Pengujian Aplikasi SIHARAPAN Menggunakan Metode *Stress Testing*

Billy Sabella
Teknologi Informasi
Politeknik Negeri Tanah Laut
Pelaihari, Indonesia
billy.sabella@politala.ac.id

Ridha Rahma Tina
Teknologi Informasi
Politeknik Negeri Tanah Laut
Pelaihari, Indonesia
ridha.rahma.tina@mhs.politala.ac.id

Ferdiyansyah Achmad
Teknologi Informasi
Politeknik Negeri Tanah Laut
Pelaihari Indonesia
ferdiyansyah.achmad@mhs.politala.ac.id

Alal Lestari
Teknologi Informasi
Politeknik Negeri Tanah Laut
Pelaihari, Indonesia
alal.lestari@mhs.politala.ac.id

Fathurrahmani
Teknologi Informasi
Politeknik Negeri Tanah Laut
Pelaihari, Indonesia
fathurrahmani@politala.ac.id

The SIHARAPAN application is developed as a tool for early detection of stunting symptoms in children and as a tool to assist the activities of integrated health posts (posyandu). Therefore, it is important to conduct stress testing to assess the performance of the application when faced with high loads. This research performs Stress Testing using two different tools, namely Apache JMeter and Locust, to measure the performance of a web application when subjected to high loads. Testing is conducted with gradually increasing numbers of users, namely 50, 150, and 250 users, with a 5-second ramp-up period. Each test involves three types of access: index access, child check, and data addition. The test results show that the application's performance in terms of response time and throughput tends to decrease with increasing numbers of users. Although throughput increases with an increase in the number of users, there is also a significant increase in the failure rate of requests, especially when the number of users reaches 250. Testing with Apache JMeter and Locust provides a comprehensive understanding of the application's performance under high loads. These results can be used to identify areas that need improvement in the application to enhance its performance and reliability.

Keywords: Stress Testing, Apache Jmeter, Locust

Abstrak*— Aplikasi SIHARAPAN adalah aplikasi yang dikembangkan sebagai alat untuk mendeteksi dini gejala stunting pada anak dan alat untuk membantu proses kegiatan posyandu. Oleh karena itu penting untuk melakukan *stress testing* untuk menguji kinerja aplikasi saat menghadapi beban yang yang tinggi. Penelitian ini melakukan pengujian Stress Testing menggunakan dua alat yang berbeda, yaitu Apache JMeter dan Locust, untuk mengukur kinerja sebuah aplikasi web saat dihadapkan pada beban tinggi. Pengujian dilakukan dengan jumlah pengguna yang bertahap meningkat, yaitu 50, 150, dan 250 pengguna, dengan periode *ramp-up* 5 detik. Setiap pengujian melibatkan tiga jenis akses yaitu, akses index, periksa anak, dan tambah data. Hasil pengujian menunjukkan bahwa kinerja aplikasi dalam hal waktu respons dan *throughput* cenderung menurun seiring peningkatan jumlah pengguna. Meskipun throughput meningkat dengan peningkatan jumlah pengguna, terdapat juga peningkatan signifikan dalam tingkat kegagalan permintaan, terutama saat jumlah pengguna mencapai 250. Pengujian dengan Apache JMeter dan Locust memberikan pemahaman yang komprehensif tentang kinerja aplikasi dalam menghadapi beban tinggi. Hasil ini dapat digunakan untuk mengidentifikasi area yang perlu ditingkatkan dalam aplikasi untuk meningkatkan kinerja dan keandalannya.

Kata Kunci : Stress Testing, Apache Jmeter, Locust

PENDAHULUAN

Pengembangan aplikasi HARAPAN sebagai sarana untuk menanggulangi stunting di desa merupakan inovasi dalam meningkatkan kualitas kesehatan anak-anak dan kesadaran gizi di masyarakat. Harapan sendiri merupakan singkatan dari Hindari Anak Rendah Asupan Pangan dan Akan Nutrisi, sehingga disingkat menjadi SIHARAPAN. Aplikasi ini tidak hanya berfungsi sebagai alat untuk mendeteksi dini gejala stunting pada anak-anak, tetapi juga sebagai sarana untuk mempermudah proses posyandu, dengan memfasilitasi pendataan, dokumentasi, dan pelaporan. Aplikasi ini memberikan kemudahan bagi petugas posyandu dalam mencatat data perkembangan balita, melakukan dokumentasi hasil pemeriksaan, serta melakukan pelaporan secara lebih akurat. Tidak hanya bagi petugas posyandu aplikasi ini juga membantu desa dalam mengelola data kesehatan anak-anak secara lebih terstruktur dan efisien, sementara masyarakat mendapatkan akses informasi gizi dan kesehatan anak yang lebih cepat dan mudah, serta peningkatan kesadaran tentang pentingnya asupan gizi yang baik untuk mencegah stunting. Bagi anak-anak, aplikasi ini memungkinkan deteksi dini gejala stunting sehingga penanganan dapat dilakukan lebih cepat dan tepat, yang pada akhirnya meningkatkan perkembangan kesehatan mereka.

Berdasarkan latar belakang yang sudah diuraikan aplikasi SIHARAPAN memiliki banyak manfaat. Namun dalam implementasinya tidak lepas dari beberapa permasalahan, salah satunya terkait ketahanan dan kehandalan aplikasi dalam menghadapi kondisi ekstrim. Kondisi ekstrem ini bisa terjadi misalnya, saat ada kampanye kesehatan atau kegiatan posyandu besar-besaran yang menyebabkan peningkatan jumlah pengguna secara tiba-tiba. Pada saat lonjakan pengguna ini, aplikasi harus mampu menangani beban tambahan tanpa mengalami penurunan kinerja yang signifikan. Jika aplikasi tidak tahan terhadap kondisi ini, maka bisa terjadi penurunan kinerja seperti lambatnya respon aplikasi, di mana waktu tunggu untuk memuat halaman atau data menjadi lebih lama dari biasanya. Selain itu, pengguna mungkin mengalami kesulitan mengakses fitur-fitur tertentu yang menjadi sangat penting dalam proses pendataan dan pelaporan, atau bahkan kegagalan sistem yang mengakibatkan aplikasi tidak bisa digunakan sama sekali. Selain ketahanan

DOI: https://doi.org/10/25047/jtit.v11i1.359

aplikasi, kehandalan aplikasi mencakup kemampuan aplikasi untuk memberikan kinerja yang konsisten dan bebas dari kesalahan selama digunakan dalam jangka waktu yang lama juga sangat penting. Hal ini karena aplikasi SIHARAPAN diharapkan tidak hanya digunakan sekali atau dua kali, tetapi secara terus-menerus baik oleh petugas posyandu, desa ataupun masyarakat. Kehandalan ini mencakup beberapa aspek, seperti integritas data yang berarti semua data yang dimasukkan harus disimpan dan dikelola dengan benar tanpa adanya korupsi data, serta ketersediaan aplikasi yang berarti aplikasi harus selalu dapat diakses kapan saja dibutuhkan tanpa adanya downtime yang tidak terencana.

Berdasarkan permasalahan tersebut, tujan dari penelitian ini adalah untuk memastikan ketahananan dan kehandalan dari aplikasi SIHARAPAN, sehingga perlu dilakukan pengujian perangkat lunak yaitu *stress testing*. Pengujian perangkat lunak merupakan proses yang menganalisis komponen perangkat lunak untuk mengidentifikasi perbedaan antara keadaan aktual dan keadaan yang diharapkan, seperti kesalahan atau kecacatan [1]. Pengujian perangkat lunak juga bertujuan untuk mengetahui kekurangan dan permasalahan dari sistem yang sudah dirancang sehingga ditemukan solusi untuk mengoptimalkan kinerja dari aplilasi [2]. Pengujian perangkat lunak dapat dilakukan dengan berbagai metode, namun menurut Ramadhani dan rekan-rekannya dalam jurnalnya menyatakan bahwa untuk menguji ketahanan dan keandalan aplikasi dapat dilakukan dengan menggunakan metode *stress testing*, yang dilakukan dengan menguji kinerja prototipe *platform* saat menangani banyak permintaan pada waktu tertentu dan memeriksa batas atas permintaan yang dapat ditangani oleh sistem [3]. *Stress Testing* sendiri merupakan sebuah teknik evaluasi performa yang difokuskan pada daya tahan, ketersediaan, dan kehandalan suatu aplikasi dalam situasi-situasi yang sangat menekan. Situasi-situasi tersebut dapat mencakup beban berat, tingkat konkurensi yang tinggi, atau keterbatasan sumber daya komputasi [4]. *Stress testing* pada suatu aplikasi dilakukan untuk menekan aplikasi dan mengamati respons serta kemampuan pemulihan sistem saat menghadapi tekanan. Tujuan utamanya adalah untuk menguji respons sistem terhadap beban berat tanpa menyebabkan kerusakan permanen [5].

Alasan penulis menggunakan metode *stress testing* karena, metode *stress testing* dapat dilakukan untuk menguji ketahanan dan kehandalan aplikasi dalam menghadapi berbagai situasi. Penggunaan *stress testing* sangat penting karena, *stress testing* mampu mengevaluasi kemampuan sistem dalam menghadapi tekanan dan menangani kesalahan ketika menghadapi kondisi beban yang ekstrem [6]. Melalui *stress testing*, sistem diuji untuk memastikan bahwa ia tetap beroperasi bahkan dalam situasi kritis tanpa mengalami kerusakan atau kegagalan. Hal ini bisa memastikan bahwa aplikasi SIHARAPAN tetap dapat diandalkan dan tersedia meskipun dalam situasi yang paling menekan, seperti lonjakan penggunaan tiba-tiba atau tingginya tingkat konkurensi pengguna. Untuk melakukan *stress testing* pada aplikasi SIHARAPAN, digunakan *tools* apache jmeter karena apache jmeter memiliki kemampuan yang kuat dalam menguji beban dan kinerja aplikasi web. Selain itu, apache jmeter dapat digunakan secara efektif dalam evaluasi kinerja server FTP. Dengan memantau total waktu, latensi, dan laju transfer transaksi FTP antara server dan klien, apache jmeter memberikan informasi yang berharga tentang kinerja server FTP [7]. Apache jmeter juga dipilih karena, berdasarkan penelitian yang dilakukan oleh [8] menunjukkan bahwa apache jmeter adalah alat yang banyak digunakan dan terbukti memiliki kinerja lebih baik dibandingkan dengan alat lain seperti Siege, LoadRunner, dan Microsoft Visual Studio. Selain Apache JMeter, tim pengembang juga menggunakan Locust sebagai alat untuk melakukan *stress testing* pada aplikasi SIHARAPAN.

Penelitian-penelitian sebelumnya yang berkaitan dengan penelitian ini berupa jurnal-jurnal dan beberapa referensi pendukungnya. Diantaranya, penelitian tentang *stress testing* yang dilakukan dengan menerapkan skenario pengujian melalui tiga tahap. Pada tahap pertama, 100 permintaan diberikan kepada 50 pengguna yang mengakses aplikasi secara bersamaan. Selanjutnya, pada tahap kedua, 200 permintaan diberikan kepada 100 pengguna yang dapat mengakses aplikasi secara bersamaan, dan pada tahap ketiga, 300 permintaan diberikan kepada 200 pengguna yang dapat mengakses aplikasi secara bersamaan. Hasil dari penelitian ini menunjukkan bahwa *stress testing* dapat memberikan hasil yang signifikan mengenai respons waktu atau durasi respon aplikasi terhadap beban kerja tertentu[9]. Pada penelitian lain, dilakukan pengujian pada Website Sistem Informasi Guru Sekolah Kejuruan Global Abiansemal Bali dengan Apache JMeter dan BlazeMeter pada empat modul yaitu, halaman login, pembaruan profil guru, pengunggahan gambar, dan pembuatan soal. Pengujian dilakukan dengan 50 dan 100 sampel, periode ramp-up 10 detik, dan loop 1 kali. Hasil dari *stress testing* pada penelitian ini menunjukkan bahwa sistem bekerja dengan baik dengan waktu respons stabil, *throughput* meningkat, dan deviation menurun [10]. Penelitian serupa juga dilakukan dengan menambahkan jumlah *virtual users* dari 25 hingga 200 yang mengakses web secara bersamaan menggunakan JMeter. Hasilnya, website hanya mengalami 2 kegagalan dari 200 pengguna, sementara jumlah pengguna di tingkat kelurahan, kecamatan, dan kota tidak mencapai 100 [11]. Selain itu, dalam penelitian yang dilakukan oleh [12] dalam jurnalnya menjelaskan bahwa dengan menggunakan metode *stress testing* dapat menjadi kunci dalam memastikan bahwa sistem tetap dapat berjalan dengan baik bahkan ketika jumlah aktivitas pengguna meningkat. Kemudian, pada penelitian yang dilakukan oleh [13] menekankan pentingnya *stress testing* dalam pengelolaan server website. *Stress testing* mengidentifikasi masalah dan batasan kinerja, memastikan server tetap beroperasi di bawah beban kerja tinggi, seperti lonjakan pengunjung atau permintaan.

Perbedaan penelitian ini dengan penelitian terkait yang sudah dilakukan sebelumnya adalah, pada pengujian aplikasi SIHARAPAN pengujian *stress testing* dibandingkan hanya menggunakan apache jmeter tim pengembang juga memperluas cakupan *tools* pengujian dengan memasukkan alat lain, yaitu Locust. Alasan tim pengembangan menambahkan penggunaan locust adalah untuk memberikan sudut pandang tambahan dengan pendekatan yang berbeda. Sehingga mungkin dapat memberikan pemahaman tentang performa sistem dalam situasi yang berbeda atau memungkinkan identifikasi masalah yang mungkin tidak terdeteksi oleh Apache JMeter saja. Selain itu, terdapat perbedaan dalam parameter pengujian seperti jumlah pengguna yang digunakan dalam stress testing. Penelitian sebelumnya menggunakan jumlah pengguna yang berbeda-beda seperti 25, 50, dan 75, sedangkan pada penelitian ini, pengujian dilakukan dengan jumlah pengguna 50, 150, dan 250. Perbedaan dalam skenario pengujian ini dapat

DOI: https://doi.org/10/25047/jtit.v11i1.359

memberikan wawasan tambahan tentang bagaimana sistem bereaksi terhadap beban pengguna yang berbeda.

METODE PENELITIAN

Penelitian ini menggunakan metode *Stress testing* untuk menguji kinerja dari aplikasi. Metode penelitian ini menjelaskan langkah-langkah pengujian yang terdiri dari beberapa tahap yang terinci sebagai berikut.

A. Instalasi *Tools* Pengujian dan Persiapan Aplikasi

Dalam konteks penelitian ini, langkah pertama yang dilakukan adalah menginstal perangkat lunak Apache Jmeter dan *framework* Locust sesuai dengan petunjuk yang tersedia dari situs *web* resminya. Instalasi ini melibatkan mengunduh JMeter dan locust dengan mengikuti langkah-langkah instalasi yang disediakan di situs web resmi nya. Setelah Apache JMeter dan locust terinstal dengan sukses, langkah selanjutnya adalah mempersiapkan aplikasi yang akan diuji. Dalam hal ini, aplikasi tersebut disiapkan untuk berjalan pada alamat "localhost:8080". Ini berarti bahwa aplikasi tersebut diatur untuk berjalan secara lokal pada komputer tempat Apache JMeter diinstal, dengan menggunakan port 8080 sebagai titik akses. Persiapan aplikasi ini termasuk memastikan bahwa semua komponen yang diperlukan telah diaktifkan dan bahwa aplikasi tersebut siap untuk diuji, baik dari segi fungsionalitas maupun kinerja. Dengan melakukan kedua langkah ini, infrastruktur yang diperlukan untuk menjalankan *stress testing* dengan menggunakan Apache JMeter dan locust telah terpasang dan aplikas yang akan diuji telah siap untuk dievaluasi kinerjanya.

B. Menentukan Skenario Pengujian

Setelah instalasi alat pengujian dan persiapan aplikasi berhasil, langkah selanjutnya adalah menentukan skenario pengujian. Dalam penelitian ini, skenario pengujian dirancang untuk menguji kinerja aplikasi SIHARAPAN melalui serangkaian situasi penggunaan yang umumnya terjadi dalam aplikasi tersebut. Ini mencakup proses tampil data, tambah data, ubah data, periksa gizi anak, dan pencetakan dokumen PDF. Dalam pengujian ini, skenario pengujian akan disusun untuk memasukkan sistem ke dalam berbagai situasi beban pengguna yang mungkin terjadi di lingkungan produksi. Ini melibatkan mensimulasikan akses ke aplikasi dengan jumlah pengguna yang besar secara bersamaan, untuk menguji respons sistem pada berbagai tingkat beban. Selain itu, skenario akan mengatur variasi dalam periode *ramp up*, yang merupakan waktu yang diperlukan untuk menambahkan pengguna baru ke dalam pengujian, dan *loop count*, yang menentukan berapa kali setiap pengguna akan menjalankan skenario pengujian. Dengan menggunakan variasi ini, pengujian akan dapat mengevaluasi bagaimana sistem SIHARAPAN merespons dalam situasi beban yang bervariasi dan memberikan wawasan yang lebih mendalam tentang kinerja sistem dalam kondisi yang beragam.

C. Konfigurasi *Tools* Pengujian

Setelah menentukan skenario pengujian langkah selanjutnya adalah melakukan konfigurasi pada *tools* pengujian, yakni Apache JMeter dan Locust, agar sesuai dengan skenario yang telah ditetapkan. *Stress testing* dilakukan dengan menggunakan parameter HTTP response dan HTTP *request* bersama dengan konfigurasi lainnya seperti jumlah *thread*, periode peningkatan bertahap (*ramp-up*), dan jumlah pengulangan *(loop count)*[14]. Pertama, *thread group* pada Apache JMeter disesuaikan dengan jumlah pengguna virtual, periode *ramp up*, *dan loop count* yang sesuai dengan skenario pengujian. Begitu pula, pada platform Locust, pengaturan jumlah pengguna virtual dan tingkat laju ditentukan agar mencerminkan variasi beban pengguna seperti yang tercantum dalam skenario. Setelah melakukan konfigurasi tersebut, langkah selanjutnya adalah mengatur skenario pengujian yang mencakup langkah-langkah seperti tampil data, tambah data, ubah data, periksa anak, dan pencetakan dokumen PDF. Dalam mengatur skenario pengujian menggunakan Apache JMeter dan Locust, pengguna perlu menyesuaikan path dan HTTP request sesuai dengan kebutuhan aplikasi SIHARAPAN yang akan diuji. Di Apache JMeter, ini melibatkan konfigurasi elemen HTTP Request yang mencakup pengaturan endpoint URL, metode HTTP yang digunakan, serta pengaturan header dan parameter lainnya. Di sisi lain, pada platform Locust, path dan HTTP request juga harus diatur dalam definisi tugas (*task*) yang akan dilakukan oleh pengguna virtual. Selain itu, *port number* juga dapat diatur sesuai dengan konfigurasi server aplikasi yang akan diuji, baik pada Apache JMeter maupun Locust.

D. Analisis Hasil Pengujian

Setelah konfigurasi selesai, pengujian dapat dilakukan. Apache JMeter dan Locust akan menjalankan skrip pengujian yang telah dikonfigurasi sesuai dengan skenario yang ditentukan sebelumnya. Setelah pengujian selesai, kedua alat akan menampilkan hasil pengujian dengan berbagai metode, termasuk tabel, grafik, dan laporan lainnya sesuai kebutuhan. Data yang dihasilkan akan mencakup beberapa metrik kinerja utama yang diperoleh meliputi waktu respons, *throughput*, dan kesalahan. Selain itu, juga penting untuk memantau rata-rata waktu tanggap, beban server, keberhasilan permintaan, rasio kesalahan, rata-rata koneksi *concurrent*, dan latensi. Metrik-metrik ini memberikan gambaran umum tentang kinerja sistem SIHARAPAN dalam kondisi penggunaan yang bervariasi. Dengan memantau dan menganalisis metrik-metrik ini, dapat diidentifikasi area-area yang memerlukan perbaikan atau peningkatan kinerja untuk memastikan sistem beroperasi secara efisien dan dapat diandalkan dalam situasi penggunaan yang nyata.

HASIL DAN PEMBAHASAN

Peneltian ini menggunakan metode pengujian *Stress testing* dengan menggunakan Apache Jmeter dan Locust untuk menghadapi beban tinggi. Apache Jmeter dan Locust digunakan untuk mengukur kinerja aplikasi dengan mengirimkan permintaan HTTP dalam jumlah besar secara bersamaan.

A. *Stress Testing* Menggunakan Apache Jmeter

Pada pengujian *Stress testing* menggunakan Apache Jmeter pengujian dilakukan dengan memasukkan jumlah pengguna secara bertahap dari 50, 150 dan 250 dengan periode *ramp up* 5 detik. Setelah jumlah pengguna ditentukan konfigurasi HTTP *Request* dengan memperhatikan *server name*, *localhost* dan *port number* 8080.

a) Hasil *Stress Testing* Apache Jmeter Dengan 50 Pengguna

Berikut adalah hasil pengujian untuk jumlah pengguna 50, pada tiga jenis akses yaitu akses index, periksa anak, dan tambah data secara berurutan dilampirkan pada tabel berikut.

DOI: https://doi.org/10/25047/jtit.v11i1.359

Tabel 1 Hasil Pengujian Apache Jmeter Dengan 50 Pengguna

Avg	Med	90% Line	95% Line	99% Line	Min
40997	42502	56636	57788	59645	3391
61362	61212	63205	63319	64854	49622
56044	56249	57307	57370	60769	54010
52801	56591	62432	62998	64406	3391
Max		error	Throug	Receive	Send
59645		0.00%	0.77395	21.63	0.19
64854		0.00%	0.41268	11.54	0.23
60769		0.00%	0.39748	11.11	0.11
64854		0.00%	0.83886	23.45	0.3

Dalam hasil pengujian ini, terdapat tiga jenis proses yang diuji yaitu akses index, periksa anak, dan tambah data secara berurutan. Untuk akses index terdapat 50 sampel yang memiliki waktu respons rata-rata sebesar 3.391 milidetik, dengan waktu respons median mencapai 40.997 milidetik. Persentase tertinggi dari sampel (90%) memiliki waktu respons di bawah 42.502 milidetik, sementara persentase tertinggi (95%) dan (99%) masing-masing adalah 56.636 milidetik dan 57.788 milidetik. Waktu respons tercepat tercatat sebesar 5.9645 milidetik, dan tidak terdapat kesalahan selama pengujian. Aplikasi mampu memproses permintaan dengan *throughput* rata-rata sebesar 0.7739458857036716 permintaan per detik, dan menerima serta mengirim data dengan kecepatan rata-rata 21.631182860194414 KB per detik dan 0.18668421266485047 KB per detik. Untuk periksa anak dan tambah data masing-masing memiliki 50 sampel dengan waktu respons rata-rata 54.010 milidetik dan 49.622 milidetik. Persentase tertinggi dari sampel (90%) memiliki waktu respons di bawah 56.249 milidetik dan 61.212 milidetik untuk periksa anak dan tambah data secara berurutan. Waktu respons tercepat tercatat sebesar 60.769 milidetik untuk periksa anak dan 64.854 milidetik untuk tambah data. Tidak ada kesalahan yang terjadi selama pengujian untuk kedua jenis akses ini. Throughput rata-rata untuk periksa anak adalah 0.39747523729271667 permintaan per detik, sedangkan untuk tambah data adalah 0.41268085738574933 permintaan per detik. Mereka menerima dan mengirim data dengan kecepatan rata-rata 11.109122354802295 KB per detik dan 11.535719591611024 secara berurutan. Secara keseluruhan, dengan total 150 sampel untuk ketiga jenis akses ini, waktu respons rata-rata adalah 3.391 milidetik, 52.801 milidetik, dan 56.591 milidetik untuk akses index, periksa anak dan tambah data secara berurutan. Persentase tertinggi dari sampel (90%) memiliki waktu respons di bawah 56.591 m ilidetik. Tidak ada kesalahan yang terjadi selama pengujian, dengan throughput rata-rata sebesar 0.8388604919077924 permintaan per detik. Aplikasi juga mampu menerima data dengan kecepatan rata-rata 23.446587655329 KB per detik dan mengirim data dengan kecepatan rata-rata 0.29764255735009565 KB per detik.

b) Hasil *Stress Testing* Jmeter Dengan 150 Pengguna

Berikut adalah hasil pengujian untuk jumlah pengguna 150, pada tiga jenis akses yaitu akses index, periksa anak, dan tambah data secara berurutan dilampirkan pada tabel berikut.

Tabel 2 Hasil Pengujian Apache Jmeter Dengan 150 Pengguna

Avg	Med	90% Line	95% Line	99% Line	Min
98394	99248	129666	135246	141596	13221
153511	153786	156888	157104	157282	142893
130542	130134	134305	134495	141500	126705
127482	130482	154897	156414	157214	13221
Max		error	Throug	Receive	Send
142233		0.00%	1.01892	28.48	0.25
157299		0.00%	0.5377	15.03	0.3
142847		0.00%	0.55413	15.49	0.15
157299		0.00%	1.05433	29.47	0.37

Dalam pengujian ini dengan 150 sampel permintaan HTT. Untuk akses index terdapat 50 sampel yang memiliki waktu respons rata-rata sebesar 98.394 milidetik, dan waktu respons median mencapai 99.248 milidetik. Persentase tertinggi dari sampel (90%) memiliki waktu respons di bawah 129.666 milidetik, sementara persentase tertinggi (95%) dan (99%) masing-masing adalah 135.246 milidetik dan 141.596 milidetik. Waktu respons tercepat tercatat sebesar 13.221 milidetik, dan tidak terdapat kesalahan selama pengujian. Aplikasi mampu memproses permintaan dengan throughput rata-rata sebesar 1.01892 permintaan per detik. Selama pengujian, aplikasi juga mampu menerima data dengan kecepatan rata-rata 28.48 KB per detik dan mengirim data dengan kecepatan rata-rata 0.25 KB per detik.

Untuk tambah data, juga dengan 150 sampel hasil waktu respons rata-rata sebesar 153.511 milidetik, dan waktu respons median mencapai 153.786 milidetik. Persentase tertinggi dari sampel (90%) memiliki waktu respons di bawah 156.888 milidetik, sementara persentase tertinggi (95%) dan (99%) masing-masing adalah 157.104 milidetik dan 157.282 milidetik. Waktu respons tercepat tercatat sebesar 142.893 milidetik, dan tidak terdapat kesalahan selama pengujian. Aplikasi mampu memproses permintaan dengan throughput rata-rata sebesar 0.5377 permintaan per detik. Selama pengujian, aplikasi juga mampu menerima data dengan kecepatan rata-rata 15.03 KB per detik dan mengirim data dengan kecepatan rata-rata 0.3 KB per detik.

Kemudian pada periksa anak juga terdapat 150 sampel dengan waktu respons rata-rata sebesar 130.542 milidetik, dan waktu respons median mencapai 130.134 milidetik. Persentase tertinggi dari sampel (90%) memiliki waktu respons di bawah 134.305 milidetik, sementara persentase tertinggi (95%) dan (99%) masing-masing adalah 134.495 milidetik dan 141.500 milidetik. Waktu respons tercepat tercatat sebesar 126.705 milidetik, dan tidak terdapat kesalahan selama pengujian. Aplikasi mampu memproses permintaan dengan throughput rata-rata sebesar 0.55413 permintaan per detik. Selama pengujian, aplikasi juga mampu menerima data dengan kecepatan rata-rata 15.49 KB per detik dan mengirim data dengan kecepatan rata-rata 0.15 KB per detik.

Secara keseluruhan, dengan total 450 sampel untuk ketiga jenis akses ini, waktu respons rata-rata adalah 127.482 milidetik, 130.482 milidetik, dan 154.897 milidetik untuk akses index, tambah data dan periksa anak secara berurutan. Persentase tertinggi dari sampel (90%) memiliki waktu respons di bawah 154.897 milidetik. Tidak ada kesalahan

DOI: https://doi.org/10/25047/jtit.v11i1.359

yang terjadi selama pengujian, dengan throughput rata-rata sebesar 1.05433 permintaan per detik. Aplikasi juga mampu menerima data dengan kecepatan rata-rata 29.47 KB per detik dan mengirim data dengan kecepatan rata-rata 0.37 KB per detik.

c) Hasil *Stress Testing* Jmeter Dengan 250 Pengguna

Setelah dilakukan pengujian dengan jumlah 50 dan 150 pengguna, pengujian dilanjutkan dengan menaikkan jumlah pengguna menjadi 250. Penaikan jumlah pengguna ini bertujuan untuk mengevaluasi respons aplikasi SIHARAPAN dalam menghadapi beban yang lebih besar lagi, Berikut adalah hasil pengujian untuk jumlah pengguna 250, pada tiga jenis akses yaitu akses index, periksa anak, dan tambah data secara berurutan dilampirkan pada tabel berikut.

Tabel 3 Hasil Pengujian Apache Jmeter Dengan 250 Pengguna

Avg	Med	90% Line	95% Line	99% Line	Min
77990	95154	144725	150621	156300	3391
92672	142297	144225	144657	144797	49622
99595	144650	162215	162808	163002	54010
90085	132591	155507	161410	162925	3391
Max		error	Throug	Receive	Send
157268		35.20%	1.54835	29.69	0.27
144848		35.20%	0.83221	15.83	0.3
163035		35.20%	0.53967	10.26	0.09
163035		35.20%	1.60414	30.59	0.38

Dalam pengujian ini dengan 250 sampel permintaan HTTP. Untuk akses index memiliki waktu respons rata-rata sebesar 77.990 milidetik, dan waktu respons median mencapai 95.154 milidetik. Persentase tertinggi dari sampel (90%) memiliki waktu respons di bawah 144.725 milidetik, sementara persentase tertinggi (95%) dan (99%) masing-masing adalah 150.621 milidetik dan 156.300 milidetik. Waktu respons tercepat tercatat sebesar 1 milidetik, dan waktu respons terlama mencapai 157.268 milidetik. Terdapat kesalahan selama pengujian sebesar 35.20%. Aplikasi mampu memproses permintaan dengan throughput rata-rata sebesar 1.54835 permintaan per detik. Selama pengujian, aplikasi juga mampu menerima data dengan kecepatan rata-rata 29.69 KB per detik dan mengirim data dengan kecepatan rata-rata 0.27 KB per detik.

Kemudian pada proses tambah data, menghasilkan waktu respons rata-rata sebesar 92.672 milidetik, dan waktu respons median mencapai 142.297 milidetik. Persentase tertinggi dari sampel (90%) memiliki waktu respons di bawah 144.225 milidetik, sementara persentase tertinggi (95%) dan (99%) masing-masing adalah 144.657 milidetik dan 144.797 milidetik. Waktu respons tercepat tercatat sebesar 1 milidetik, dan waktu respons terlama mencapai 144.848 milidetik. Terdapat kesalahan selama pengujian sebesar 35.20%. Aplikasi mampu memproses permintaan dengan throughput rata-rata sebesar 0.83221 permintaan per detik. Selama pengujian, aplikasi juga mampu menerima data dengan kecepatan rata-rata 15.83 KB per detik dan mengirim data dengan kecepatan rata-rata 0.3 KB per detik.

Pada proses periksa anak juga terdapat 150 sampel dengan waktu respons rata-rata sebesar 99.595 milidetik, dan waktu respons median mencapai 144.650 milidetik. Persentase tertinggi dari sampel (90%) memiliki waktu respons di bawah 162.215 milidetik, sementara persentase tertinggi (95%) dan (99%) masing-masing adalah 162.808 milidetik dan 163.002 milidetik. Waktu respons tercepat tercatat sebesar 1 milidetik, dan waktu respons terlama mencapai 163.035 milidetik. Terdapat kesalahan selama pengujian sebesar 35.20%. Aplikasi mampu memproses permintaan dengan throughput rata-rata sebesar 0.53967 permintaan per detik. Selama pengujian, aplikasi juga mampu menerima data dengan kecepatan rata-rata 10.26 KB per detik dan mengirim data dengan kecepatan rata-rata 0.09 KB per detik.

Secara keseluruhan, dengan total 750 sampel untuk ketiga jenis akses ini, waktu respons rata-rata adalah 90.085 milidetik, 132.591 milidetik, dan 155.507 milidetik untuk "Akses Index", "Tambah Data", dan "Periksa Anak" secara berurutan. Persentase tertinggi dari sampel (90%) memiliki waktu respons di bawah 155.507 milidetik. Terdapat kesalahan yang terjadi selama pengujian sebesar 35.20%, dengan throughput rata-rata sebesar 1.60414 permintaan per detik. Aplikasi juga mampu menerima data dengan kecepatan rata-rata 30.59 KB per detik dan mengirim data dengan kecepatan rata-rata 0.38 KB per detik.

B) *Stress Testing* Menggunakan Locust

Setelah melakukan *Stress testing* menggunakan Apache JMeter, skenario pengujian yang sama juga dilakukan dengan menggunakan *tools* Locust. Dalam konfigurasi locust, *path* dan jenis permintaan HTTP harus ditentukan dalam definisi tugas (*task)*. Setiap tugas akan diberikan kepada setiap pengguna sesuai dengan konfigurasi yang di tentukan, dan meskipun jumlah pengguna telah mencapai batas yang ditetapkan, jumlah permintaan terus bertambah karena setiap pengguna terus menjalankan tugas-tugas mereka. Dalam kasus ini, jumlah pengguna simultan dan periode *ramp-up* yang digunakan pada pengujian dengan Locust menggunakan skenario yang sama seperti sebelumnya, yaitu 50, 150, dan 250 pengguna dengan *ramp-up* 5 detik. Berikut adalah hasil pengujian pertama dari pengujian yang dilakukan untuk jumlah pengguna sebanyak 50.

a) Hasil *Stress Testing* Locust Dengan 50 Pengguna

Berikut adalah hasil pengujian locust dengan jumlah pengguna sebanyak 50 pengguna, pada tiga jenis akses yaitu akses index, periksa anak, dan tambah data secara berurutan dilampirkan pada tabel berikut..

Tabel 4 Hasil Pengujian Locust Dengan 50 Pengguna

Req	Fails	Med	95%-ile (ms)	99%-ile (ms)	Avg
53	0	48000	50000	50000	42929.80
94	0	48000	50000	50000	42777.85
64	0	48000	50000	50000	44841.11
171	0	48000	50000	50000	43016.5
Min (ms)		Max (ms)	Average size (bytes)	Current RPS	Current Failures
3639		50547	27048	0.2	3639
3573		50514	27048	0.3	3573
1712		50572	27048	0.6	1712
3639		50572	27048	0.8	3639

DOI: https://doi.org/10.25047/jtit.v11i1.359

Pada hasil pengujian menggunakan locust, terdapat tiga jenis akses yang diuji yaitu akses index, periksa anak, dan tambah data. Pada proses akses halaman index terjadi 53 permintaan yang dilakukan, tidak ada permintaan yang gagal. Waktu respons median adalah 48.000 milidetik, sementara persentase tertinggi (95%) dan (99%) masing-masing adalah 50.000 milidetik dan 51.000 milidetik. Waktu respons rata-rata adalah 42.972,89 milidetik, dengan waktu respons tercepat sebesar 3.039 milidetik dan waktu respons terlama mencapai 50.541 milidetik. Ukuran rata-rata respons adalah 27.498 byte. Saat pengujian berlangsung, permintaan rata-rata per detik (RPS) adalah 0,2, tanpa adanya kegagalan. dari total 54 permintaan yang dilakukan, tidak ada permintaan yang gagal. Waktu respons median adalah 48.000 milidetik, sementara persentase tertinggi (95%) dan (99%) masing-masing adalah 50.000 milidetik dan 51.000 milidetik. Waktu respons rata-rata adalah 42.777,53 milidetik, dengan waktu respons tercepat sebesar 3.575 milidetik dan waktu respons terlama mencapai 50.572 milidetik. Ukuran rata-rata respons adalah 27.498 byte. Saat pengujian berlangsung, permintaan rata-rata per detik (RPS) adalah 0, tanpa adanya kegagalan. Sementara itu, pada permintaan POST ke proses tambah data, dari total 64 permintaan yang dilakukan, tidak ada permintaan yang gagal. Waktu respons median adalah 48.000 milidetik, sementara persentase tertinggi (95%) dan (99%) masing-masing adalah 50.000 milidetik dan 50.000 milidetik. Waktu respons rata-rata adalah 44.841,61 milidetik, dengan waktu respons tercepat sebesar 17.572 milidetik dan waktu respons terlama mencapai 50.373 milidetik. Ukuran rata-rata respons adalah 27.498 byte. Saat pengujian berlangsung, permintaan rata-rata per detik (RPS) adalah 0,6, tanpa adanya kegagalan.

Secara agregat, dari total 171 sampel permintaan, tidak ada permintaan yang gagal. Waktu respons median adalah 48.000 milidetik, sementara persentase tertinggi (95%) dan (99%) masing-masing adalah 50.000 milidetik dan 51.000 milidetik. Waktu respons rata-rata adalah 43.610,6 milidetik, dengan waktu respons tercepat sebesar 3.039 milidetik dan waktu respons terlama mencapai 50.572 milidetik. Ukuran rata-rata respons adalah 27.498 byte. Saat pengujian berlangsung, permintaan rata-rata per detik (RPS) adalah 0,8, tanpa adanya kegagalan.

b) Hasil *Stress Testing* Locust Dengan 150 Pengguna

Setelah dilakukan pengujian dengan jumlah 50 pengguna, pengujian dilanjutkan dengan menaikkan jumlah pengguna menjadi 150. Berikut adalah hasil dari pengujian yang dilakukan untuk jumlah pengguna sebanyak 150 pada tiga jenis akses yaitu akses index, periksa anak, dan tambah data secara berurutan dilampirkan pada tabel berikut.

Tabel 5 Hasil Pengujian Locust Dengan 150 Pengguna

Req	Fails	Med	95%-ile (ms)	99%-ile (ms)	Avg
129	0	150000	190000	190000	149115.21
132	0	150000	190000	190000	145064.43
184	0	150000	190000	190000	44841.11
499	0	150000	190000	190000	139781.11
Min (ms)		Max (ms)	Average size (bytes)	Current RPS	Current Failures
4035		195239	27048	0.4	0
4364		195779	27048	0.3	0
2192		195824	27048	0.6	0
2192		195693	27048	1.5	0

Setelah jumlah akses pengguna ditingkatkan menjadi 150 dengan ramp up 5 detik, untuk akses halaman index terdapat 159 permintaan tanpa kegagalan. Waktu respons median adalah 151.000 milidetik, sementara persentase tertinggi (95%) dan (99%) masing-masing adalah 159.000 milidetik dan 160.000 milidetik. Waktu respons rata-rata adalah 138.119,21 milidetik, dengan waktu respons tercepat sebesar 43.069 milidetik dan waktu respons terlama mencapai 159.772 milidetik. Ukuran rata-rata respons adalah 27.498 byte. Saat pengujian berlangsung, permintaan rata-rata per detik (RPS) adalah 0,4, tanpa kegagalan. Kemudian, pada pengujian proses hitung gizi terdapat 152 permintaan tanpa kegagalan. Waktu respons median adalah 152.000 milidetik, sementara persentase tertinggi (95%) dan (99%) masing-masing adalah 159.000 milidetik dan 160.000 milidetik. Waktu respons rata-rata adalah 142.665,43 milidetik, dengan waktu respons tercepat sebesar 43.654 milidetik dan waktu respons terlama mencapai 159.895 milidetik. Ukuran rata-rata respons adalah 27.498 byte. Saat pengujian berlangsung, permintaan rata-rata per detik (RPS) adalah 0,3, tanpa kegagalan.

Sementara itu, untuk proses tambah data terdapat 184 permintaan tanpa kegagalan. Waktu respons median adalah 151.000 milidetik, sementara persentase tertinggi (95%) dan (99%) masing-masing adalah 159.000 milidetik dan 160.000 milidetik. Waktu respons rata-rata adalah 138.806,8 milidetik, dengan waktu respons tercepat sebesar 21.302 milidetik dan waktu respons terlama mencapai 159.669 milidetik. Ukuran rata-rata respons adalah 27.498 byte. Saat pengujian berlangsung, permintaan rata-rata per detik (RPS) adalah 0,8, tanpa kegagalan. Secara agregat, dari total 495 sampel permintaan, tidak ada kegagalan yang terjadi. Waktu respons median adalah 151.000 milidetik, sementara persentase tertinggi (95%) dan (99%) masing-masing adalah 159.000 milidetik dan 160.000 milidetik. Waktu respons rata-rata adalah 139.770,81 milidetik, dengan waktu respons tercepat sebesar 21.302 milidetik dan waktu respons terlama mencapai 159.895 milidetik. Ukuran rata-rata respons adalah 27.498 byte. Saat pengujian berlangsung, permintaan rata-rata per detik (RPS) adalah 1,5, tanpa kegagalan.

c) Hasil *Stress Testing* Locust Dengan 250 Pengguna

Setelah dilakukan pengujian dengan jumlah 50 dan 150 pengguna, pengujian dilanjutkan dengan menaikkan jumlah pengguna menjadi 250. Berikut adalah hasil dari pengujian yang dilakukan untuk jumlah pengguna sebanyak 250 pada tiga jenis akses yaitu akses index, periksa anak, dan tambah data secara berurutan dilampirkan pada tabel berikut.

Tabel 6 Hasil Pengujian Locust Dengan 250 Pengguna

Req	Fails	Med	95%-ile (ms)	99%-ile (ms)	Avg
129	0	150000	190000	190000	149115.21
132	0	150000	190000	190000	145064.43
184	0	150000	190000	190000	139901.8
499	0	150000	190000	190000	139781.11
Min (ms)		Max (ms)	Average size (bytes)	Current RPS	Current Failures)
4035		195239	27048	0.4	0

DOI: https://doi.org/10/25047/jtit.v11i1.359

4364	195779	27048	0.3	0
2192	195824	27048	0.5	0
2192	195693	27048	1.5	0

Setelah jumlah akses pengguna ditingkatkan menjadi 250 dengan ramp up 5 detik, pada proses akses halaman index dari total 744 permintaan yang dilakukan, terdapat 716 permintaan yang gagal. Waktu respons median adalah 4600 milidetik, sementara persentase tertinggi (95%) dan (99%) masing-masing adalah 118.000 milidetik dan 137.000 milidetik. Waktu respons rata-rata adalah 19.727,47 milidetik, dengan waktu respons tercepat sebesar 2.965 milidetik dan waktu respons terlama mencapai 231.743 milidetik. Ukuran rata-rata respons adalah 1034.87 byte. Saat pengujian berlangsung, permintaan rata-rata per detik (RPS) adalah 1, dengan tingkat kegagalan juga 1 permintaan per detik. Kemudian untuk pengujian pada proses hitung gizi dari total 751 permintaan yang dilakukan, terdapat 720 permintaan yang gagal. Waktu respons median adalah 4600 milidetik, sementara persentase tertinggi (95%) dan (99%) masing-masing adalah 119.000 milidetik dan 151.000 milidetik. Waktu respons rata-rata adalah 20.431,11 milidetik, dengan waktu respons tercepat sebesar 4.035 milidetik dan waktu respons terlama mencapai 229.584 milidetik. Ukuran rata-rata respons adalah 1135.07 byte. Saat pengujian berlangsung, permintaan rata-rata per detik (RPS) adalah 1.6, dengan tingkat kegagalan juga 1.6 permintaan per detik.

Sementara itu, untuk proses tambah data dengan permintaan POST dari total 715 permintaan yang dilakukan, terdapat 690 permintaan yang gagal. Waktu respons median adalah 4600 milidetik, sementara persentase tertinggi (95%) dan (99%) masing-masing adalah 120.000 milidetik dan 143.000 milidetik. Waktu respons rata-rata adalah 19.680,9 milidetik, dengan waktu respons tercepat sebesar 4.032 milidetik dan waktu respons terlama mencapai 212.685 milidetik. Ukuran rata-rata respons adalah 961.47 byte. Saat pengujian berlangsung, permintaan rata-rata per detik (RPS) adalah 1.4, dengan tingkat kegagalan juga 1.4 permintaan per detik. Secara agregat, dari total 2.210 sampel permintaan, terdapat 2.126 permintaan yang gagal. Waktu respons median adalah 4600 milidetik, sementara persentase tertinggi (95%) dan (99%) masing-masing adalah 118.000 milidetik dan 142.000 milidetik. Waktu respons rata-rata adalah 19.951,52 milidetik, dengan waktu respons tercepat sebesar 2.965 milidetik dan waktu respons terlama mencapai 231.743 milidetik. Ukuran rata-rata respons adalah 1045.17 byte. Saat pengujian berlangsung, permintaan rata-rata per detik (RPS) adalah 4, dengan tingkat kegagalan juga 4 permintaan per detik.

C) Rekomendasi Perbaikan Aplikasi

Berdasarkan hasil pengujian *stress testing* menggunakan Apache JMeter dan Locust, aplikasi menunjukkan kinerja yang baik dengan 50 hingga 150 pengguna, tetapi mengalami masalah ketika jumlah pengguna mencapai 250, dengan kesalahan sebesar 35,20%. Untuk meningkatkan kinerja dan keandalan aplikasi, disarankan beberapa langkah yaitu, menambah kapasitas hardware server, menggunakan load balancer, dan mekanisme caching untuk mengurangi beban server. Optimasi query database dan perbaikan kode juga diperlukan untuk meningkatkan efisiensi aplikasi. Selain itu, penambahan server untuk mendukung lebih banyak pengguna. Penting juga untuk memantau kinerja aplikasi secara *real-time* dan menguji perubahan kode sebelum diterapkan. Terakhir, penggunaan mekanisme penanganan kesalahan yang lebih baik dan pembatasan penggunaan (*rate limiting*) akan membantu menjaga stabilitas aplikasi dan mencegah serangan.

KESIMPULAN

Berdasarkan hasil pengujian *stress testing* yang dilakukan, ketahanan dan kehandalan aplikasi SIHARAPAN sudah cukup memadai. Aplikasi mampu menangani beban tinggi dalam beberapa skenario pengujian, meskipun waktu respons bervariasi secara signifikan dan meningkat seiring dengan peningkatan jumlah pengguna. Pada skenario dengan 250 pengguna bersamaan, peningkatan jumlah pengguna tidak selalu diimbangi dengan peningkatan throughput secara linier dan terdapat beberapa error yang terjadi. Namun, error tersebut masih dalam batas yang dapat diterima. Secara umum, aplikasi SIHARAPAN menunjukkan ketahanan dan kehandalan yang memadai dalam situasi tekanan tinggi, sehingga memenuhi tujuan penelitian untuk memastikan ketahanan dan kehandalan aplikasi melalui *stress testing*. Penelitian selanjutnya dapat difokuskan pada optimasi lebih lanjut untuk mengurangi variasi waktu respons dan meningkatkan throughput pada jumlah pengguna yang lebih tinggi.

REFERENSI

[1] A. Zulianto et al., “Pemanfaatan Katalon Studio untuk Otomatisasi Pengujian Black-Box pada Aplikasi iPosyandu,” Jepin (Jurnal Edukasi Dan Penelitian Informatika), vol. 7(3), no. 3, pp. 370–378, 2021.

[2] R. Wirawan, A. D. W. Rahmadana, A. A. Wibowo, P. I. Wardhani, S. Bachri, and M. Muhajir, “Performa dan Stress Testing dalam Upaya Mengoptimalkan WebGIS Open Source Studi Kasus WebGIS Ekowisata Sungai Mudal Kulon Progo,” Jurnal Geomatika, vol. 27, no. 1, pp. 19–26, 2021.

[3] A. M. Ramadhani, A. Rakhmatsyah, and R. Yasirandi, “Analysis and Implementation of Microservice Architecture Related to Patient Drug Schedule Based on FHIR Standard,” Jurnal Ilmiah Teknik Elektro Komputer dan Informatika, vol. 7, no. 2, pp. 296–305, 2021.

[4] Y. Arta, R. Wandri, A. Hanafiah, B. K. Pranoto, and M. R. Fadhilah, “Analisa Perbandingan Web Server Untuk Kebutuhan Open Journal System (OJS) Menggunakan Secure Tunnel,” CogITo Smart Journal, vol. 8, no. 2, pp. 537–548, 2022.

[5] J. Sukarni and H. Jati, “Pengembangan Sistem Informasi Kemitraan Sekolah dengan Orang Tua Berdasarkan Epstein’s Framework,” J. Edukasi dan Penelit. Inform., vol. 6, no. 3, pp. 408–416, 2020.

[6] N. L. A. S. Ginasari, K. S. Wibawa, and N. K. A. Wirdiani, “Pengujian Stress Testing API Sistem Pelayanan dengan Apache JMeter,” Jurnal Edukasi dan Penelitian Informatika, vol. 2, no. 3, 2021.

[7] A. R. Putri, R. Munadi, and R. M. Negara, “Performance analysis of multi services on container Docker, LXC, and LXD,” Bulletin of Electrical Engineering and Informatics, vol. 9, no. 5, pp. 2008–

2011, 2020.

[8] O. Johnson and O. Dinyo, “Comparative Analysis of Single-Core and Multi-Core Systems,” International Journal of Computer Science and Information Technology, vol. 7, no. 6, pp. 117–130, 2015.

[9] M. Saputra and R. M. Fadlila, “An Effective Open ERP System for Automation in Financial Reporting for SMEs based on Service Oriented Architecture,” International Journal on Informatics Visualization, vol. 7, no. 3–2, pp. 207–215, 2023.

[10] Indrianto, “Performance Testing on Web Information System Using Apache Jmeter and Blazemeter,” Jurnal Ilmiah Ilmu Terapan Universitas Jambi, vol. 7, no. 2, pp. 138–149, 2023.

[11] B. Z. Ferdiyan, E. S. Nugroho, T. Informatika, T. Informasi, and P. C. Riau, “Sistem Informasi Rekapitulasi Pemilukada Kota Pekanbaru menggunakan Input dari Telegram API,” Rekayasa Sistem dan Teknologi Informasi, vol. 4, no. 1, pp. 56–63, 2020.

[12] D. Madhani, E. Darwiyanto, and A. Ghandi, “Performance Testing Menggunakan Metode Load Testing dan Stress Testing pada Sistem Core Banking PT. XYZ,” e-Proceeding of Engineering, vol. 10, no. 6, pp. 1–11, 2023.

[13] A. R. Sofyan and S. D. Y. Kusuma, “Implementasi Load Balancing Web Server menggunakan Haproxy pada Virtual Server Direktorat SMK Kemendikbudristek,” Jurnal Pendidikan Tambusai, vol. 6, pp. 9669–9682, 2022.

[14] A. Muttaqin and S. R. Akbar, “Web Server Embedded System,” JJurnal Teknologi Informasi dan Ilmu Komputer (JTIIK), vol. Muttaqin, no. 1, pp. 50–54, 2014.

DOI: https://doi.org/10/25047/jtit.v11i1.359