Performance Testing Menggunakan Metode Load Testing dan Stress Testing pada Sistem Core Banking PT. XYZ

1st Dzaki Madhani
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
dzakimadhani@students.telkomuniversity.ac.id

2nd Eko Darwiyanto
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
ekodarwiyanto@telkomuniversity.ac.id,

3rd Arfive Gandhi
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
arfivegandhi@telkomuniversity.ac.id

***Abstrak* — Core banking adalah aplikasi inti dari sistem perbankan yang berfungsi untuk mengelola aktivitas transaction, customer information file, dan berbagai layanan untuk nasabah perbankan lainnya. Oleh karena itu penting untuk dilakukan performance testing agar dapat dipastikan bahwa sistem core banking tetap dapat berjalan dengan baik ketika jumlah aktivitas nasabah di perbankan tersebut meningkat. Performance testing merupakan salah satu pengujian non fungsional pada perangkat lunak yang bertujuan untuk mengukur kehandalan suatu aplikasi seperti waktu respon, transaksi per detik, dan kestabilan akses dari sistem dan aplikasi dibawah beban kerja yang diberikan. Ada beberapa metode dalam performance testing yang akan digunakan yaitu seperti load testing dan stress testing. Load testing dilakukan untuk menguji aplikasi dibawah beban yang diharapkan. Sebelum dijalankan load testing akan dijalankan stress testing terlebih dahulu untuk mendapatkan nilai maksimum beban yang dapat ditangani oleh aplikasi yang kemudian akan digunakan untuk menjalankan load testing. Setelah dilakukan eksekusi performance test, ditemukan hasil yang dapat memenuhi semua objektifnya. Dari hasil yang didapatkan menujukkan bahwa sistem core banking pada PT. XYZ cukup stabil untuk menangani beban yang diharapkan.**

***Kata kunci*—performance test, load testing, stress testing, loadrunner, core banking**

I. PENDAHULUAN

A. Latar Belakang

Setiap perbankan memiliki sistem core banking yang merupakan sebuah aplikasi inti dan kunci dari sebuah perbankan. Sistem tersebut berfungsi untuk mengelola aktivitas-aktivitas inti perbankan yang meliputi transaction, saving, loan, dan customer information file [1]. Aktivitas inti tersebut erat hubungannya dengan nasabah bank yang secara tidak langsung merupakan user yang mengakses core banking tersebut. Terdapat momen di setiap tahun dalam satu hari di mana jumlah pengguna yang mengakses core banking secara bersamaan meningkat berkali-kali lipat. Pada PT. XYZ sedang melakukan pengembangan mobile banking versi terbaru untuk menggantikan mobile banking versi lama mereka. Pengembangan mobile banking baru tersebut juga diharapkan dapat menangani kenaikan jumlah nasabah setiap tahunnya pada PT. XYZ. Akan menjadi hal yang berbahaya jika sistem tersebut menjadi down karena akan dapat mengurangi tingkat kepercayaan nasabah terhadap bank nya dan dapat mengakibatkan penurunan jumlah nasabah pada bank tersebut. Hal tersebut mengharuskan sistem core banking tetap berjalan dengan baik di saat itu.

Untuk memastikan sistem core banking tersebut dapat berjalan dengan baik maka perlu dilakukan pengujian performa (performance testing). Performance testing merupakan pengujian yang berfungsi untuk mengetaui seberapa jauh performa dari sebuah aplikasi ketika bejalan di bawah beban kerja yang diharapkan [2]. Hasil performance testing dapat dikatakan baik apabila dapat mencapai ekspektasi dari pengguna aplikasi atau sistem yang diuji. Tidak ada patokan yang pasti untuk menentukan hasil yang baik pada performance test. Setiap aplikasi memiliki karakteristiknya masing-masing dan baik tidak hasilnya bergantung pada kebutuhan bisnis yang ada pada aplikasi tersebut.

Pada performance testing dapat dilakukan dengan bermacam metode pengujian, diantara yaitu stress testing, load testing, endurance testing, spike testing, dan lain sebagainya. Adapun metode yang akan digunakan untuk melakukan performance testing yaitu stress testing dan load testing. Metode tersebut dipilih karena pada load testing dapat digunakan untuk mengetahui apakah sistem core banking PT. XYZ dapat menangani trafik sesuai dengan ekspektasi untuk menghadapi momen kenaikan nasabah setiap tahunnya. Sedangkan dari hasil metode stress testing akan dapat membantu menemukan jumlah maksimum pengguna berjalan bersamaan (concurrent user) dan juga untuk menghindari terjadinya kegagalan sistem ketika didorong sampai kondisi puncaknya.

Dalam menjalankan pengujian tersebut perlu adanya performance testing tools salah satunya adalah Loadrunner. Loadrunner merupakan performance testing tools yang dapat mendukung stress testing maupun load testing dan menyertakan fungsionalitas untuk memudahkan dalam menganalisis hasil dari pengujian yang sudah dijalankan. Di Loadrunner beban kerja user akan ditentukan oleh virtual user yaitu concurrent user yang akan mengirimkan request ke aplikasi yang akan dilakukan pengujian [3].

Performance test pada core banking PT. XYZ sudah pernah dilakukan sebelumnya di environment dengan spesifikasi dan arsitektur server yang berbeda dengan environment

production. Environment production merupakan versi sistem core banking yang saat ini digunakan untuk keperluan aktivitas nasabah yang ada di PT. XYZ. Sehingga PT. XYZ menyediakan environment core banking baru yang disebut dengan environment QA untuk keperluan performance testing ini dengan spesifikasi dan arsitektur server yang lebih menyerupai dengan versi production. Karena jika dilakukan performance test di environment yang sama lagi maka yang terjadi adalah hasil dari performa aplikasi tidak akan menunjukkan hasil yang mencerminkan versi production. Hal tersebut akan membuat aplikasi tersebut belum diketahui hasil dari performanya dan tidak terlacak jika ada sumber permasalahan dalam performanya. Dengan spesifikasi dan arsitektur yang menyerupai versi production, maka hasil yang akan didapatkan juga akan menggambarkan performa core banking yang saat ini sedang digunakan untuk mengelola aktivitas nasabah pada bank tersebut, karena performance test yang dilakukan sebelumnya pada core banking tersebut memiliki kondisi spesifikasi dan arsitektur yang jauh berbeda dengan versi production.

Maka dari itu penelitian ini akan melakukan performance testing dengan menggunakan metode stress testing dan load testing pada sistem core banking sebagai tools yang akan dipakai.

B. Rumusan Masalah

Berdasarkan latar belakang yang telah disampaikan maka didapatkan rumusan masalah sebagai berikut.

Bagaimana cara penerapan performance testing menggunakan metode load testing dan stress testing pada sistem core banking PT. XYZ?

Bagaimana pencapaian hasil performance testing terhadap objektif yang telah ditentukan?

C. Batasan Masalah

Adapun ruang lingkup batasan masalah pada penelitian ini yaitu sebagai berikut.

Performance testing yang akan dilakukan menggunakan metode load testing dan stress testing.

Tools yang akan digunakan untuk pengujian ini adalah Loadrunner.

Parameter yang akan dinilai dalam pengujian ini yaitu response time, transactions per second (TPS), dan error rate.

D. Tujuan

Berdasarkan rumusan masalah yang ada di atas, maka tujuan dari penelitian ini sebagai berikut.

Melakukan load testing dan stress testing pada core banking menggunakan tools Loadrunner.

Menganalisis hasil performance testing yang sudah dilakukan.

II. KAJIAN TEORI

A. Pengujian Perangkat Lunak

Pengujian perangkat lunak merupakan proses untuk mengevaluasi dan memverifikasi perangkat lunak sudah sesuai dengan yang diharapkan, mencegah adanya *bug* ataupun *error*, dan juga memastikan bebas dari *defect*. Ada banyak jenis pengujian perangkat lunak, diantaranya adalah *functional testing* dan *non-functional testing*.

1. Functional Testing

Functional testing adalah pengujian yang menguji fungsi-fungsi yang ada pada sebuah perangkat lunak. Perangkat lunak akan diuji dengan memberikan *input* data yang kemudian akan dicek hasil *output*-nya [4].

2. *Non-Functional Testing*

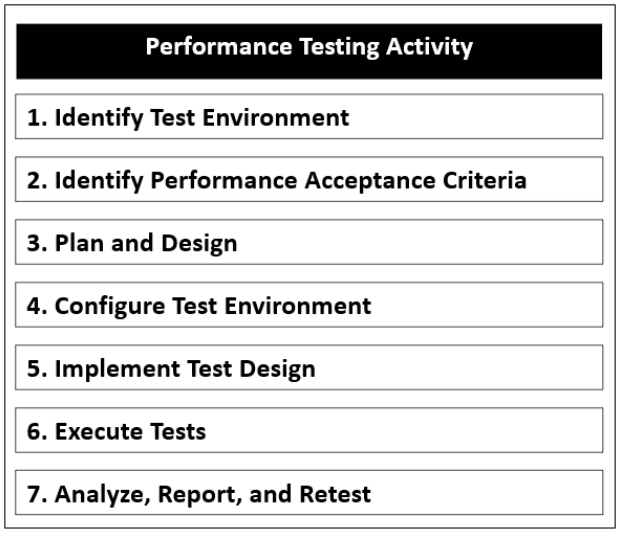
Non-functional seperti namanya, merupakan *requirement* yang tidak secara langsung terkait dengan spesifik *services* yang diberikan oleh sistem kepada penggunanya. Non-functional biasanya tidak mementingkan fungsional individu perangkat lunak, tetapi menggambarkan seberapa bagus suatu perangkat lunak atau sistem terhadap *reliability* dan *response time*. Beberapa yang termasuk pengujian non-functional yaitu *performance, security, availability* dan lain-lainya [5].

B. *Performance Testing*

Performance Test merupakan salah satu *non-fuctional testing* yang menentukan atau memverifikasi karakteristik dari *speed*, *scalability*, dan *stability* pada aplikasi yang sedang diuji di bawah beban kerja tertentu [4]. Berikut hal-hal yang dapat dicapai ketika dilakukan *performance testing*.

- Menilai kesiapan produksi
- Mengevaluasi terhadap kriteria performa
- Membandingkan karakteristik performa dari beberapa sistem atau konfigurasi sistem
- Menemukan sumber suatu permasalahan performa
- Support perbaikan peningkatan performa
- Menemukan level *throughput*

Jadi *performance testing* biasanya dilakukan untuk membantu mengidentifikasi *bottlenecks* dalam sebuah sistem, menjadi acuan dasar untuk pengujian di masa mendatang, mendukung perbaikan peningkatan performa, dan mengumpulkan data terkait performa untuk membantu membuat keputusan yang tepat terkait dengan kualitas keseluruhan dari aplikasi yang sedang diuji. Selain itu, hasil dari pengujian dan analisis *performace test* dapat membantu untuk memperkirakan konfigurasi *hardware* yang diperlukan untuk mendukung sebuah aplikasi yang berencana untuk "*go live*" ke versi *production*. Berikut beberapa proses aktivitas yang perlu dilakukan dalam *performance testing* [6].



GAMBAR **Error! No text of specified style in document.**-1
Aktivitas *Performance Testing* [6]

Hal pertama yang perlu dilakukan sebelum menjalankan *performance testing* yaitu mengidentifikasi lingkungan (*environment*) dari aplikasi yang akan diuji. Memahami lingkungan sistem yang akan diuji akan membantu perencanaan pengujian yang lebih efisien. Kemudian setelah

mengidentifikasi lingkungan sistem yang akan diuji, hal selanjutnya adalah menentukan kriteria-kriteria yang akan diuji untuk menjadi ketentuan penilaian yang perlu dicapai dalam pengujian ini. Kriteria yang dimaksud meliputi *response time*, *transactions per second* (TPS), *error rate*.

Setelah selesai mengidentifikasi hal-hal di atas, maka bisa dimulai untuk membuat perencanaan dan desain pengujian (*plan and design*) dari hal-hal yang sudah diidentifikasi. Beberapa yang perlu diidentifikasi dari perencanaan tersebut adalah membuat *script*, skenario, dan menentukan data pengujian. Dari perencanaan tersebut akan diimplementasikan, dijalankan, dan dianalisis. Untuk mengimplementasikan maka perlu mulai dipersiapkan komponen-komponen untuk melakukan pengujian. Setelah setiap komponen siap, maka pengujian bisa mulai dieskekusi atau dijalankan.

Proses terakhir dalam pengujian ini adalah menganalisa hasil pengujian yang telah dieksekusi dan membuat laporan pengujian. Kemudian lakukan pengujian ulang jika diperlukan untuk memvalidasi hasil dari pengujian yang sebelumnya.

1. Stress Testing

Stress test adalah jenis dari *performance test* yang dirancang untuk mengevaluasi *behavior* aplikasi saat didorong melampaui kondisi normalnya atau kondisi puncaknya [6]. Tujuan dari *stress testing* adalah untuk menemukan *bug* aplikasi yang hanya muncul ketika beban aplikasi sedang dalam kondisi yang tinggi. *Bug* tersebut dapat mencakup hal-hal seperti masalah sinkronisasi, *race conditions*, dan kebocoran memori. *Stress test* juga memungkinkan untuk mengindentifikasi titik lemah dari sebuah aplikasi dan juga menunjukkan bagaimana aplikasi kondisi aplikasi di bawah tekanan beban yang ekstrem.

2. Load Testing

Load test merupakan salah satu jenis *performance test* di mana sistem disimulasikan untuk menangani beban yang sesuai dengan beban yang dapat ditangani oleh aplikasi yang diuji [6]. Untuk dapat menentukan beban yang sesuai salah satu caranya adalah dapat memanfaatkan hasil dari *stress test*. Dari hasil *stress test* akan dianalisa untuk mengetahui beban maksimal yang dapat ditangani aplikasi tersebut. Kemudian dari hasil tersebut dapat dipakai untuk kebutuhan *load test*.

C. Core Banking

Sistem *core banking* adalah aplikasi inti dan merupakan jantung dari sebuah perbankan [7]. *Core banking* berfungsi untuk memproses transaksi perbankan sehari-hari yang mencakup fungsi nasabah, simpanan, pinjaman, akuntansi, dan pelaporan. Aplikasi *core banking* pada PT. XYZ merupakan aplikasi berbasis *microservice*. *Microservice* merupakan salah satu contoh arsitektur perangkat lunak yang memecah aplikasi menjadi beberapa bagian kecil dan aplikasi kecil tersebut adalah *service*. Pada aplikasi *core banking* tersebut memiliki beberapa *service* API yang di mana di setiap *service* nya merupakan fungsionalitas dari aplikasi tersebut. Dari *service* yang ada tersebut akan dibagi menjadi 2 jenis fungsionalitas yaitu:

1. Transaksional

Fungsionalitas transaksional akan mewakili *service core banking* yang mengelola segala jenis transaksi yang ada di bank PT. XYZ seperti *service* untuk mengatur aktivitas penarikan uang, penyetoran yang, dan lain-lain. Beberapa *services* transaksional yang ada pada *core banking* PT. XYZ meliputi *fund transfers*, *cash transaction*, *reversal funds transfer*, *web service SKN*, dan lain-lain.

2. Non-transaksional.

Pada fungsionalitas non-transaksional akan mewakili *service* selain yang berhubungan dengan transaksi yang ada di bank PT. XYZ seperti *service* yang mengelola informasi nasabah yang meliputi informasi saldo, portofolio, pembukaan rekening, dan lain-lain. Beberapa *services* non-transaksional yang ada pada *core banking* PT. XYZ meliputi *enquiry account*, *account details*, *customer portofolio*, *enquiry portofolio*, *open account*, dan lain-lain.

D. Loadrunner

Loadrunner adalah sebuah alat *performance test* yang dapat memperkirakan perilaku maupun properti sebuah sistem dan menemukan permasalahan melalui sejumlah simulasi pengguna yang akan melakukan hit ke lingkungan aplikasi yang akan diuji. Pada alat pengujian ini juga dapat melakukan *monitoring* untuk melihat utilisasi *server* dari aplikasi yang sedang diuji. Dengan menggunakan Loadrunner maka akan dapat mempersingkat waktu pengujian, mengoptimalkan kinerja pengujian, dan mempercepat siklus peluncuran sistem aplikasi yang telah selesai dibangun [8]. Ada 3 komponen yang ada pada Loadrunner, yaitu *Virtual Generator* (VuGen), *Controller*, *Analysis*.

Pada Loadrunner memiliki langkah-langkah penggunaan yang dapat mendukung aktivitas *performance testing* yang sudah dijelaskan di atas. Berikut langkah-langkah pada Loadrunner.

- *Planning the test*
- *Creating the vuser script*
- *Creating the scenario*
- *Running the scenario*
- *Analyzing test results*

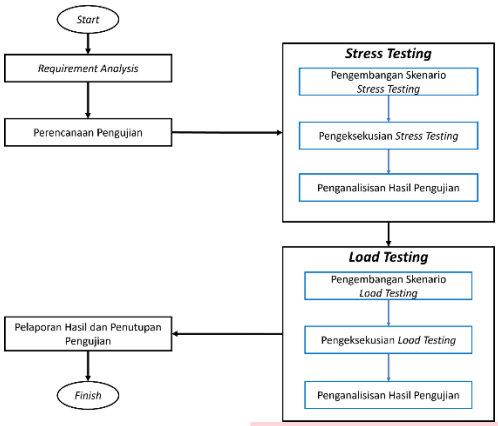
Selama bagian *planning* semua tujuan atau *objectives* pengujian perlu dipastikan dapat tercapai agar mendapatkan hasil yang lebih baik. Kemudian langkah selanjutnya adalah pembuatan *script testing*. Pembuatan *script testing* tersebut dapat dilakukan di komponen *Virtual Generator* di mana semua pengaturan *script* akan diatur di komponen tersebut. Setelah menyelesaikan pembuatan *script*, maka langkah selanjutnya adalah mempersiapkan skenario pengujian yang dapat diatur di komponen yang bernama *Controller*. Pada *Controller* dapat diatur skenario yang meliputi banyaknya concurrent user pengujian, durasi pengujian, dan lain-lain. Pada skenario yang telah dibuat tersebut akan menentukan apakah pengujian tersebut termasuk dalam metode *stress test* atau *load test*. Setelah skenario selesai dibuat maka pengujian dapat dijalankan. Selama pengujian berjalan, Loadrunner akan mengumpulkan data-data pengujian tersebut dan akan dijadikan hasil dari pengujian yang dapat dilihat pada komponen *Analysis*.

III. METODE

A. Alur Penelitian

Pada penelitian ini akan dilakukan *performance testing* pada *core banking* PT. XYZ untuk dapat diukur seberapa jauh aplikasi tersebut dapat berjalan di bawah beban kinerja yang diharapkan. Alur penelitian perlu dibuat agar aktivitas *performance testing* dapat berjalan dengan baik dan akan digunakan sebagai kerangka aktivitas penelitian ini. Pada alur

penelitian ini akan mengadaptasi *Software Testing Life Cycle* (STLC).



GAMBAR **Error! No text of specified style in document.**-2
Alur Penelitian

B. Requirement Analysis

Pada tahap pertama *Software Testing Life Cycle* (STLC) yaitu *requirement analysis* di mana pada tahap ini penguji perlu mengetahui serta memahami kebutuhan bidang yang akan diuji dan persyaratan dari pengujian [9]. Tahap pertama perlu dilakukan koordinasi dengan tim *developer* maupun *user* dari perangkat lunak yang akan diuji.

1. Pengumpulan Data

Sebelum melakukan pengujian performa dalam penelitian ini diperlukan aktivitas pengumpulan data dari pihak PT. XYZ untuk menentukan perencanaan dalam pengujian ini. Data didapatkan dengan melakukan wawancara untuk memperoleh informasi langsung dari pihak PT. XYZ. Proses wawancara dilakukan dengan mengajukan pertanyaan yang dapat dilihat pada lampiran 1.

2. Identifikasi Bisnis Proses

Pada umumnya sistem *core banking* terdiri dari beberapa fungsi atau modul yang terintegrasi, antara lain yaitu modul kredit (pinjaman), modul dana (deposito), modul akuntansi (buku besar), modul pengiriman uang, dan sebagainya [10]. Fitur pada *core banking* PT. XYZ digambarkan nama *services* pada *core banking* itu sendiri. Setiap perbankan memiliki nama *service* yang berbeda di setiap *core banking*-nya, seperti *services* milik *core banking* PT. XYZ dapat dilihat pada lampiran 2. Setiap *service* pada *core banking* PT. XYZ dapat mengelola lebih dari satu fitur pada aplikasi *mobile banking* mereka.

3. Identifikasi Lingkungan Pengujian

Informasi *environment* aplikasi yang akan dijalankan diperlukan untuk menjalankan *performance test*. Jika tujuan pengujian ini adalah untuk melihat karakteristik kinerja aplikasi dalam *environment production* maka *environment* pengujian perlu dibuat untuk menyerupai versi *production*-nya. Faktor kunci dalam mengidentifikasi lingkungan pengujian adalah memahami sepenuhnya persamaan dan perbedaan antara lingkungan pengujian dan versi *production*. Berikut beberapa faktor yang perlu diperhatikan.

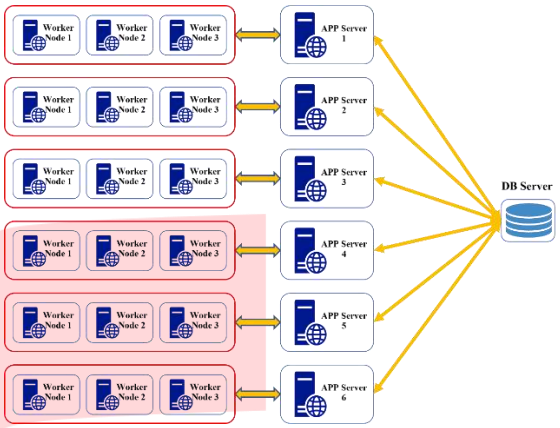
a. Hardware
- Konfigurasi
- Spesifikasi (CPU, RAM, dll.)

b. Software
- Tipe server yang digunakan
- Kapasitas Penyimpanan

c. Faktor Eksternal
- Interaksi dengan sistem lain
- *Traffic* pada jaringan

4. Arsitektur Aplikasi

Berikut merupakan arsitektur dari *core banking* dari PT. XYZ berdasarkan data yang didapatkan.



GAMBAR **Error! No text of specified style in document.**-3
Arsitektur Aplikasi yang Akan Diuji

5. Spesifikasi Server Aplikasi

TABEL **Error! No text of specified style in document.**-1
Spesifikasi Server Aplikasi

No	Server	*IP Address*	Konfigurasi *Hardware*			*OS* (*+version*)
			CPU (*Core*)	*Memory* (GB)	*Disk* (GB)	
1	APP Server 1	10.0.19.153	80	1.536	556	Windows Server 2012 R2
2	APP Server 2	10.0.19.154	80	1.536	556	Windows Server 2012 R2
3	APP Server 3	10.0.19.156	44	512	556	Windows Server 2012 R2
4	APP Server 4	10.0.19.159	24	120	698	Windows Server 2012 R2
5	APP Server 5	10.0.19.160	24	120	698	Windows Server 2012 R2
6	APP Server 7	10.0.19.161	24	120	698	Windows Server 2012 R2

6. Identifikasi *Acceptance Criteria*

Untuk mengetahi performa dari aplikasi *core banking* ini dapat dengan mengamati perilaku aplikasi di bawah beban

pengujian. Berikut merupakan parameter objektif yang ingin dicapai dalam *performance test* kali ini.

TABEL **Error! No text of specified style in document.**-2
Parameter Objektif Pengujian

No.	Parameter
1.	*Error rate*
2.	*Transactions Per Second* (TPS)
3.	*Response time*

7. Pengembangan *Script* Pengujian

Berikut merupakan bisnis proses/ fitur yang diidentifikasi untuk *performance testing* berdasarkan pembagian 2 jenis fungsionalitas, transaksional dan non-transaksional. Proses bisnis ini telah mengambil bagian dalam skenario *performance test*.

TABEL **Error! No text of specified style in document.**-3
Bisnis Proses *Core Banking*

ID Fitur	Fitur	Deskripsi	Jenis Fungsionalitas
01	*Enquiry Account*	Pada fitur ini berfungsi untuk menampilkan informasi nasabah yang di antaranya adalah nama pemilik *account*, nomor *customer*, nomor rekening nasabah, kantor cabang *account*, isi saldo rekening, dan lain-lain.	Non-Transaksional
02	*Fund Transfer*	Pada fitur ini digunakan untuk mengatur transaksi transfer saldo nasabah sesama bank.	Transaksional

Dari masalah yang ada pada PT. XYZ yang sedang mengembangkan dan mempersiapkan peluncuran aplikasi *mobile banking* mereka, maka akan diambil kedua *services* pada *core banking* tersebut yaitu *fund transfer* dan *enquiry account*. Berdasarkan data transaksi harian pada *mobile banking* milik PT. XYZ yang dapat dilihat pada lampiran 3, transfer pinbuk (transfer antar bank PT. XYZ) merupakan transaksi terbanyak yang dilakukan oleh nasabah PT. XYZ. *Fund transfers* merupakan *service core banking* untuk mengatur transfer transfer antar bank PT. XYZ. Sedangkan pada fitur non-transaksional diambil *enquiry account* sebagai *service* yang mengatur beberapa fitur seperti cek saldo, nomor *customer* bank, dan fitur-fitur non-transaksional lainnya. Sehingga diambil 2 *service* atau fitur tersebut untuk dilakukan *performance testing*.

Untuk melakukan pengujian perlu dibuat *script* masing-masing dari 2 fitur di atas. Pengembangan *script* akan dilakukan di salah satu komponen yang dimiliki Loadrunner yaitu komponen VuGen (*Virtual Generator*). Untuk membuat *script* tersebut dibutuhkan *API Request* dari 2 fitur di atas yang dapat di lihat pada lampiran 3.

C. Perencanaan Pengujian

Bagian ini akan berisi desain skenario *stress test* dan *load test* yang akan digunakan untuk *performance testing core banking* pada PT. XYZ untuk menjalankan semua proses bisnis.

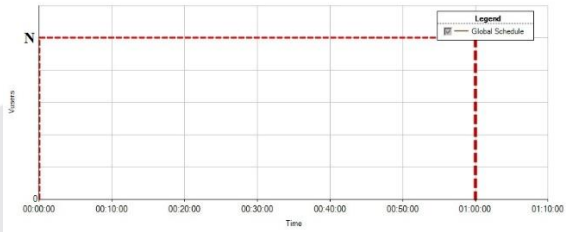
1. Desain Skenario *Stress Test*

GAMBAR **Error! No text of specified style in document.**-4
Skenario *Stress Test*

a. 2 Proses Bisnis/ Fitur akan digunakan untuk *stress test*.
b. Meningkat N *virtual users* setiap 5 menit sampai NMax *virtual users*, lalu jalankan tes selama 5 menit. Kemudian hentikan semua *virtual users* secara bersamaan. Nmax = jumlah maksimal *virtual users*. N = jumlah kenaikan *virtual users* secara konstan.
c. Hasil akan dikumpulkan pada setiap peningkatan N *virtual users* sampai NMax.
d. Distribusi beban *virtual user* yang digunakan untuk semua skenario adalah sebagai berikut:

Fitur ID	Fitur	Distribusi beban (%)
01	*Enquiry Account*	50 %
02	*Fund Transfer*	50 %

2. Desain Skenario *Load Test*



Gambar **Error! No text of specified style in document.**-5
Skenario *Load Test*

a. 2 Proses Bisnis/ Fitur akan digunakan untuk *load test*.
b. Mulai N *virtual users* secara bersamaan lalu jalankan selama 2 jam, lalu hentikan semua vuser secara bersamaan. N = jumlah beban *virtual users* maksimal yang dapat ditangani oleh aplikasi.
c. Hasil akan dikumpulkan dari awal sampai akhir pengujian.
d. Distribusi beban *virtual user* yang digunakan untuk semua skenario adalah sebagai berikut:

Fitur ID	Fitur	Distribusi beban (%)
01	*Enquiry Account*	50 %
02	*Fund Transfer*	50

IV. HASIL DAN PEMBAHASAN

A. Objektif Pengujian

Untuk menentukan apakah sistem atau aplikasi yang akan diuji dapat dikatakan berjalan baik maka perlu diidentifikasi objektif nya dari sisi bisnis. Pada *core banking* PT. XZY didapatkan objektif yang sudah ditentukan sebagai berikut.

TABEL **Error! No text of specified style in document.**-4
Objektif *Performance Test*

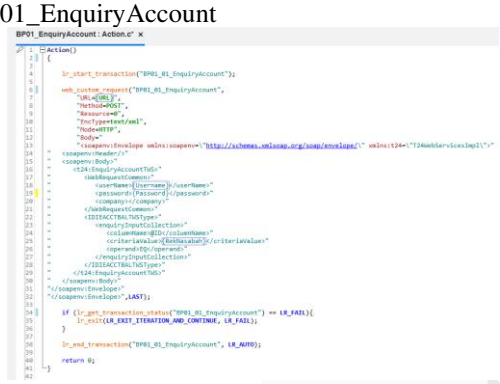
No.	Parameter	Objektif
1.	*Error rate*	<= 1%
2.	*Transactions Per Second* (TPS)	> 1000 TPS
3.	*Response time*	< 2,5 detik

B. Implementasi Desain Pengujian

1. Script Pengujian Menggunakan Loadrunner

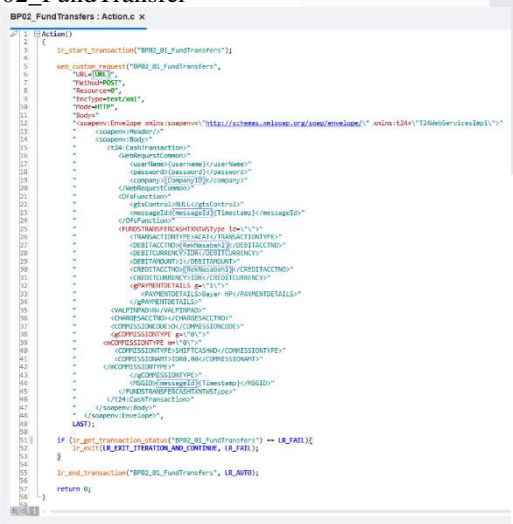
Terdapat 2 Fitur yang akan dibuat *script* untuk *performance testing* menggunakan komponen *Loadrunner* yang bernama VuGen (*Virtual Generator*). Fitur tersebut adalah *Enquiry Account* dan *Fund Transfer* didapatkan *API Request* dari masing-masing untuk diproses menjadi *script* pengujian. Berikut merupakan gambaran *script* yang sudah dibuat dari masing-masing fitur

a. BP01_EnquiryAccount



GAMBAR **Error! No text of specified style in document.**-6
Script Enquiry Account

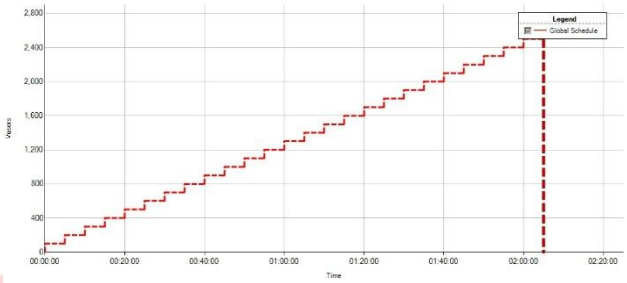
b. BP02_FundTransfer



GAMBAR **Error! No text of specified style in document.**-7
Script Fund Transfers

2. Skenario Pengujian

Dalam penelitian kali ini perlu dilakukan *stress testing* terlebih dahulu untuk mengetahui untuk mendapatkan hasil maksimal *load* yang dapat diatasi oleh sistem *core banking* ini. Maka berikut merupakan skenario *stress test* yang perlu dilakukan terlebih dahulu.



GAMBAR **Error! No text of specified style in document.**-8
Skenario *Stress Test* 2500 *Users*

Meningkat 100 *virtual users* setiap 5 menit sampai 2.500 *virtual users*, lalu jalankan tes selama 5 menit. Kemudian hentikan semua *virtual users* secara bersamaan. Setelah dilakukan eksekusi pengujian pada skenario *stress test* ini akan dibuat skenario untuk melakukan *load test*.

C. Eksekusi Pengujian

Stress *Testing*

Hasil Rangkuman Pengujian

Setelah dilakukan eksekusi *stress test* menggunakan Loadrunner, didapatkan data hasil pengujian dalam bentuk tabel dan grafik dari komponen Loadrunner yang bernama Analysis. Pada *stress test* ini akan ditampilkan hasil di setiap kenaikan user sampai *max concurrent user*s yaitu 2.500 *user*. Jadi terdapat 25 hasil *stress test* yang menggambarkan hasil pada setiap kenaikan *user* pada proses *stress test* dari 100 user sampai 2.500 *user*. Berikut hasil dari *stress test* yang telah dilakukan.

TABEL **Error! No text of specified style in document.**-5
Hasil *Stress Testing*

Load User	Fitur	Response Time (detik)			Jumlah Hit			Error Rate	TPS
		Min	Avg	Max	Passed	Failed	Total		
100	*Enquiry Account*	0,07	0,11	4,40	40.178	0	40.178	0,00%	209
	Fund Transfers	0,25	0,39	4,52	22.293	0	22.293	0,00%	
TOTAL					62.471	0	62.471	0,00%	
200	*Enquiry Account*	0,09	0,12	1,54	79.962	0	79.962	0,00%	408
	Fund Transfers	0,26	0,46	4,79	41.906	0	41.906	0,00%	
TOTAL					121.868	0	121.868	0,00%	
300	*Enquiry Account*	0,10	0,28	2,38	83.983	0	83.983	0,00%	473

Load User	Fitur	Response Time (detik)			Jumlah Hit			Error Rate	TPS
		Min	Avg	Max	Passed	Failed	Total		
	Fund Transfers	0,26	0,53	5,14	57.408	0	57.408	0,00%	
TOTAL					141.391	0	141.391	0,00%	
400	*Enquiry Account*	0,11	0,45	3,57	84.705	0	84.705	0,00%	506
	Fund Transfers	0,28	0,64	7,67	66.715	0	66.715	0,00%	
TOTAL					151.420	0	151.420	0,00%	
500	*Enquiry Account*	0,11	0,64	1,61	83.904	0	83.904	0,00%	525
	Fund Transfers	0,28	0,77	6,44	72.931	0	72.931	0,00%	
TOTAL					156.835	0	156.835	0,00%	
600	*Enquiry Account*	0,10	0,81	3,01	84.454	0	84.454	0,00%	533
	Fund Transfers	0,29	0,95	8,17	74.881	0	74.881	0,00%	
TOTAL					159.335	0	159.335	0,00%	
700	*Enquiry Account*	0,11	1,00	2,76	83.824	0	83.824	0,00%	537
	Fund Transfers	0,29	1,11	9,60	76.862	0	76.862	0,00%	
TOTAL					160.686	0	160.686	0,00%	
800	*Enquiry Account*	0,11	1,18	4,11	83.748	0	83.748	0,00%	542
	Fund Transfers	0,29	1,27	10,37	78.317	0	78.317	0,00%	
TOTAL					162.065	0	162.065	0,00%	
900	*Enquiry Account*	0,11	1,36	4,18	83.547	0	83.547	0,00%	544
	Fund Transfers	0,30	1,45	11,78	79.042	0	79.042	0,00%	
TOTAL					162.589	0	162.589	0,00%	

Load User	Fitur	Response Time (detik)			Jumlah Hit			Error Rate	TPS
		Min	Avg	Max	Passed	Failed	Total		
1000	*Enquiry Account*	0,11	1,53	4,32	83.713	0	83.713	0,00%	545
	Fund Transfers	0,31	1,63	12,94	79.309	0	79.309	0,00%	
TOTAL					163.022	0	163.022	0,00%	
1100	Enquiry Account	0,11	1,72	4,98	83.241	0	83.241	0,00%	544
	Fund Transfers	0,35	1,81	13,99	79.509	0	79.509	0,00%	
TOTAL					162.750	0	162.750	0,00%	
1200	*Enquiry Account*	0,11	1,91	5,48	83.078	0	83.078	0,00%	538
	Fund Transfers	0,36	2,05	21,93	77.699	0	77.699	0,00%	
TOTAL					160.777	0	160.777	0,00%	
1300	*Enquiry Account*	0,11	2,09	6,18	82.930	0	82.930	0,00%	542
	Fund Transfers	0,38	2,20	24,25	79.175	0	79.175	0,00%	
TOTAL					162.105	0	162.105	0,00%	
1400	*Enquiry Account*	0,11	2,27	6,55	82.940	0	82.940	0,00%	546
	Fund Transfers	0,60	2,36	17,82	80.215	0	80.215	0,00%	
TOTAL					163.155	0	163.155	0,00%	
1500	*Enquiry Account*	0,11	2,45	7,09	83.010	0	83.010	0,00%	546
	Fund Transfers	0,65	2,54	18,77	80.340	0	80.340	0,00%	
TOTAL					163.350	0	163.350	0,00%	
1640	*Enquiry Account*	0,15	2,62	7,59	83.212	0	83.212	0,00%	548
	Fund Transfers	0,67	2,72	20,07	80.547	0	80.547	0,00%	

Load User	Fitur	Response Time (detik)			Jumlah Hit			Error Rate	TPS
		Min	Avg	Max	Passed	Failed	Total		
TOTAL					163.759	0	163.759	0,00%	
1700	*Enquiry Account*	0,30	2,80	7,82	83.141	0	83.141	0,00%	547
	Fund Transfers	0,70	2,90	21,27	80.446	0	80.446	0,00%	
TOTAL					163.587	0	163.587	0,00%	
1800	*Enquiry Account*	0,41	3,02	8,20	82.354	0	82.354	0,00%	545
	Fund Transfers	0,87	3,09	22,42	80.542	0	80.542	0,00%	
TOTAL					162.896	0	162.896	0,00%	
1900	*Enquiry Account*	0,70	3,19	8,25	82.525	0	82.525	0,00%	547
	Fund Transfers	0,91	3,26	25,13	80.891	0	80.891	0,00%	
TOTAL					163.416	0	163.416	0,00%	
2000	*Enquiry Account*	0,58	3,37	8,39	82.649	0	82.649	0,00%	547
	Fund Transfers	0,97	3,44	25,17	80.930	0	80.930	0,00%	
TOTAL					163.579	0	163.579	0,00%	
2100	*Enquiry Account*	1,04	3,54	8,66	82.754	0	82.754	0,00%	548
	Fund Transfers	0,99	3,62	26,66	81.103	0	81.103	0,00%	
TOTAL					163.857	0	163.857	0,00%	
2200	*Enquiry Account*	1,22	3,73	8,79	82.566	0	82.566	0,00%	547
	Fund Transfers	1,05	3,80	27,57	81.082	0	81.082	0,00%	
TOTAL					163.648	0	163.648	0,00%	
2300	*Enquiry Account*	1,39	3,89	8,92	82.876	0	82.876	0,00%	519
	Fund Transfers	1,18	4,18	42,68	71.897	0	71.897	0,00%	
TOTAL					154.773	0	154.773	0,00%	
2400	*Enquiry Account*	1,60	4,10	99,84	82.075	16	82.091	0,02%	280
	Fund Transfers	2,94	6,88	26,82	1.718	191.642	193.360	99,11%	
TOTAL					83.793	191.658	275.451	69,58%	
2500	*Enquiry Account*	1,81	4,33	9,30	81.612	0	81.612	0,00%	273
	Fund Transfers	0,00	0,00	0,00	0	287.976	287.976	100,00%	
TOTAL					81.612	287.976	369.588	77,92%	

D. Analisis Stress Test

Dari hasil *stress test* yang telah dilakukan, ditemukan *error rate* yang mulai muncul ketika di *hit* lebih dari 2.300 *user*. Pada 2.400 *user* didapatkan *error rate* sebanyak 69,58%. Berikut *error* yang berhasil ditangkap oleh Loadrunner pada saat mencapai 2.400 *user*.

TABEL **Error! No text of specified style in document.**-6
Error yang Ditemukan Selama *Stress Test*

No.	Error
1.	*Failed to connect to server* "10.0.19.156:8180" [10061] *Connection refused*
2.	*Failed to connect to server* "10.0.19.156:8280" [10061] *Connection refused*
3.	*Failed to connect to server* "10.0.19.156:8380" [10061] *Connection refused*
4.	*Failed to connect to server* "10.0.19.159:8180" [10061] *Connection refused*
5..	*Failed to connect to server* "10.0.19.159:8280" [10061] *Connection refused*
6.	*Failed to connect to server* "10.0.19.159:8380" [10061] *Connection refused*
7.	*Failed to connect to server* "10.0.19.160:8180" [10061] *Connection refused*
8.	*Failed to connect to server* "10.0.19.160:8280" [10061] *Connection refused*
9.	*Failed to connect to server* "10.0.19.160:8380" [10061] *Connection refused*
10.	*Failed to connect to server* "10.0.19.161:8180" [10060] *Connection timed out*
11.	*Failed to connect to server* "10.0.19.161:8180" [10061] *Connection refused*
12.	*Failed to connect to server* "10.0.19.161:8280" [10061] *Connection refused*
13.	*Failed to connect to server* "10.0.19.161:8380" [10060] *Connection timed out*

14.	*Failed to connect to server* "10.0.19.161:8380" [10061] *Connection refused*

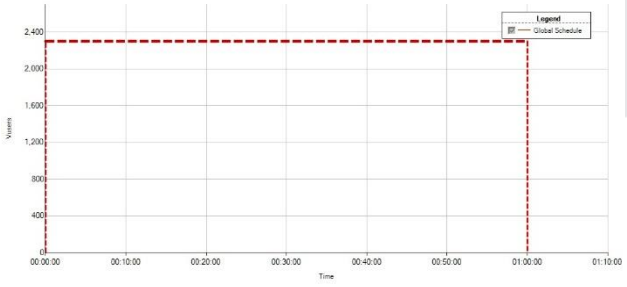
Dari *error* yang di atas tersebut, terlihat untuk *error* ketika *hit* pada server beralamatkan ip 10.0.19.156, 10.0.19.159, 10.0.19.160, 10.0.19.161. Mayoritas dari *error* tersebut terkait *connection refused*, yaitu di mana *request* yang dikirim oleh user ditolak oleh server sehingga *user* tidak menerima *response* dari *request* yang dikirimnya. *Error connection refused* tersebut tidak dapat disimpulkan begitu saja penyebabnya adalah karena koneksi internet dari *user*. Penyebab lain dari *error* tersebut adalah jumlah *request* yang kirim oleh *users* sudah mencapai maksimum yang dapat diterima oleh server, sehingga server menolak *request* yang dikirimkan oleh *user*. Dapat dilihat juga dari tabel 3-1 spesifikasi dari server yang mengalami *error* berbeda atau lebih kecil jika dibandingkan dengan server yang tidak mengalami *error*, di mana hal tersebut bisa menjadi salah satu indikasi yaitu untuk perlu dilakukan peningkatan spesifikasi pada server yang mengalami *error* dengan menyamakan dengan spesifikasi server yang tidak mengalami *error*.

Setelah dilakukannya *stress testing* juga diharapkan dapat mengidentifikasi perilaku aplikasi yang diuji ketika didorong pada kondisi puncak atau kondisi di atas normalnya. Dari hasil *stress test* yang telah dijabarkan, *core banking* PT. XYZ mengalami kondisi *error* di mana server dari *core banking* tersebut *down* sehingga *request* yang berikan oleh *user* banyak yang tidak dapat ditangani oleh server tersebut lagi. Untuk menghindari hal tersebut, perlu dilakukannya upgrade spesifikasi server ataupun juga bisa menambahkan jumlah server pada *core banking* PT. XYZ sehingga ketika terjadi kenaikan jumlah *concurrent user* di atas normalnya maka *core banking* ini masih dapat mengolah request yang diberikan oleh user tanpa mengalami *down*. Untuk dapat mengetahui ukuran *upgrade* spesifikasi ataupun penambahan server, diperlukan penelusuran lebih lanjut.

E. Load Testing

1. Skenario Pengujian

Pada hasil *stress test* yang sudah dilakukan, didapatkan bahwa titik kuat yang dapat ditangani oleh sistem *core banking* adalah di 2.300 user. Dari hasil tersebut akan diambil sebagai acuan untuk dilakukan *load test* sebanyak 2.300 *concurrent user* selama 1 jam untuk menguji kestabilan dari beban yang dapat ditangi oleh sistem ini. Maka dapat digambarkan skenario dari *load test* pada *core banking* ini. Berikut merupakan grafik skenario *load test*.



GAMBAR **Error! No text of specified style in document.**-9
Skenario Load Test 2300 User

2. Hasil Rangkuman Pengujian

Eksekusi *load test* pada *core banking* PT. XYZ berhasil dilakukan dengan skenario 2.300 *concurrent user* selama 1 jam. Berikut merupakan hasil *load testing* setelah dilakukan eksekusi.

TABEL **Error! No text of specified style in document.**-7
Hasil *Load Testing* 2300 *Concurrent User*

Load User	Fitur	*Response Time* (detik)			Jumlah Hit			*Error Rate*	TPS
		Min	*Avg*	*Max*	*Passed*	*Failed*	Total		
2300	*Enquiry Account*	0,08	1,17	35,47	2.917.888	4.603	2.922.491	0,16 %	1.264
	Fund Transfers	0,36	2,29	86,90	1.629.384	75	1.629.459	0,00 %	
TOTAL					4.547.272	4.678	4.551.950	0,10 %	

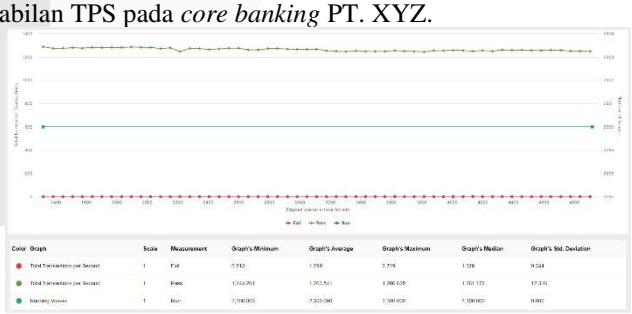
F. Analisis Load Test

Dari hasil *load testing* yang sudah dilakukan, didapatkan hasil yang memenuhi semua objektif yang ada. Adapun objektif yang telah didefinisikan pada able 4-1 yang meliputi sebagai berikut.

TABEL **Error! No text of specified style in document.**-8
Ringkasan Hasil Objektif

No.	Parameter	Objektif	Hasil
1.	*Error rate*	<= 1%	0,10%
2.	*Transactions Per Second* (TPS)	>= 1000 TPS	1,264 TPS
3.	*Response time*	<= 2,5 detik	*Enquiry Account* = 1,17 detik *Fund Transfers* = 2,29 detik

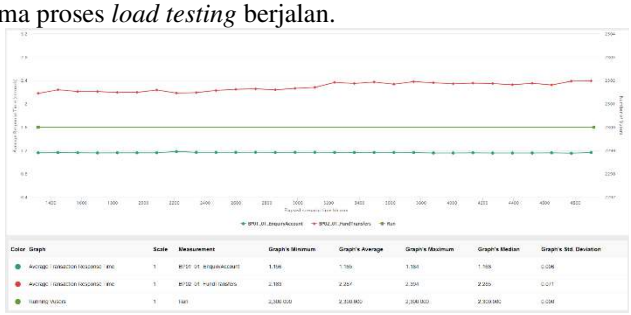
Dari gambar di bawah ini didapatkan *Transactions Per Second* (TPS) sebanyak 1.264 TPS di mana hasil tersebut sudah melampaui ekspektasi pada *core banking* tersebut. Grafik yang ditunjukkan pada gambar 4-5 juga menunjukan kestabilan TPS pada *core banking* PT. XYZ.



GAMBAR **Error! No text of specified style in document.**-10
TPS *Load Testing*

Kemudian untuk *response time* dari kedua fitur juga mendapatkan rata-rata nilai di bawah yang diharapkan. Pada *Enquiry Account* mendapatkan rata-rata *response time* 1,17 detik, sedangkan *Fund Transfers* mendapatkan rata-rata *response time* yang sedikit lebih tinggi yaitu 2,29 detik. Dari hal tersebut bisa diartikan bahwa untuk fitur transaksional memiliki proses yang lebih berat dibandingkan dengan non-transaksional sehingga server memerlukan waktu lebih lama dalam memproses fitur transaksional dibandingkan dengan

non-transaksional. Berikut grafik *response time* pada kedua fitur yang dapat menunjukkan kestabilan untuk keduanya selama proses *load testing* berjalan.



GAMBAR **Error! No text of specified style in document.**-11
Rata-Rata *Response Time Load Testing*

Dalam eksekusi *load testing* ini, ditemukan total *request error* sebanyak 4.678 dari total *request* hit ke *core banking* sebanyak 4.551.950. Meskipun begitu, *error rate* tersebut masih memenuhi ekpektasi yang ditentukan dibawah 1% yaitu 0,10%. Berikut daftar *error* yang ditemukan selama *load testing* berjalan.

TABEL **Error! No text of specified style in document.**-9
Error yang Ditemukan Selama *Load Testing*

No.	Error
1.	*Failed to connect to server* "10.0.19.153:8180" [10061] *Connection refused*
2.	*Failed to connect to server* "10.0.19.153:8280" [10061] *Connection refused*
3.	*Failed to connect to server* "10.0.19.153:8380" [10061] *Connection refused*
4.	*Failed to connect to server* "10.0.19.154:8180" [10061] *Connection refused*
5..	*Failed to connect to server* "10.0.19.154:8280" [10061] *Connection refused*
6.	*Failed to connect to server* "10.0.19.154:8380" [10061] *Connection refused*
7.	*Failed to connect to server* "10.0.19.161:8280" [10060] *Connection timed out*

V. KESIMPULAN

A. Kesimpulan

Kesimpulan yang dapat diambil berdasarkan penelitian yang sudah dilakukan ini adalah:

1. Performance test pada core banking PT. XYZ dieksekusi dengan menjalankan metode stress testing terlebih dahulu sehingga mendapatkan hasil maksimum concurrent users. Kemudian dari hasil tersebut akan menjadi acuan dalam menjalankan load testing sesuai dengan kondisi normalnya.
2. Performance test telah dilakukan pada core banking PT. XYZ dan dengan menggunakan metode load testing dapat dipenuhi aspek pengujian speed dan stability di mana aspek speed pada core banking diukur dari hasil rata-rata response time 1,17 detik untuk non-transaksional dan 2,29 detik untuk transaksional, dan stability pada hasil grafik yang didapatkan tidak ada terjadi kenaikan atau penurunan signifikan sehingga menunjukkan kestabilan pada aplikasi.
3. Sedangkan menggunakan metode stress testing dapat memenuhi aspek pengujian scalability, di mana ketika diuji dengan menaikkan jumlah concurrent user perlahan pada durasi tertentu ditemukan bahwa maksimum beban yang dapat diterima oleh core banking berikut adalah 2.300 concurrent user. Selain didapatkan nilai maksimum beban, pada stress test juga ditemukan behavior dari core banking PT. XYZ yang mengalami server down atau tidak dapat lagi menangani request yang dikirim oleh user ketika didorong pada kondisi melebihi normalnya sehingga perlu dilakukan upgrade spesifikasi atau penambahan server untuk mencegah adanya server down ketika terdapat kenaikan concurrent user ekstrem pada environment production.
4. 2.300 concurrent user hasil dari stress test tersebut diambil dan dijadikan patokan untuk menjalankan load testing. Hasil yang didapatkan setelah melakukan load testing menunjukkan bahwa semua objektif sudah tercapai sehingga dapat dikatakan bahwa sistem core banking tersebut telah stabil di 1.264 TPS, dengan rata-rata response time pada masing-masing fitur berada dibawah 2,5 detik yaitu transaksional di angka 2,29 detik dan non-transaksional berada di angka 1,17 detik, dan memiliki error rate dibawah 1% yaitu sebesar 0,10%.
5. Hasil rata-rata response time fitur transaksional di angka 2,29 detik, sedangkan untuk non-transaksional berada di angka 1,17 detik. Hasil tersebut menunjukkan proses fitur transaksional lebih lama dibandingkan dengan non-transaksional.

5.2 Saran

Berikut beberapa saran yang dapat dilakukan pada penelitian selanjutnya :

1. Pada penelitian performance test selanjutnya diharapkan dapat menggunakan metode yang lain dengan harapan ditemukan kondisi bottleneck pada saat dilakukan pengujian.
2. Melakukan monitoring pada semua server sistem yang dilakukan pengujian agar dapat melihat hasil utilisasi pada server-server tersebut ketika dilakukan pengujian.

REFERENSI

[1] Andarwati M. (2016). Analisis Faktor Yang Mempengaruhi Kesuksesan Penggunaan Core Banking System (CBS) Dengan Menggunakan Model Delone Dan Mclean. Jurnal Keuangan dan Perbankan, 20(3), 458-467.

[2] Permatasari D I, Ardani M, & Ma'ulfa A Y. (2020). Pengujian Aplikasi Menggunakan Metode Load Testing dengan Apache Jmeter pada Sistem Informasi Pertanian. JUSTIN (Jurnal Sistem dan Teknologi Informasi), 8(1), 135-139.

[3] Nandal V, Solanki K, & Dhankhar A. (2018). Performance Testing on Web-based Application using LoadRunner. Jetir, 5(9) 37-42.

[4] Gantini T, Djajalaksana Y M, Yefta S K. (2018). Pengujian Perangkat Lunak itworkforceindoensia.org. Jurnal Teknik Informatika dan Sistem Informasi, 4(3) 355-363.

[5] Sianturi R A, Sinaga A M, Pratama Y. (2021). Perancangan Pengujian Fungsional Dan Non Fungsional Aplikasi Siappara Di Kabupaten Humbang Hasundutan. J-ICON, 9(2), 133-141.

[6] Meier J D, Farre C, Bansode P, Barber S, & Rea D. (2007). Performance Testing Guidance for Web Applications. Microsoft Corporation.

[7] Ari D P S. (2013). Pengaruh Technology Acceptance Model Dan Pengembangannya Dalam Perilaku Menggunakan Core Banking System. Jurnal Keuangan dan Perbankan, 17(2), 267-278.

[8] Hui-li Z, Shu Z, Xiao-jie L, Pei Z, & Shao-bo L. (2012). Research of Load Testing and Result Application Based on LoadRunner. CITCS.

[9] Diastama I G N P D D, Sukarsa I M, Wirdiani N K A. (2021). Pengembangan Test Script untuk Load Testing Web dengan metode Software Testing Life Cycle. JITTER. 2(1).

[10] Ambodo B S, Suryanto R, & Sofyani H. (2017). Testing of Technology Acceptance Model on Core Banking System: A Perspective on Mandatory Use. Jurnal Dinamika Akuntansi, 9(1), 11-22.

[11] Jamil M A, Arif M, Abubakar N S A. (2017). Software Testing Techniques: A Literature Review. Proceedings - 6th International Conference on Information and Communication Technology for the Muslim World. 177-182.

[12] Sarojadevi H. (2011). Performance Testing: Methodologies and Tools. Journal of Information Engineering and Applications, 1(5), 5-13.

[13] Fageria P & Kaushik M. (2014). Research of Load Testing and Result Based on Loadrunner. SSRG International Journal of Civil Engineering, 1(2), 1-4.

[14] Chou C Y, Fang Y B, Wang S T, & Kuo F A. (2020). Smoke and Stress Tests for Travel Service Applications via LoadRunner. LNISA, 11894, 366-373.

[15] Khan R & Amjad M. (2016). Web Application's Performance Testing Using HP LoadRunner and CA Wily Introscope Tools. ICCCA2016, 802-806.

[16] Ramakrishnan R, Shrawan V, & Singh P. (2017). Setting Realistic Think Times in Performance Testing - A Practitioner's Approach. ISEC '17: Proceedings of the 10th Innovations in Software Engineering Conference, 157-164. https://dl.acm.org/doi/10.1145/3021460.3021479

[17] Delta E N & Asmunin. (2016). Performance Test Dan Stress Website Menggunakan Open Source Tools. Jurnal Manajemen Informatika, 6(1), 208-215.

[18] Fansha D A, Setyawan M Y H, & Fauzan M N. (2021). Load Test pada Microservice yang menerapkan CQRS dan Event Sourcing. Jurnal Buana Informatika, 12(2), 126-134.

[19] Pratama M A C, Widowati S, & Junaedi D. (2017). Automasi Performance Load Testing Aplikasi Berbasis Web (Studi Kasus: Sistem Informasi Production Enterprise PT. Bio Farma). Retrieved from https://openlibrary.telkomuniversity.ac.id/home/catalog/id/138996/slug/automasi-performance-load-testing-aplikasi-berbasis-web-studi-kasus-sistem-informasi-production-enterprise-pt-bio-farma-.html

Andriansyah D. (2019). Performance Dan Stress Testing Dalam Mengoptimasi Website. CBIS Journal, 07(1), 23-28.