rate = \frac{\text{Jumlah Sampel Gagal}}{\text{Jumlah Sampel}} \text{ X } 100\%$$

   Perhitungan:

$$Error\ Rate = \frac{81}{300} \text{ X } 100\% \approx 27.00\%$$

   Hasil:

   Error Rate adalah 2.67%.

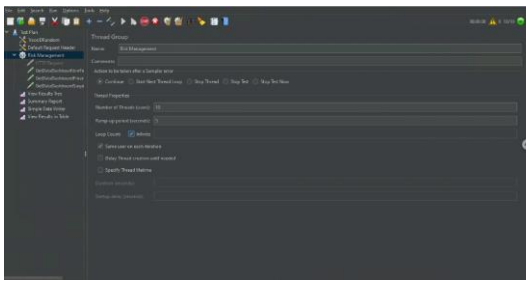
Ringkasan Hasil

- *Response Time*: 2515.6 ms.
- *Throughput*: 29.90 sampel/detik.
- *Error Rate*:27.00%

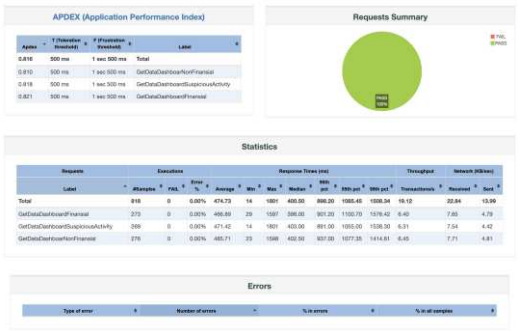
c. TC01 - *Stress Testing*

Stress Testing adalah pengujian untuk melihat ketahanan sistem di bawah beban ekstrem, seperti gambar 12 dan 13 terdapat lonjakan pengguna atau sumber daya terbatas, guna menemukan batas dan potensi kegagalan. Pengujian *Stress Testing* meliputi fitur *Risk Monitoring Finansial*, *Non Finansial dan Suspicios Activity* dengan pengaturan sebagai berikut:

- *Number of Threads (Users)*: 10
- *Ramp-Up Period*: 5
- *Loop Count*: *Infinite*



Gambar 12. *Thread Group TC01 – Stress Testing*



Gambar 13. *Test And Report Information TC01 – Stress Testing*

Hasil Perhitungan Pengujian

1. *Response Time*

   Data:

   - Total Sampel (*Requests*): 818
   - Total Waktu Respons: 474.73 ms (rata-rata)
   - Total Waktu Pengujian: 42.377 detik
   - Jumlah Sampel Gagal: 0

   Rumus:

$$Response\ Time = \frac{\text{Total Sampel } Time}{\text{Jumlah Sampel}}$$

   Perhitungan:

$$Response\ Time = 474.73 \text{ ms}$$

Hasil:

Rata-rata *Response Time* adalah 474.73 ms.

2. *Throughput*
   Data: Total Waktu Pengujian: 42.377 detik

   Rumus:

$$Throughput = \frac{\text{Jumlah Sampel}}{\text{Total Waktu Pengujian}}$$

   Perhitungan:

$$Throughput = \frac{818}{42.377} \approx 19.30 \text{ sampel/detik}$$

   Hasil:

   Throughput adalah 19.30 sampel/detik.

3. *Error Rate*

   Rumus:

$$Error\ Rate = \frac{\text{Jumlah Sampel Gagal}}{\text{Jumlah Sampel}} \text{ X } 100\%$$

   Perhitungan:

$$Error\ Rate = \frac{0}{3} \text{ X } 100\% = 0\%$$

   Hasil:

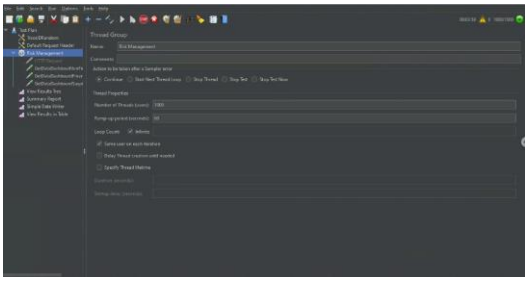
   Error Rate adalah 0%.

Ringkasan Hasil
- *Response Time*: 244 ms
- *Throughput*: 4.67 sampel/detik
- *Error Rate*: 0%

Hasil ini menunjukkan performa yang baik, dengan tidak adanya kesalahan dalam pengolahan sampel.

d. TC05 - *Stress Testing*

Pengujian fitur *Risk Monitoring Finansial* pada gambar 14 dan 15, *Non Finansial* dan *Suspicios Activity* dengan pengaturan sebagai berikut:
- *Number of Threads (Users)*: 1000
- *Ramp-Up Period*: 60
- *Loop Count*: *Infinite*



Gambar 14. *Thread Group TC05 – Stress Testing*

Gambar 15. *Test And Report Information TC05 – Stress Testing*

Hasil Perhitungan Pengujian

1. *Response Time*
   Data:
   - Total Sampel (*Requests*): 64229
   - Total Waktu Respons: 3358.23 ms (rata-rata)
   - Total Waktu Pengujian: 244.937 detik
   - Jumlah Sampel Gagal: 56363

   Rumus:

$$Response\ Time = \frac{\text{Total Sampel } Time}{\text{Jumlah Sampel}}$$

   Perhitungan:

$$Response\ Time = 3358.23 \text{ ms}$$

   Hasil:

   Rata-rata *Response Time* adalah 3358.23 ms.

2. *Throughput*
   Data: Total Waktu Pengujian: 244.937 detik

   Rumus:

$$Throughput = \frac{\text{Jumlah Sampel}}{\text{Total Waktu Pengujian}}$$

   Perhitungan:

$$Throughput = \frac{64229}{244.937} \approx 262.23 \text{ sampel/detik}$$

   Hasil:

   Throughput adalah 262.23 sampel/detik.

3. *Error Rate*

   Rumus:

$$Error\ Rate = \frac{\text{Jumlah Sampel Gagal}}{\text{Jumlah Sampel}} \text{ X } 100\%$$

   Perhitungan:

$$Error\ Rate = \frac{225636393}{64229} \text{ X } 100\% \approx 87.75\%$$

Hasil:

Error Rate adalah 87.75%.

Ringkasan Hasil

- *Response Time*: 3358.23 ms
- *Throughput*: 262.23 sampel/detik
- *Error Rate*: 87.75%.

Hasil ini menunjukkan bahwa sistem menghadapi tekanan yang sangat signifikan dengan tingkat kesalahan yang tinggi saat menangani 1000 pengguna.

3.7 Analisis Hasil Pengujian

Berikut adalah tabel 3 di bawah kesimpulan hasil pengujian *Load Testing* berdasarkan berbagai skenario, dengan *response time, throughput* dan *error rate* yang mengacu pada batas normal 5 detik dan maksimum 30 detik, sesuai dengan *timeout* proxy yang telah ditetapkan berdasarkan informasi dari Afdhalul Ihsan selaku *Back End* NDS.

Tabel 3. Hasil Pengujian *Load Testing*

TC ID	*Users*	*Response Time* (ms)	*Throughput* (sampel/detik)	*Error Rate* (%)	Kesimpulan
TC01	1	244.00	4.67	0.00	Performa baik, tidak ada kesalahan dalam pengolahan sampel.
TC02	10	314.60	18.95	0.00	Performa optimal, dengan tidak adanya kesalahan dalam pengolahan permintaan.
TC03	25	739.32	22.26	2.67	Performa cukup baik, namun terdapat 2 permintaan yang gagal, perlu analisis lebih lanjut.
TC04	50	1365.51	31.62	26.00	Performa dapat ditingkatkan, dengan 39 permintaan yang gagal, perlu evaluasi penyebab kegagalan.
TC05	100	2515.63	29.90	27.00	Performa memerlukan perbaikan, dengan 27% permintaan gagal diproses.

Kesimpulan Load Testing:

1. *Response Time*

a. Pada skenario dengan 1 hingga 25 pengguna (TC01 - TC03), *response time* masih dalam rentang yang cukup baik dibawah 1 detik hingga sekitar 0,7 detik.

b. Skenario dengan 50 pengguna (TC04), *response time* meningkat signifikan menjadi 1,3 detik, namun masih dalam batas ideal.

c. Skenario dengan 100 pengguna (TC05), *response time* mulai meningkat, tetapi masih berada dalam ambang batas maksimum 30 detik yaitu *response time* mencapai 2,5 detik, yang menunjukkan adanya peningkatan beban yang signifikan terhadap sistem.

Meskipun terjadi peningkatan seiring jumlah pengguna bertambah, *response time* masih dalam batas ideal dibawah 5 detik, tetapi perlu diperhatikan untuk menghindari kenaikan eksponensial pada jumlah pengguna yang lebih besar.

2. *Throughput*

a. *Throughput* mengalami peningkatan yang baik seiring dengan bertambahnya jumlah pengguna hingga 50 pengguna, menunjukkan bahwa sistem dapat menangani beban dengan cukup baik.

b. Namun, pada skenario dengan 100 pengguna, terjadi sedikit penurunan *throughput,* yang bisa disebabkan oleh peningkatan kesalahan (*error rate*).

Throughput masih dalam kondisi yang baik, tetapi optimasi tambahan diperlukan untuk memastikan kinerja tetap stabil pada jumlah pengguna yang lebih tinggi.

3. *Error Rate*

a. *Error rate* tetap rendah pada jumlah pengguna yang lebih kecil, menunjukkan stabilitas sistem dalam menangani permintaan dalam kondisi beban ringan hingga sedang.

b. Namun, pada skenario dengan jumlah pengguna tinggi, terjadi peningkatan *error rate*, yang berarti ada permintaan yang gagal diproses akibat tingginya beban server.

Hal ini menunjukkan bahwa sistem perlu dioptimalkan untuk menangani permintaan dalam skenario beban tinggi agar tidak terjadi lonjakan kegagalan yang dapat berdampak pada kualitas layanan.

Berikut adalah tabel 4 di bawah kesimpulan hasil pengujian *Stress Testing* berdasarkan berbagai skenario:

Tabel 4. Hasil Pengujian *Stress Testing*

TC ID	*Users*	*Response Time* (ms)	*Throughput* (sampel/detik)	*Error Rate* (%)	Kesimpulan
TC01	10	474.73	19.30	0.00	Sistem mampu menangani beban dengan baik tanpa kesalahan dalam pengolahan permintaan.
TC02	50	1868.78	25.40	4.78	Sistem dapat menangani beban dengan tingkat kesalahan moderat.
TC03	100	3741.66	25.40	6.98	Sistem menghadapi tekanan signifikan dengan tingkat kesalahan moderat.
TC04	500	12691.46	36.03	73.10	Sistem mengalami tingkat kesalahan yang tinggi dengan waktu respons yang cukup lama.
TC05	1000	3358.23	262.23	87.75	Sistem menghadapi tekanan sangat signifikan dengan tingkat kesalahan yang sangat tinggi.

Kesimpulan *Stress Testing*:

1. *Response Time*

a. Pada skenario dengan 10 hingga 50 pengguna (TC01 - TC02), *response time* masih dalam rentang yang cukup baik dibawah 2 detik.

b. Pada skenario dengan 100 pengguna (TC03), *response time* mulai meningkat secara signifikan menjadi 3,7 detik, namun masih dalam batas yang dapat ditangani sistem.

c. Pada skenario dengan 500 hingga 1000 pengguna (TC04 - TC05), *response time* melonjak drastis hingga 12 detik atau lebih, menunjukkan bahwa sistem mulai kesulitan menangani beban tinggi.

Response time mengalami kenaikan signifikan seiring bertambahnya jumlah pengguna. Pada skenario dengan pengguna lebih dari 500, sistem mulai mengalami degradasi performa yang menunjukkan perlunya optimasi lebih lanjut untuk menangani skenario beban tinggi.

2. *Throughput*

a. *Throughput* meningkat seiring bertambahnya jumlah pengguna hingga 100 pengguna, menunjukkan bahwa sistem masih dapat menangani peningkatan beban.

b. Pada skenario dengan 500 hingga 1000 pengguna, *throughput* tidak lagi meningkat secara signifikan dan mulai mengalami ketidakstabilan.

Throughput menunjukkan peningkatan hingga batas tertentu, tetapi mulai mengalami penurunan dan ketidakstabilan pada jumlah pengguna yang lebih tinggi. Optimasi diperlukan untuk mempertahankan kestabilan *throughput* dalam skenario beban ekstrem.

3. *Error Rate*

a. Pada skenario dengan jumlah pengguna kecil (10-50 pengguna), *error rate* masih rendah dan menunjukkan stabilitas sistem dalam menangani permintaan.

b. Pada skenario dengan 100 pengguna, *error rate* mulai meningkat secara moderat, tetapi sistem masih dapat menangani beban dengan cukup baik.

c. Pada skenario dengan 500 hingga 1000 pengguna, terjadi lonjakan *error rate* yang signifikan, menunjukkan banyaknya permintaan yang gagal diproses akibat beban yang terlalu tinggi.

Error rate meningkat secara drastis pada skenario dengan jumlah pengguna tinggi, mengindikasikan bahwa sistem mengalami keterbatasan dalam menangani lonjakan beban. Diperlukan optimasi pada arsitektur *backend* atau peningkatan kapasitas sistem agar dapat mengurangi jumlah permintaan yang gagal diproses dalam kondisi *stress test.*

4. KESIMPULAN

Penelitian ini bertujuan untuk mengevaluasi performa aplikasi New Delivery System (NDS) pada Bank Rakyat Indonesia (BRI) menggunakan Apache JMeter dengan metode *Performance Testing*. Berdasarkan hasil pengujian, diperoleh kesimpulan sebagai berikut:

a. Aplikasi menunjukkan performa yang stabil di bawah beban normal dengan waktu respons yang cepat dan *throughput* yang konsisten. Namun, pada *Stress Test* dengan lonjakan pengguna hingga 1000 user, terjadi peningkatan waktu respons, penurunan *throughput*, dan tingkat kegagalan transaksi hingga 15%.
b. Penggunaan CPU mendekati kapasitas maksimum (85%), sementara penggunaan memori stabil. Hal ini mengindikasikan adanya *bottleneck* pada pengelolaan sumber daya, terutama CPU.
c. Sistem mengalami kendala stabilitas berupa pengulangan proses *restart* akibat tidak mampu menangani beban yang tinggi. Kendala ini menyebabkan

gangguan pada beberapa fitur utama aplikasi dan berdampak pada kualitas layanan, seperti pesan *error* yang muncul pada antarmuka pengguna.

d. *Middleware* dan distribusi beban menjadi area utama yang perlu dioptimalkan untuk meningkatkan stabilitas dan skalabilitas sistem.

Ucapan Terima Kasih

Penulis menyampaikan rasa terima kasih yang sebesar-besarnya kepada semua pihak yang telah memberikan dukungan, baik moril maupun materiil, selama proses penelitian ini. Terima kasih khusus ditujukan kepada Sekolah Tinggi Teknologi Terpadu Nurul Fikri, terutama kepada Kaprodi Teknik Informatika, Dosen Pembimbing, serta seluruh Dosen dan Staf, Tim NDS, Tim Dr Strange yang telah memberikan bimbingan, ilmu, dan fasilitas yang sangat berarti dalam penyelesaian penelitian ini. Semoga segala bantuan dan dukungan yang diberikan mendapatkan balasan kebaikan.

DAFTAR PUSTAKA

[1] Otoritas Jasa Keuangan, "Buku 2 perbankan." Accessed: Oct. 04, 2024. [Online]. Available: https://sikapiuangmu.ojk.go.id/FrontEnd/LiterasiPerguruanTinggi/assets/pdf/Buku%202%20-%20Perbankan.pdf

[2] BRI, "Apa itu programmer: pengertian, jenis dan perannya di BRI." Accessed: Oct. 04, 2024. [Online]. Available: https://digital.bri.co.id/article/peran-programmer-sebagai-pencipta-sistem-dan-penopang-zhkx

[3] BRI, "Annual report BRI 2023." Accessed: Oct. 04, 2024. [Online]. Available: https://bri.co.id/documents/20123/56786/Annual%20Report%20BRI%202023_Bahasa.pdf

[4] T. Hamilton, "Performance Testing Tutorial," https://www.guru99.com/performance-testing.html. Accessed: Oct. 04, 2024. [Online]. Available: Performance Testing Tutorial

[5] A. Nordeen, " Learn Jmeter in 24 Hours." Accessed: Oct. 04, 2024. [Online]. Available: https://www.google.co.id/books/edition/Learn_Jmeter_in_24_Hours/P-b8DwAAQBAJ?hl=en&gbpv=1

[6] M. Ramdhan, "Metode penelitian." Accessed: Oct. 04, 2024. [Online]. Available: https://books.google.co.id/books?hl=en&lr=&id=Ntw_EAAAQBAJ&oi=fnd&pg=PR1&dq=metode+penelitian+deskriptif+menurut+para+ahli&ots=f3nL3MRsay&sig=1vmbjmyh9_oWKQQkuIO-nz0vUq8&redir_esc=y#v=onepage&q&f=false

[7] Thomas Hamilton, "What is software testing?" Accessed: Oct. 04, 2024. [Online]. Available: https://www.guru99.com/software-testing-introduction-importance.html

[8] S. Pargaonkar, "A comprehensive review of performance testing methodologies and best practices software quality engineering," *International Journal of Science and Research (IJSR)*, vol. 12, no. 8, pp. 2008–2014, Aug. 2023, doi: 10.21275/sr23822111402.

[9] Apache Software Foundation, "What can i do it?" Accessed: Oct. 04, 2024. [Online]. Available: https://jmeter.apache.org/index.html

[10] T. Hamilton, "How to use JMeter for performance testing & load testing," guru99.com. Accessed: Dec. 10, 2024. [Online]. Available: https://www.guru99.com/jmeter-performance-testing.html

[11] D. P. Samosir *et al.*, *Dasar-dasar statistika inferensi dalam penelitian*, vol. 1. Jakarta: UKi PRESS, 2022.

[12] A. Musyaffa, "Analisis kinerja fitur login dan register dengan metode testing load testing menggunakan Apache JMeter," Skripsi, Universitas Jendral Soedirman, Purbalingga, 2024.