I. PENDAHULUAN

1.1 Latar Belakang

Website E-Logistik persedian obat yang dikembangkan oleh PT Microdata Indonesia berfungsi sebagai pusat informasi yang terintegrasi untuk mengelola seluruh aspek dalam rantai pasokan obat. Sistem ini mencakup proses pemesanan, pengadaan, pemantauan stok, manajemen gudang, serta koordinasi antara berbagai pihak yang terlibat dalam distribusi obat. Adanya sistem e-logistik persedian obat, proses manajemen dan distribusi obat dapat dilakukan dengan lebih terstruktur dan terotomatisasi. Informasi mengenai ketersediaan obat, status pengiriman, dan kondisi stok dapat diakses secara *real-time*, memungkinkan para pihak terkait untuk mengambil keputusan yang lebih cepat dan tepat.

Sebuah *website* harus akurat dan cepat dalam menyajikan informasi kepada pengguna. Kecepatan laman *web* menyajikan informasi dipengaruhi oleh sejumlah faktor umum, seperti penggunaan gambar beresolusi tinggi pada laman situs atau struktur *script* yang tidak efisien, yang menyebabkan waktu *respons* yang lama saat memuat laman *web* (Andriansyah, 2019). Keterbatasan tentang pemahaman yang dimiliki oleh pengembang terkait kinerja suatu situs *web*. Dampaknya adalah situs *web* tersebut memiliki kinerja atau optimalisasi yang kurang baik, yang pada akhirnya mempengaruhi pengalaman yang dirasakan oleh para pengguna (Diastama dkk, 2021). Jika jumlah pengguna pada sistem terlalu tinggi, dapat mengakibatkan pemborosan sumber daya. Di sisi lain, meremehkan sistem dapat menyebabkan kekurangan sumber daya, yang pada akhirnya dapat menyebabkan sistem *crash* atau kelebihan beban. Penurunan kinerja sistem, seperti waktu *respons* yang lambat, sering terjadi karena peningkatan jumlah pengguna aktif. Ini dapat menyebabkan *server down* atau *crash*. Masalah ini menjadi sangat serius, terutama saat mengakses sistem yang menyimpan informasi yang sangat sensitif (Putri dkk., 2017).

Kualitas produk memiliki peran yang sangat vital dalam industri pembuatan perangkat lunak. Kualitas produk yang dihasilkan sangat ditentukan oleh pengujian perangkat lunak sebelum perangkat lunak tersebut

diimplementasikan kepada pelanggan. Jika perangkat lunak telah melewati uji, maka risiko terjadinya masalah saat perangkat lunak digunakan atau dioperasikan dapat diminimalisir (Ardana, 2019). Faktor penting dalam menentukan kesuksesan suatu situs *web* dalam bersaing dengan pesaing lainnya adalah kemampuannya untuk bersaing dalam hal kecepatan pemuatan. Salah satu metode untuk mengukur kecepatan pemuatan situs *web* ini adalah melalui pengujian kinerja (Tejaya dkk., 2023). (Andriansyah, 2019) *Performance testing* atau pengujian performa bertujuan untuk melakukan verifikasi terhadap performa sistem dengan fokus pada aspek khusus seperti waktu tanggapan, ketersediaan layanan, dan volume halaman yang diakses. Pengujian ini dijalankan melalui simulasi di mana sejumlah besar pengguna bertindak secara bersamaan dalam jangka waktu yang telah ditetapkan. Dalam konteks aplikasi berbasis web, kinerja sistem menjadi isu kritis yang perlu diperhatikan. Ada beberapa jenis *performance test*, yaitu tes tekanan (*stress test*), tes beban (*load test*), tes kekuatan (*strength test*), dan tes volume (*volume test*) (Tejaya dkk., 2023).

Metode pengujian beban (*load test*) lebih sesuai untuk digunakan dalam skenario ini dibandingkan dengan metode lainnya, karena metode pengujian beban mampu mengungkapkan bagaimana sistem berperilaku dalam kondisi beban kerja yang beragam dan bertahap. Dalam penelitian ini, variasi dan bertahannya beban kerja diwakili oleh jumlah pengguna yang mengakses situs *web*/sistem. *JMeter* adalah *tools open source* yang digunakan untuk pengujian beban. Hasil perbandingan dengan beberapa alat lain yang sejenis dapat dijadikan bahan pertimbangan dalam pemilihan alat yang akan dilakukan untuk melakukan pengujian kinerja. *JMeter* cocok untuk pengujian kinerja karena ringan dan mudah dipasang. Hal inilah yang menjadi salah satu alasan pemilihan *JMeter* untuk mendukung pengujian ini (Husufa & Prihandi, 2022; Tejaya dkk., 2023).

Pada tugas akhir ini akan dilakukan pengujian terhadap aplikasi *website* e-logistik persedian obat. Jenis pengujian yang akan dipakai untuk menguji aplikasi tersebut adalah *load testing*. Pengujian performa dengan *load testing* berfokus pada empat indikator yaitu (i) *response time*: waktu rata-rata *respon* pada *server*, (ii) *throughput*: penanganan *request* pada *sever*, (iii) *Error* rate: presentase *error*

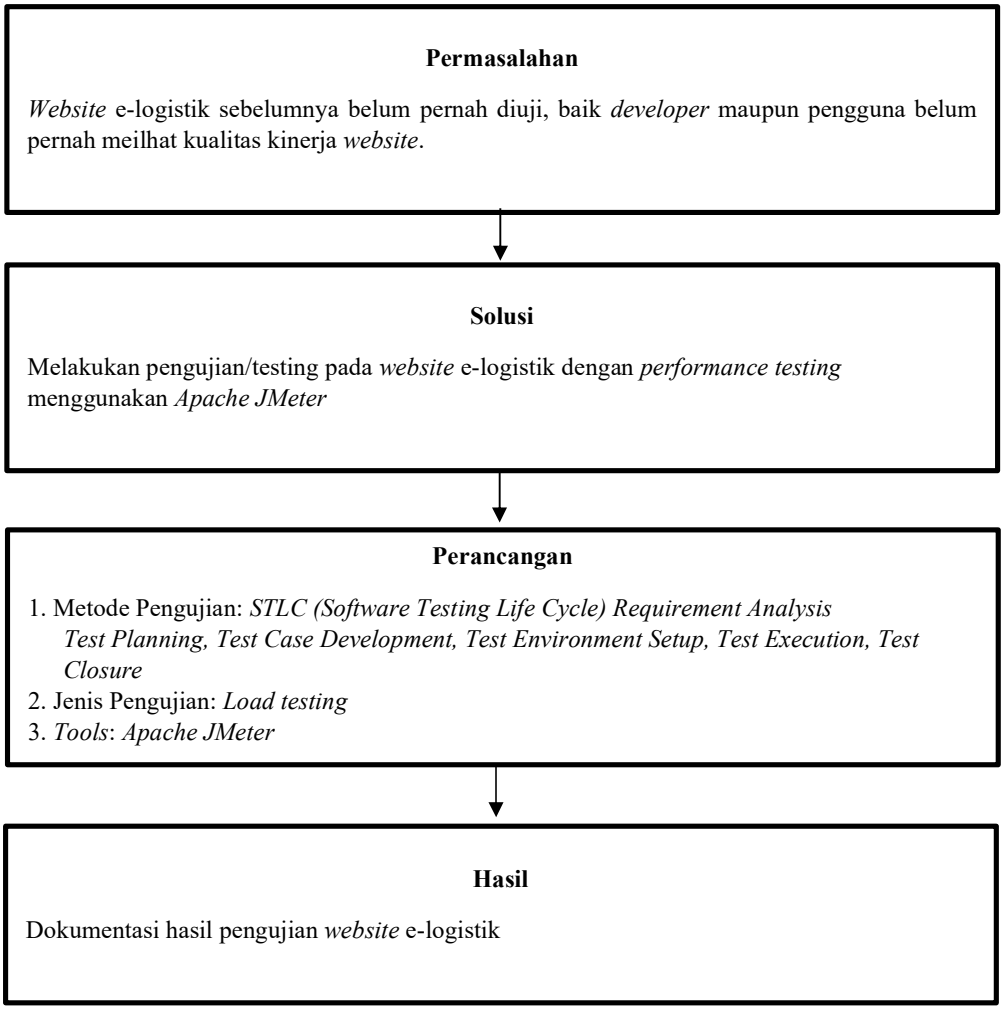
pada saat penanganan *request* berlangsung, (iv) *Apdex:* metrik kinerja aplikasi/ukuran numerik dari kepuasan pengguna.

1.2 Tujuan

Tujuan dari pembuatan tugas akhir ini adalah dengan adanya Implementasi *Performance testing* Pada *Website* E-Logistik Dengan Menggunakan *Apache JMeter*, dapat megetahui hasil kinerja/*perfromance website* e-logistik dalam bentuk dokumentsai.

1.3 Kerangka Pemikiran

Berdasarkan latar belakang masalah dapat disimpulkan dalam suatu kerangka pemikiran yang disajikan dalam gambar 1.



Gambar 1. Kerangka Pemikiran

1.4 Kontribusi

Adapun harapan dengan adanya Implementasi *Performance testing* pada *Website* E-Logistik dengan Menggunakan *Apache JMeter* dapat memberikan kontribusi kepada beberapa pihak. Berikut ini adalah kontribusi yang dapat diberikan:

1. PT Microdata Indonesia

a. Peningkatan Kepercayaan Pelanggan: Dengan melakukan *performance testing* secara menyeluruh pada *website* E-logistik, PT Microdata dapat menunjukkan komitmen mereka terhadap kualitas dan kinerja produk yang disediakan. Hal ini dapat meningkatkan kepercayaan pelanggan terhadap perusahaan dan layanan yang mereka tawarkan.

b. Identifikasi Masalah Lebih Awal: Melalui implementasi *performance testing* menggunakan *Apache JMeter*, PT Microdata dapat mendeteksi masalah kinerja pada *website* E-logistik sebelum diperkenalkan ke lingkungan produksi. Dengan demikian, perusahaan dapat mengidentifikasi dan mengatasi masalah sebelum berdampak negatif pada pengalaman pengguna dan reputasi perusahaan.

2. Klien/Pengguna

a. Performa *Website* yang Lebih Baik: Melalui *performance testing* yang dilakukan oleh PT Microdata dengan menggunakan *Apache JMeter*, klien akan mendapatkan *website* E-logistik yang telah diuji dan dioptimalkan untuk kinerja maksimal. Hal ini akan mengurangi risiko *downtime* dan memastikan pengalaman pengguna yang lebih lancar dan responsif.

b. Keandalan dan Kestabilan: Dengan mengidentifikasi dan memperbaiki masalah kinerja melalui *performance testing*, klien dapat mengandalkan *website* E-logistik mereka untuk berfungsi secara konsisten dan stabil, bahkan saat menghadapi lonjakan trafik atau penggunaan tinggi.

c. Efisiensi Biaya dan Waktu: Dengan menguji dan mengatasi masalah kinerja sebelum diluncurkan, klien dapat menghindari biaya dan waktu tambahan yang terkait dengan perbaikan setelah *website* sudah beroperasi. *Performance testing* membantu dalam mengidentifikasi potensi masalah sejak awal sehingga dapat ditangani dengan efisien.

II. TINJAUAN PUSTAKA

2.1 *Website*

Kumpulan *web page* terdiri dari *image*, video, dan digital file lainnya yang tersimpan pada *server* situs *web* yang dapat diakses melalui Internet. *Web* merupakan kumpulan folder dan file dengan banyak perintah dan kegunaan khusus seperti melihat maupun mengelola penyimpanan informasi. *Web* dapat dikelompokkan ke dalam banyak kategori, seperti *website* pribadi, *website* bisnis dan *website* pemerintah (Dimas, 2021).

2.2 E-Logistik

Auramo J. dalam Sahroni (2017) menyatakan e-logistik merujuk pada proses yang diperlukan untuk mengirimkan barang yang diperdagangkan melalui internet kepada konsumen. Salah satu aspek lebih lanjut yang lebih inovatif adalah bahwa e-logistik terkait dengan integrasi dari rantai pasokan, yang memiliki dampak dalam menghilangkan perantara seperti grosir atau pengecer, serta mendorong munculnya pelaku baru seperti perusahaan logistik. Perusahaan-perusahaan ini harus beradaptasi dengan elemen-elemen tradisional dari rantai pasokan untuk memenuhi kebutuhan bisnis daring (e-bisnis).

2.3 *Software Testing*

Pengujian perangkat lunak merupakan proses untuk menemukan kelemahan atau kesalahan (*bug*) di setiap elemen perangkat lunak, kemudian mencatat hasilnya, dan dilanjutkan dengan mengevaluasi setiap aspek dari setiap komponen serta mengevaluasi fungsionalitas perangkat lunak yang sedang dikembangkan. Tujuan utama dari pengujian perangkat lunak adalah untuk menemukan *bug* atau kesalahan lainnya dalam perangkat lunak sehingga perangkat lunak tersebut sesuai dengan keinginan dan kebutuhan. *Bug* adalah kelemahan atau kerusakan di suatu perangkat lunak yang tidak diinginkan karena tidak sesuai dengan kebutuhan perangkat lunak tersebut. Pengujian perangkat lunak juga dapat dilakukan untuk mengetahui apakah perangkat lunak mampu menghasilkan *output* yang benar untuk berbagai *input* yang berbeda, mampu

menyelesaikan tugas dalam batas waktu yang dapat diterima, dan dapat berjalan dengan baik pada lingkungan yang berbeda(Arifandi dkk., 2022).

2.4 *Performance Testing*

Performance testing atau pengujian kinerja adalah jenis pengujian yang bertujuan untuk mengidentifikasi *response time, throughput, scalability* pada sistem atau aplikasi di bawah beban kerja tertentu. Proses ini juga dapat diartikan sebagai langkah untuk menilai kecepatan atau efektivitas sistem, jaringan, aplikasi perangkat lunak, atau lainnya. Biasanya, pengujian dilakukan secara otomatis agar dapat mensimulasikan berulang-ulang kondisi beban normal, puncak, dan luar biasa dengan mudah. Proses ini juga memungkinkan pembandingan aplikasi berdasarkan parameter seperti kecepatan, transfer data, *throughput*, *bandwidth*, efisiensi, atau keandalan. Selain itu, pengujian kinerja juga dapat berperan sebagai alat diagnostik dalam mengidentifikasi hambatan dan titik kegagalan(Erinle, 2017).

Pengujian kinerja berbeda dari pengujian fungsional. Dalam pengujian kinerja, perhatian utama adalah pada kecepatan, sementara pengujian fungsional berkaitan dengan memastikan perilaku aplikasi yang benar. Meskipun uji kinerja menjadi bagian dari siklus pengembangan yang berkelanjutan bersama dengan uji fungsional, namun hasil dari uji kinerja sebaiknya baru diinterpretasikan setelah aplikasi berhasil melewati pengujian fungsional (Matam, S., & Jain, J., 2017).

2.5 *Load testing*

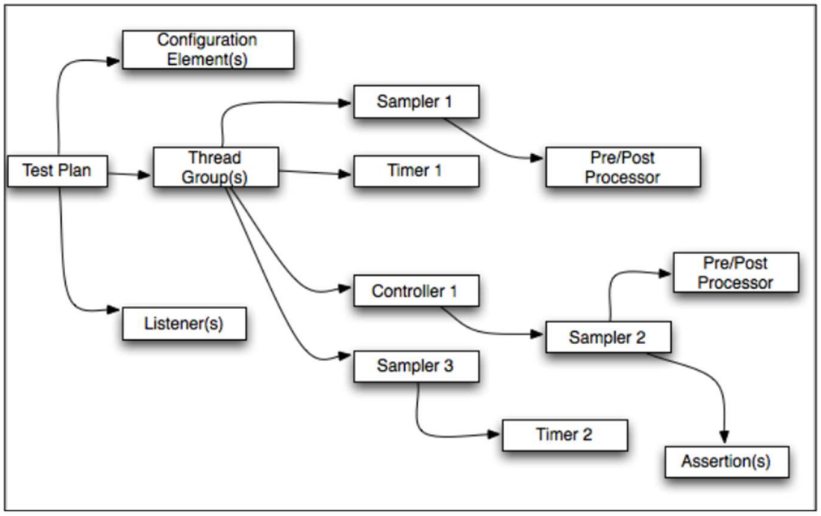
Load testing adalah teknik *performance testing* yang mengukur respons suatu sistem dalam berbagai kondisi beban. Tes ini membantu menentukan bagaimana perilaku perangkat lunak ketika banyak pengguna menggunakan perangkat lunak pada waktu yang sama. Pengujian beban diperlukan untuk mensimulasikan penggunaan situs secara bersamaan (Erinle, 2017; Hendayun dkk., 2023; Permatasari dkk., 2019; Setiawan dkk., 2022).

Pengujian beban (*load testing*) berpusat pada tiga indikator, yaitu: *Response time* mengukur periode menunggu saat klien (pengguna) mengirimkan permintaan ke *server* hingga mendapatkan respons kembali dari *server*, *throughput* adalah jumlah permintaan dari pengguna yang diproses oleh *server* dalam setiap detik / permintaan per detik (*rps*), dan *error rate* menghitung

persentase dari proses yang mengalami kegagalan selama proses permintaan dan respons berlangsung (Setiawan dkk., 2022).

2.6 *Apache JMeter*

Aplikasi *Apache JMeter* merupakan perangkat lunak sumber terbuka yang sepenuhnya berbasis *Java*, dirancang khusus untuk melakukan pengujian perilaku fungsional dan pengukuran kinerja. Pada awalnya dikembangkan untuk menguji Aplikasi *Web*, namun kemudian diperluas untuk mencakup fungsi pengujian lainnya, seperti *FTP*, pengujian *server* basis data, objek *Java*, dan fungsi lainnya. Perangkat lunak ini mampu mensimulasikan jumlah pengguna yang digunakan dan menyediakan analisis dan laporan hasil pengujian (Permatasari dkk., 2019; Raharjo, 2020). Struktur tes *apache jmeter* dapat dilihat pada Gambar 2.



Gambar 2. Struktur Tes *Apache JMeter*

Sumber: (Erlina 2017)

2.6.1 *Test Plan*

Elemen utama dari skrip *JMeter* yang berfungsi sebagai wadah untuk komponen-komponen lain, termasuk *Threads*, *Config Elements*, *Timer*, *PreProcessors*, *PostProcessors*, *Assertions*, dan *Listeners*. Elemen utama ini juga memiliki beberapa konfigurasi sendiri yang dapat diatur (Erinle, 2017).

2.6.2 *Thread Groups*

Titik awal dari setiap rencana pengujian di *JMeter*. Mereka menggambarkan jumlah utas atau pengguna yang akan digunakan oleh *JMeter*

untuk menjalankan rencana pengujian. Semua kontrol dan pembuat sampel untuk pengujian harus berada di bawah grup utas ini. Elemen-elemen lain, seperti pemroses, dapat ditempatkan langsung di bawah rencana pengujian jika ingin menerapkannya ke semua grup utas, atau di bawah grup utas jika terkait hanya dengan grup tertentu. Pengaturan grup utas memungkinkan untuk menentukan jumlah utas yang akan digunakan untuk rencana pengujian, durasi waktu yang diperlukan untuk semua utas menjadi aktif (*ramp-up*), dan berapa kali pengujian akan diulang. Setiap utas akan menjalankan rencana pengujian sepenuhnya secara independen dari utas lainnya. *JMeter* memutar beberapa utas untuk mensimulasikan koneksi bersamaan ke *server*. Penting untuk memastikan "*ramp-up*" cukup panjang agar menghindari beban kerja yang terlalu besar pada awal pengujian, karena hal ini sering menyebabkan kejenuhan jaringan dan hasil pengujian yang tidak valid (Erinle, 2017).

2.6.3 *Controllers*

Controllers terdiri dari *sampler controllers* dan *logic controllers*. *Sampler controller* berfungsi untuk mengirim permintaan ke *server*. Jenis-jenis permintaan ini mencakup *HTTP, FTP, JDBC, LDAP*, dan lain-lain. Walaupun *JMeter* memiliki berbagai macam sampler yang lengkap, tetapi lebih fokus pada penggunaan sampler permintaan *HTTP*, karena penekanan utama kami adalah pada pengujian aplikasi *web*. *Logic controller*, di sisi lain, memungkinkan penyesuaian logika yang digunakan untuk mengirim permintaan. Contohnya, pengontrol *loop* memungkinkan operasi untuk diulang beberapa kali. Pengontrol *if* digunakan untuk mengeksekusi permintaan secara selektif, sementara pengontrol *while* mengizinkan eksekusi permintaan berlanjut hingga beberapa kondisi tidak terpenuhi, dan seterusnya (Erinle, 2017).

2.6.4 *Samplers*

Komponen ini membantu mengirimkan permintaan ke *server* dan menunggu tanggapan. Permintaan diproses sesuai urutan kemunculannya di hierarki (Erinle, 2017). *JMeter dibundel* dengan *sampler* berikut:

a. *HTTP request*
b. *JDBC request*
c. *LDAP request*

d. *SOAP/XML-RPC request*
e. *Webservice (SOAP) request*
f. *FTP request*

2.6.5 *Logic Controllers*

Logic controllers ini berguna untuk mengadaptasi logika yang digunakan dalam menentukan bagaimana permintaan dikirimkan ke *server*. Fungsinya meliputi kemampuan untuk mengubah permintaan, mengulang permintaan, menyisipkan permintaan, mengatur durasi eksekusi permintaan, beralih antar permintaan, mengukur waktu total yang dibutuhkan untuk menjalankan permintaan, dan sebagainya. Saat ini, *JMeter* menyediakan total 16 pengontrol logis (Erinle, 2017).

2.6.6 *Test Fragments*

Terdapat jenis pengontrol khusus yang dirancang khusus untuk digunakan dalam rencana pengujian sebagai upaya untuk memanfaatkan kembali kode tertentu. Pengontrol-pengontrol ini ditempatkan pada tingkat yang sama dengan elemen grup utas dalam pohon rencana pengujian dan tidak akan dieksekusi kecuali jika diakses melalui referensi dari pengontrol *Include* atau *Module Controller* (Erinle, 2017).

2.6.7 *Listeners*

Komponen ini berfungsi mengumpulkan hasil dari jalannya pengujian, sehingga memungkinkan hasil tersebut untuk dianalisis lebih lanjut. Selain itu, listener juga memberikan kemampuan untuk mengarahkan data ke sebuah berkas untuk digunakan di masa mendatang. Lebih lanjut, *listener* juga memungkinkan pengguna untuk menentukan kolom mana yang akan disimpan dan apakah akan menggunakan format *CSV* atau *XML*. Meskipun semua *listener* menyimpan data yang sama, perbedaannya terletak pada cara data tersebut disajikan di layar. *Listener* dapat ditambahkan di berbagai posisi dalam pengujian, termasuk langsung di bawah rencana pengujian. Mereka akan mengumpulkan data hanya dari elemen-elemen yang berada di bawah atau di bawah level mereka (Erinle, 2017).

2.6.8 *Timers*

Secara bawaan, utas-utas *JMeter* mengirimkan permintaan tanpa ada jeda antara masing-masing permintaan. Disarankan untuk menentukan jeda dengan menambahkan salah satu timer yang tersedia ke dalam grup utas. Langkah ini juga membantu membuat rencana pengujian Anda lebih realistis karena pengguna sebenarnya tidak akan mengirimkan permintaan dengan kecepatan secepat itu. *Timer* menyebabkan *JMeter* berhenti sejenak selama jangka waktu tertentu sebelum setiap sampler yang berada dalam lingkupnya (Erinle, 2017).

2.6.9 *Assertions*

Komponen *Assertion* memungkinkan Anda untuk memeriksa respon yang diterima dari *server*. Pada dasarnya, komponen ini berguna untuk memastikan bahwa aplikasi berfungsi dengan benar dan *server* mengembalikan hasil yang sesuai harapan. Assertion dapat digunakan untuk memverifikasi *respon* dalam format *XML, JSON, HTTP*, dan format lain yang dikirimkan oleh *server*. Namun, perlu diingat bahwa penggunaan *Assertion* dapat menyebabkan penggunaan sumber daya yang tinggi, oleh karena itu pastikan Anda tidak mengaktifkannya saat melakukan pengujian yang sesungguhnya (Erinle, 2017).

2.6.10 *Configuration Elements*

Elemen konfigurasi berinteraksi secara erat dengan *sampler*, memungkinkan untuk melakukan modifikasi atau penambahan pada permintaan. Elemen-elemen ini hanya dapat diakses dari dalam cabang pohon tempat elemen tersebut ditempatkan. Beberapa contoh elemen konfigurasi ini meliputi *HTTP Cookie Manager*, *HTTP Header Manager*, dan sebagainya (Erinle, 2017).

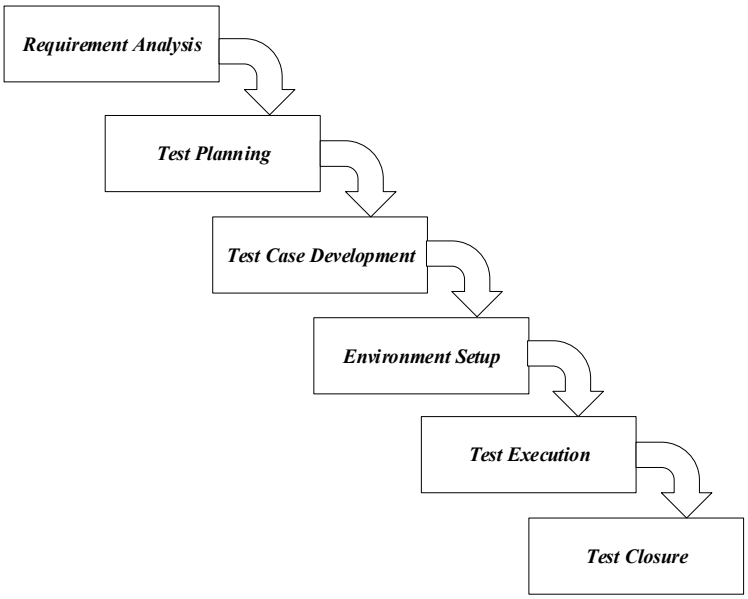
2.6.11 *Preprocessor and Postprocessor Elements*

Elemen pra-pemroses, seperti namanya, melakukan beberapa tindakan sebelum permintaan dilakukan. Elemen ini sering digunakan untuk mengubah pengaturan permintaan sebelum permintaan itu dijalankan atau untuk memperbarui variabel yang tidak diambil dari teks respons (Erinle, 2017).

2.7 *Software Testing Life Cycle (STLC)*

Siklus Hidup Pengujian Perangkat Lunak (*Software Testing Life Cycle* atau *STLC*) adalah rangkaian tindakan yang dilakukan untuk menguji suatu perangkat lunak. Tahap ini merujuk pada rangkaian langkah-langkah yang khusus

dalam proses pengujian. Dalam STLC, setiap tahapan dilakukan dengan metode yang terencana dan terstruktur. Beberapa tahapan dalam Siklus Hidup Pengujian Perangkat Lunak meliputi Analisis Persyaratan (*Requerement Analysis)*, Perencanaan Pengujian (*Test Planning*), Pembuatan Kasus Pengujian (*Test Case Development*), Persiapan Lingkungan (*Environmrent Setup*), Pelaksanaan Pengujian (*Test Execution*), &Penutupan Tes (*Test Closure*) (Kurnia & Yulianti, 2021). Siklus *software testing life cycle* dapat dilihat pada Gambar 3.



Gambar 3. Siklus *STLC* (*Software Testing Life Cycle*)

2.8 *Throughput*

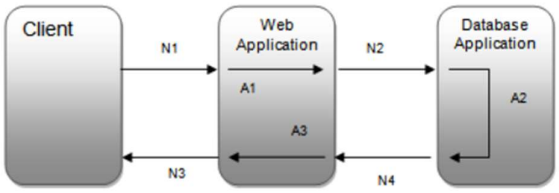
Throughput adalah parameter yang paling penting. Hal ini menggambarkan kapasitas *server* dalam mengatasi beban yang besar. Semakin tinggi *Throughput*, semakin optimal performa *server*. *Throughput* dihitung dengan mengukur jumlah permintaan yang diproses dalam satu unit waktu. Waktu yang dihitung dimulai dari saat sampel pertama dimulai hingga sampel terakhir selesai. Rentang waktu ini juga mencakup interval antara sampel, yang harus mencerminkan beban pada *server*. Rumusnya adalah: $\left(Throughput = \frac{(\text{jumlah permintaan})}{(\text{total waktu})}\right)$ (Ghodasara, Y. R., dkk, 2018).

2.9 *Response Time*

Performa yang baik pada suatu *web* juga dapat dikenali dari *response time* yang dihasilkan, semakin kecil *response time* yang dihasilkan, maka semakin baik kinerja *web* tersebut. *Response time* adalah waktu yang dibutuhkan oleh pengguna untuk menunggu halaman *web* ditampilkan saat pertama kali dikunjungi. Secara umum, pengguna menginginkan waktu akses yang cepat saat mengakses situs *web*, tetapi kecepatan *response time* juga dipengaruhi oleh beberapa faktor seperti jumlah pengguna yang mengakses dan kinerja jaringan (Raharjo, 2020).

a. Semakin banyak pengguna, maka performa *web server* dapat menjadi lebih lambat. Namun, dalam waktu puncak (*peak load periods*), jumlah pengguna bisa melebihi perkiraan rata-rata dan menyebabkan penurunan kinerja.
b. Kinerja jaringan atau kecepatan transmisi mengacu pada jumlah data yang dapat melewati suatu media dalam satu detik. Biasanya diukur dalam bit per detik (*bit per second*) dan disimbolkan sebagai bit/s atau *bps*.

Response time mengacu pada interval waktu dari saat pengguna mengirim permintaan ke *server* hingga *server* menyelesaikan *respons* terhadap permintaan tersebut. Dalam konteks aplikasi *web*, waktu *respons* didefinisikan sebagai jumlah waktu jaringan (N1, N2, N3, N4) dan waktu aplikasi (A1, A2, A3)(Ilham & Niswar, dkk, 2023). Proses *web request* dapat dilihat pada Gambar 4.



Gambar 4. Proses *Web Request*

Kriteria utama untuk menentukan apakah sistem memiliki kinerja yang optimal: waktu proses tidak lebih dari 3 detik (*response time* < 3000ms) (Permatasari, 2020; Qomariyah dkk., 2023).

2.10 *Apdex*

APDEX adalah pedoman yang dibentuk oleh aliansi perusahaan (*www.Apdex.org*) untuk menetapkan format baku dalam melaporkan, mengukur,

dan melacak kinerja aplikasi (Stadnik dan Nowak, 2018). *APDEX* berfungsi sebagai metrik yang mengukur kepuasan pengguna terhadap kinerja aplikasi dengan skala standar dari 0 hingga 1 (0 = tidak ada pengguna yang puas, 1 = semua pengguna puas). Indeks ini berlandaskan pada tiga kategori tanggapan aplikasi, yakni Kepuasan (*Satisfied*), Toleransi (*Tolerating*), dan Keterbatasan (*Frustrated*). Kategori Kepuasan merepresentasikan nilai respons di bawah batas waktu (T detik) di mana pengguna tidak terganggu oleh waktu tanggapan aplikasi, mengindikasikan produktivitas penuh dari pengguna. Kategori Toleransi mengindikasikan bahwa pengguna menganggap kinerja aplikasi lambat dalam respon lebih batas waktu F-detik sehingga menjadi tidak dapat diterima, dan pengguna bisa mengabaikan atau menghentikan proses tersebut. Formula *APDEX* $\left(Apdex_T = \frac{Satisfied\ count + \frac{Tolerating\ count}{2}}{Total\ samples}\right)$. *Apdex qualitative reporting rules* dapat dilihat pada Tabel 1.

Tabel 1. *Apdex Qualitative Reporting Rules*

Apdex	**Rating**
0.94 to 1.00	*Excellent*
0.85 to 0.93	*Good*
0.70 to 0.84	*Fair*
0.50 to 0.69	*Poor*
0.00 to 0.49	*Unacceptable or UNAX*

Sumber: *Apdex Alliance, Inc.*

2.11 *Error Rate*

Error diartikan sebagai perbuatan yang tidak diharapkan atau tidak disengaja, seperti slip, kesalahan, atau kelalaian yang terjadi ketika pengguna sedang melakukan tugas. Komponen kesalahan diukur dengan menggunakan tingkat kesalahan (*error rate*) (Situmorang dkk., 2019).

2.12 Artikel Ilmiah Terkait

Berikut beberapa artikel ilmiah terkait sebagai referensi dalam pembuatan tugas akhir ini:

Tabel 2. Artikel Terkait

No	Nama Penulis	Judul Artikel Ilmiah Terkait	Hasil Artikel Ilmiah Terkait
1.	Wanta Tejaya, Syaiful Rahman, Abdul Munir (2023)	Pengujian *Website Invitees* menggunakan Metode *Load Testing* dengan *Apache JMeter*	Hasil pengujian yang telah dilaksanakan menunjukkan bahwa kinerja situs web

			Invitees tidak berjalan secara optimal. Rata-rata total waktu pemuatan dari tiga skenario pertama adalah sekitar 3.1 detik. Ini berdasarkan skenario pertama dan kedua. Meskipun begitu, ada kebutuhan yang signifikan untuk melakukan perbaikan, terutama pada fungsi untuk melihat undangan.
2.	Cahaya Putri Agustika, Wahyu SJ Saputra, Mohammad Idhom (2021)	Pengujian Aplikasi *Greenwallet* dengan Metode *Load Testing* dan *Apache JMeter*	Hasil dari penelitian menunjukkan bahwa sebagian besar skenario yang telah diuji telah berhasil mencapai tujuan pengujian dengan memiliki waktu respons kurang dari 1.0 untuk *server* layanan hosting.
3.	Mohamd Hendayun1, Arief Ginanjar, Yoan Ihsan (2023)	*Analysis Of Application Performance testing Using Load Testing And Stress Testing Methods In Api Service*	Studi ini menganalisis perilaku sistem di lingkungan *server* yang sedang berjalan dan kemudian mengoptimalkan konfigurasi layanan dan *server* untuk mencapai pengguna bersamaan dengan 500 pengguna sekaligus dengan *JMeter* sebagai alat uji kinerja.
4.	Doni Andriansyah (2019)	*Performance* dan *Stress Testing* dalam mengoptimasi *Website*	Hasil dari penelitian ini berupa dokumentasi pengujian yang telah dilakukan dan diimplementasikan terhadap *Website* terkait. Terdapat perbedaan score yang signifikan terhadap hasil pengujian sebelum dilakukannya optimasi dengan setelah dilakukannya optimasi *Website*.
5.	Desy Intan Permatasari, Budi Santoso, Nadia Ningtias, M. Halim Y.R), Rafidah Atika, Nadia Widad, Ikbar Maulana, Aditya Abdurrahman (2019)	Pengukuran *Throughput Load Testing* menggunakan *Test Case Sampling Gorilla Testing*	Hasil pengujian load testing menunjukkan bahwa dengan menggunakan *Apache JMeter*, target pengujian berupa loading time dan process memory sudah terpenuhi.