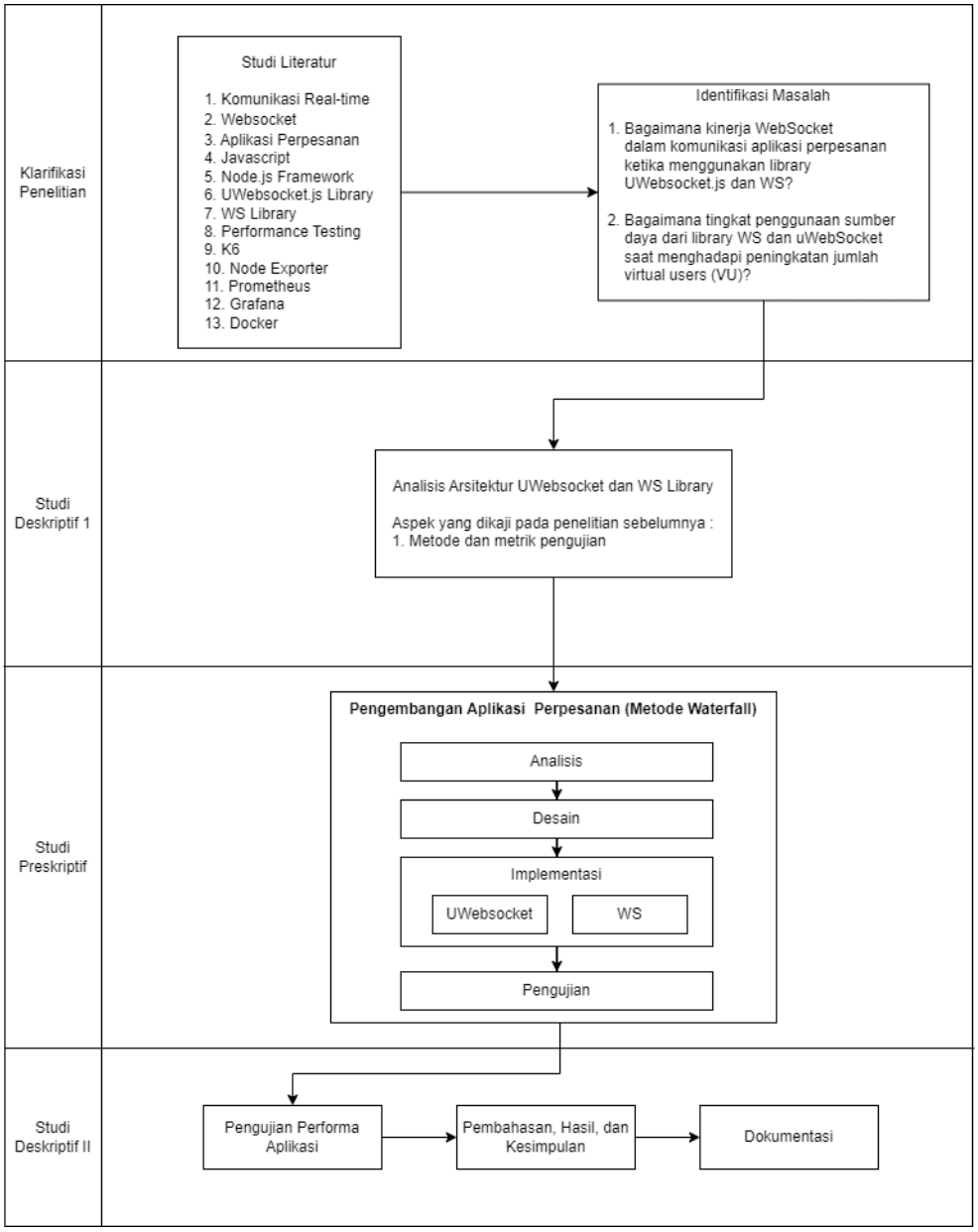
BAB III
METODE PENELITIAN

3.1 Desain Penelitian

Alur metodologi penelitian ini berdasarkan *Design Research and Methodology (DRM).* Metode ini merupakan suatu pendekatan yang digunakan untuk mengkaji, mengembangkan, dan menguji berbagai desain dalam konteks inovasi dan pemecahan masalah. Metode ini terutama digunakan dalam bidang desain, seni, dan teknik, tetapi juga dapat diterapkan pada berbagai disiplin ilmu lainnya. *Design Research and Methodology (DRM)* menekankan proses desain sebagai inti penelitian. Tujuannya adalah untuk menciptakan sesuatu yang baru, menyelesaikan masalah, atau meningkatkan desain produk, layanan, sistem, atau pengalaman pengguna.

DRM mengembangkan dan memeriksa berbagai pengetahuan, metode yang digunakan, dan alat yang bisa digunakan berdasarkan teori-teori tersebut untuk membuat proses desain menjadi lebih baik. *DRM* terdiri dari 4 tahapan, yaitu Klarifikasi Penelitian atau Research Clarification (RC), Studi Deskriptif I atau Descriptive Study I (DS-I), Studi Preskriptif atau Prescriptive Study (PS), dan Studi Deskriptif II atau Descriptive Study II (DS-II). Setiap tahap memiliki tujuan dan metode yang berbeda, dan tahapan-tahapan ini saling terkait dan saling melengkapi untuk menghasilkan solusi desain yang efektif.

Model *DRM* ini dipilih karena menyediakan kerangka kerja dalam penelitian desain, membantu mengidentifikasi bidang penelitian dan mengembangkan argumentasi. Jika digunakan secara fleksibel, metodologi ini akan membantu membuat penelitian desain mendapatkan solusi baru menurut Blessing dan Chakrabarti pada penelitian Lawrie dkk., 2024. Berikut merupakan proses alur metodologi penelitian.



Gambar 3.1 Metodologi Penelitian

3.1.1 Klarifikasi Penelitian

Tahap ini melibatkan klarifikasi penelitian, termasuk mengidentifikasi tujuan penelitian, jenis penelitian, dan rencana penelitian secara keseluruhan. Dalam tahap ini, melibatkan studi literatur yang bertujuan untuk memahami konteks penelitian dan melihat apa yang telah dikemukakan oleh peneliti-peneliti sebelumnya dalam bidang yang sama.

Teori yang dikaji dalam studi literatur diantaranya adalah Komunikasi *Real-time*, WebSocket, Aplikasi Perpesanan, Javascript, Node.js *Framework*, uWebSocket.js Libary, WS *Library*, *Performance testing*, K6. teori-teori tersebut didapatkan dari berbagai artikel, jurnal, dan lain lain untuk mengidentifikasi masalah tersebut.

3.1.2 Studi Deskriptif 1

Melalui tahap sebelumnya tujuan penelitian, jenis penelitian, dan rencana keseluruhan telah diidentifikasi secara sistematis. Pendekatan ini memungkinkan peneliti untuk menetapkan landasan yang kokoh sebelum memasuki ranah penelitian yang lebih mendalam. Studi literatur yang dilakukan sebelumnya memberikan pemahaman penting tentang konteks penelitian yang telah dilakukan oleh peneliti peneliti-peneliti sebelumnya.

Pada tahap ini fokus akan ditujukan pada dua aspek utama, yaitu analisis mendalam terhadap arsitektur uWebSocket.js dan WS *Library* serta deskripsi tentang aplikasi perpesanan. Analisis ini bertujuan untuk memahami struktur dan fungsi masing-masing *library* secara terperinci, serta mengidentifikasi kekuatan dan kelemahan yang mungkin mempengaruhi implementasi aplikasi perpesanan. Selain itu, studi ini juga akan memberikan deskripsi rinci tentang aplikasi perpesanan dan memperjelas bagaimana aplikasi ini berinteraksi dengan kedua *library* tersebut dalam konteks komunikasi *real-time*.

3.1.3 Studi Preskriptif

Pada tahap ini, pendekatan metode yang digunakan adalah metode waterfall yang digunakan untuk membangun kedua aplikasi. Metode waterfall adalah proses pengembangan perangkat lunak secara bertahap, di mana setiap tahap harus diselesaikan sebelum melanjutkan ke tahap berikutnya. Proses ini melibatkan serangkaian fase yang harus dilalui secara berurutan untuk membangun perangkat lunak komputer dengan sukses. Pada dasarnya, model waterfall terdiri dari lima fase yaitu analisis, desain, implementasi, pengujian, dan pemeliharaan (Bassil, 2012). Tahap pemeliharaan, yang merupakan tahap terakhir dalam metode waterfall, tidak akan dilakukan dalam pengembangan aplikasi ini karena tujuan penelitian ini berfokus pada pembuktian konsep dan evaluasi kinerja aplikasi.

Dengan menggunakan metode waterfall, pengembangan aplikasi perpesanan akan dilakukan melalui serangkaian langkah yang terstruktur sebagai berikut:

3.1.3.1 Analisis

Tahap analisis memfokuskan pada pemahaman tentang kebutuhan dan tujuan aplikasi. Ini mencakup analisis fitur, alur kerja aplikasi, dan batasan aplikasi. Pengembangan aplikasi ini terdiri dari dua komponen utama yaitu sisi klien dan sisi server. Pada sisi klien, aplikasi dibangun menggunakan HTML, CSS, dan Javascript, dengan kebutuhan untuk mengirim dan menerima pesan secara *real-time*. Aplikasi ini tidak akan memuat sistem login dan room chat. Dengan kata lain, aplikasi ini akan fokus pada fitur komunikasi dasar tanpa implementasi otentikasi pengguna atau penggunaan room chat antar *user*. Selain itu, tidak ada pembedaan antara pengguna yang mengirim pesan dengan pengguna lain. Artinya, semua pesan yang diterima oleh klien akan ditampilkan tanpa mengidentifikasi siapa yang mengirimnya. Batasan ini ditetapkan untuk menyederhanakan pengembangan dan fokus pada evaluasi kinerja komunikasi *real-time* menggunakan uWebSocket.js dan WS *Library*.

Pada sisi server, aplikasi menggunakan *framework* Node.js dan kedua *library* yaitu uWebSocket.js dan WS *Library*, untuk menangani komunikasi WebSocket. Kedua *library* ini akan dijalankan pada server Node.js dengan *script* dasar tanpa penambahan fitur fitur yang ada pada kedua *library* tersebut. Kedua aplikasi, baik sisi klien maupun sisi server, akan dijalankan pada server Linux Ubuntu yang di-*host* di *virtual machine* menggunakan VirtualBox. Pengujian akan dilakukan secara bergantian di *virtual machine* yang sama untuk memastikan bahwa tidak ada perbedaan dalam sumber daya yang digunakan antara pengujian. Hal ini bertujuan untuk memastikan keakuratan dan konsistensi hasil pengujian dengan mengisolasi variabel-variabel lingkungan yang mungkin mempengaruhi kinerja aplikasi.

Pembuatan *script* pengujian yang akan dilakukan menggunakan K6 dirancang untuk menyesuaikan dengan metrik yang telah ditentukan, seperti waktu respon, pemanfaatan sumber daya, dan kapasitas. K6 akan digunakan untuk mengukur kinerja komunikasi *real-time* antara klien dan server dengan menggunakan teknik *load test*. Untuk memastikan konsistensi dalam pengujian kinerja antara kedua

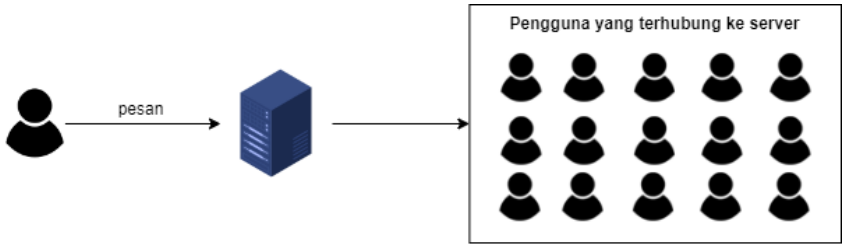
library, skrip pengujian K6 akan dibuat serupa. Skrip K6 ini akan dirancang untuk mengukur metrik waktu respon dan kapasitas. Skrip akan mengeksekusi *load test* dengan skenario yang berbeda untuk kedua *library*. Pengukuran metrik pemanfaatan sumber daya akan dilakukan di Grafana untuk mendapatkan visualisasi dari data yang didapat.

Dalam menyiapkan tampilan dashboard untuk mengamati server, persiapan dimulai dengan menambahkan Node Exporter ke *virtual machine* yang menjalankan server aplikasi. Node Exporter akan mengumpulkan metrik sistem dari server, seperti penggunaan CPU dan memori. Untuk pengelolaan dan pemantauan, akan dibuat *script* Docker untuk menjalankan Grafana dan Prometheus dalam lingkungan kontainerisasi. Dengan menggunakan Docker, setiap komponen dapat dijalankan secara terpisah dalam kontainer. Prometheus akan dikonfigurasi dalam Docker untuk mengumpulkan metrik dari Node Exporter. Grafana akan dikonfigurasi untuk mengambil data dari Prometheus dan menampilkan informasi tersebut dalam bentuk dashboard untuk memvisualisasikan metrik secara *real-time*.

Secara keseluruhan, skrip K6 mencatat metrik waktu respon, tingkat kesalahan, dan jumlah pesan yang dapat diproses. Node Exporter memantau penggunaan sumber daya sistem seperti CPU dan memori, mengirim data ke Prometheus, dengan Grafana menampilkan visualisasi data ini untuk analisis performa.

3.1.3.2 Desain

Selanjutnya, dalam tahap desain, akan dibuat desain untuk aplikasi perpesanan, termasuk struktur arsitektur dan logika aplikasi. Ketika pengguna membuka aplikasi, mereka akan langsung terhubung ke server WebSocket tanpa perlu melalui proses login atau memilih room chat. Pengguna dapat langsung mulai mengirim pesan, yang akan langsung diterima oleh server. Setelah server menerima pesan dari pengguna, server akan mengirimkan kembali pesan tersebut kepada pengguna atau pengguna lainnya yang terhubung ke server seperti yang diilustrasikan dibawah ini.



Gambar 3.2 Logika Aplikasi

3.1.3.3 Implementasi

Tahap implementasi akan melibatkan pengkodean kedua aplikasi berdasarkan desain yang telah dibuat, dengan fokus pada integrasi komunikasi *real-time* dan implementasi fitur-fitur utama.

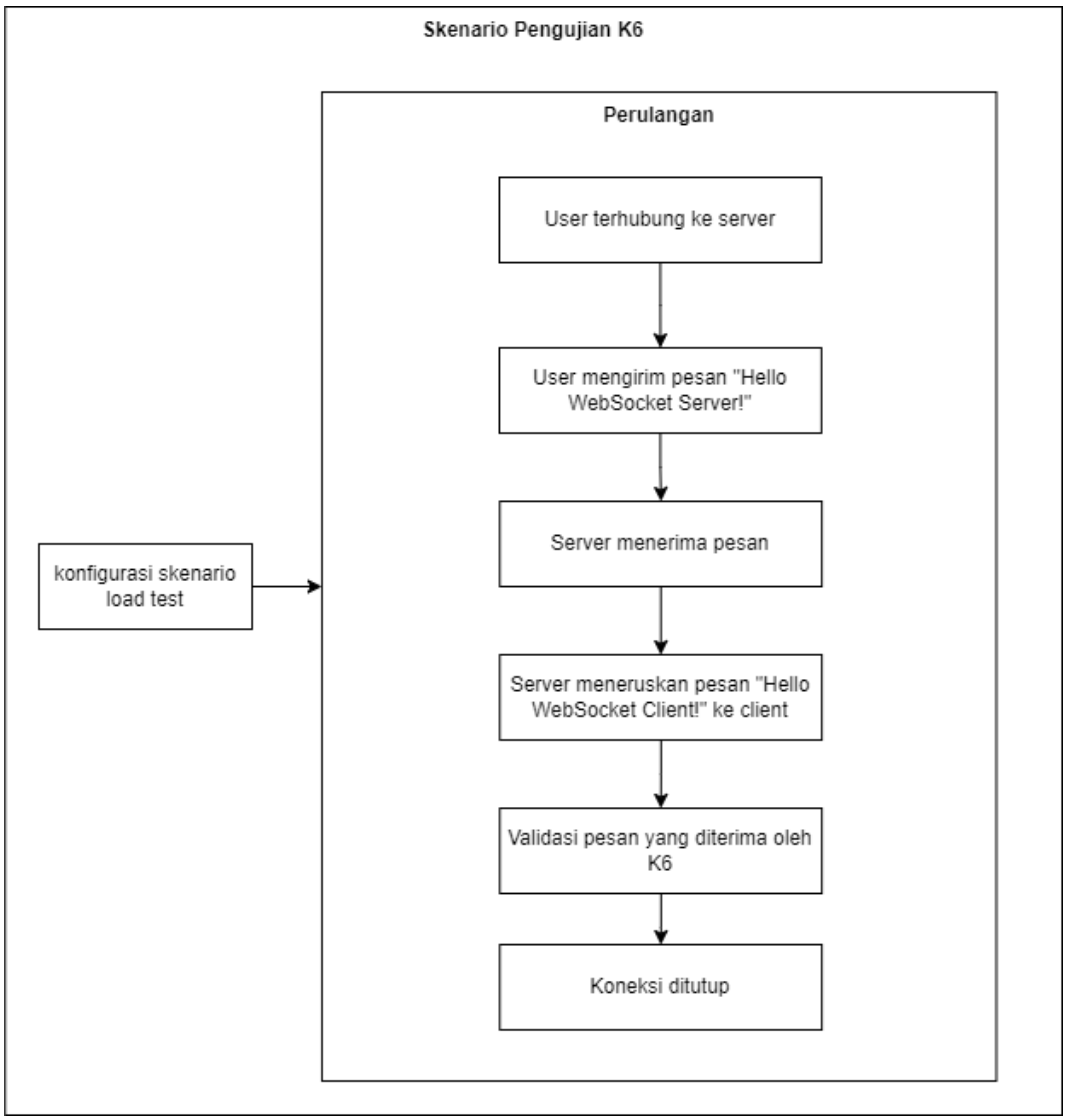
3.1.3.4 Pengujian

Terakhir, tahap pengujian akan menguji kinerja dari aplikasi. Pada tahap ini akan difokuskan pada persiapan lingkungan pengujian, pembuatan *script* pengujian serta pengujian kinerja yang akan menguji berbagai aspek seperti *Time Behavior*, *Resource Utilization*, *Capacity*, dan *Realibility*. *Load test*ing akan digunakan untuk mensimulasikan kondisi beban yang berbeda pada aplikasi.

Seperti yang dijelaskan pada bagian analisis, Pengujian akan dilakukan secara bergantian di *virtual machine* yang sama untuk memastikan bahwa tidak ada perbedaan dalam sumber daya yang digunakan antara pengujian. Dalam pengujian ini, satu server digunakan secara bergantian untuk menguji performa aplikasi yang dikembangkan dengan UWebSocket.js dan WS. Persiapan server mencakup instalasi dan konfigurasi aplikasi yang akan diuji, baik yang menggunakan UWebSocket.js maupun WS. Selain itu, alat monitoring juga dipersiapkan. Node Exporter diinstal pada server, Prometheus kemudian dikonfigurasi untuk mengumpulkan metrik-metrik tersebut dari Node Exporter secara periodik. Data yang dikumpulkan oleh Prometheus ini nantinya akan digunakan untuk memantau performa server selama pengujian berlangsung.

Dalam pengujian *load test*, K6 digunakan untuk mensimulasikan sejumlah klien yang terhubung ke server WebSocket dan mengirim pesan secara kontinu

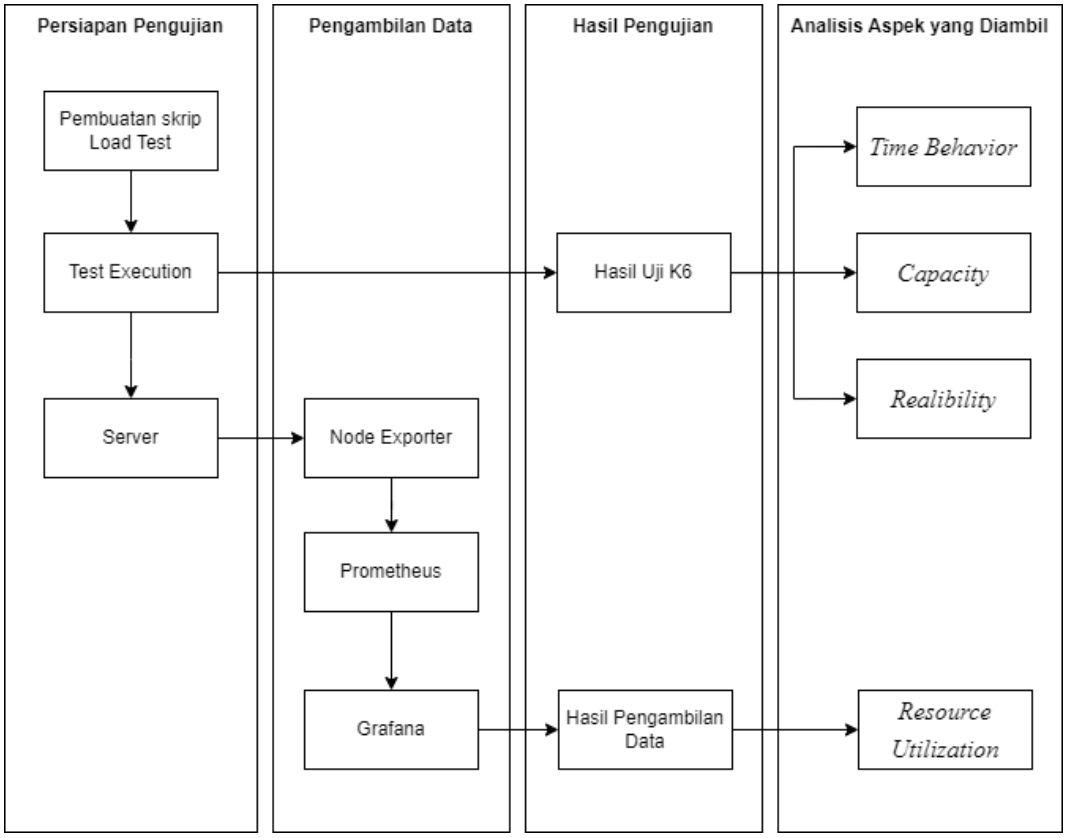
dengan lima tingkatan beban 50, 500, 1000, 2000, 3000 klien (Delta dan Asmunin, 2016). Setiap skenario berlangsung selama 300 detik dengan beban tersebut (Suryawan dan Muliantara, 2024). Hal ini bertujuan untuk menguji performa server dengan berbagai tingkatan beban untuk mengevaluasi bagaimana server menangani peningkatan jumlah klien dan frekuensi pengiriman pesan secara bersamaan.



Gambar 3. 3 Skenario Pengujian

Proses pengujian dimulai dengan konfigurasi skenario di K6, di mana tingkat beban (jumlah klien) dan durasi pengujian ditentukan. Selanjutnya, pengujian dilakukan dalam beberapa iterasi yang melibatkan beberapa langkah. Pertama, klien yang disimulasikan oleh K6 akan terhubung ke server WebSocket. Setelah

terhubung, setiap klien mengirimkan pesan "Hello WebSocket Server!" kepada server. Setelah menerima pesan, server merespons dengan mengirimkan balasan berupa pesan "Hello WebSocket Client!" ke klien. K6 kemudian memvalidasi apakah pesan balasan yang diterima oleh klien sesuai dengan yang diharapkan, yang menandakan bahwa server berhasil menangani interaksi tersebut. Setelah validasi selesai, koneksi WebSocket antara klien dan server ditutup, yang menandai akhir dari satu iterasi pengujian. Setiap langkah ini diulang dalam beberapa iterasi segera setelah 1 iterasi tersebut selesai untuk memastikan bahwa server mampu menangani beban klien yang terus meningkat dengan performa yang stabil.



Gambar 3.4 Alur Pengujian

Proses pengujian performa sistem melibatkan beberapa tahapan yang didesain untuk mengukur berbagai aspek performa. Tahap pertama adalah pengujian, yang dimulai dengan pembuatan skrip untuk pengujian beban (*Load test*). Setelah skrip selesai dibuat, skrip tersebut akan dieksekusi, mengirimkan permintaan ke server sesuai dengan skenario yang telah ditentukan. Server yang

menjadi objek pengujian kemudian akan menerima permintaan dan data performanya akan dipantau untuk mengumpulkan informasi yang dibutuhkan.

Tahap kedua adalah pengambilan data, di mana alat monitoring digunakan untuk mengumpulkan data performa dari server selama pengujian berlangsung. Node Exporter akan mengumpulkan metrik dari server, data ini kemudian dikumpulkan dan disimpan oleh Prometheus, yang mengumpulkan data secara periodik dari Node Exporter. Grafana digunakan untuk visualisasi data tersebut. Dengan Grafana, data performa server dapat divisualisasikan dalam bentuk dashboard yang menunjukkan metrik-metrik tersebut secara *real-time* selama pengujian berlangsung.

Hasil pengujian melibatkan dua aspek utama yaitu hasil uji K6 dan hasil pengambilan data dari Grafana. Hasil dari pengujian beban dengan K6 memberikan berbagai metrik performa seperti *latency, throughput*, dan *error rate*, yang digunakan untuk menganalisis performa server. Sementara itu, data yang dikumpulkan oleh Prometheus divisualisasikan dengan Grafana untuk menunjukkan penggunaan sumber daya sistem selama pengujian berlangsung.

Tahapan terakhir adalah analisis aspek-aspek yang diambil dari hasil pengujian. Aspek yang diambil dari K6 adalah *Time Behavior, Capacity,* dan *Realibility*. Sementara aspek yang diambil dari Grafana adalah *Resource Utilization*. Tahapan-tahapan ini dirancang untuk memastikan bahwa pengujian performa dilakukan secara menyeluruh dan hasilnya dapat digunakan untuk mengevaluasi serta meningkatkan performa sistem yang diuji.

3.1.4 Studi Deskriptif 2

Untuk menilai keberhasilan pada penelitian yang telah dilakukan, serta untuk mengevaluasi dan menyimpulkan hasil secara tepat. Alat yang akan digunakan untuk menguji aplikasi dan mengamati pengujian yang telah di implementasi adalah K6 dan Grafana. Alat ini dapat membantu dalam menilai performa aplikasi dengan serangkaian pengujian untuk mengevaluasi berbagai aspek kinerja aplikasi perpesanan yang telah diimplementasikan. Hasil pengujian

yang diperoleh akan memberikan gambaran mengenai kinerja kedua aplikasi untuk di evaluasi dan menarik kesimpulan penelitian.

3.2 Alat dan Bahan Penelitian

3.2.1 Alat Penelitian

Dalam penelitian ini, alat yang digunakan untuk penelitian adalah sebagai berikut:

1. Perangkat Keras

   Laptop Acer Nitro V 15 dengan spesifikasi :

   a. Prosesor Intel I5 - 13420H 8 Core
   b. OS Windows 11 versi 23H2
   c. RAM 8GB DDR5
   d. VGA Nvidia GeForce RTX 2050 4GB VRAM
   e. 512GB SSD NVMe

   Virtual machine dengan spesifikasi :

   a. 3 Core CPUs
   b. OS Ubuntu Server versi 24.04
   c. RAM 1556MB
   d. 16MB VRAM
   e. 20GB Storage

2. Perangkat Lunak
   a. Visual Studio Code
   b. K6
   c. Node JS
   d. Node Package Manager (NPM)
   e. Virtual Box
   f. Grafana
   g. Prometheus
   h. Node Exporter
   i. Docker

3.2.2 Bahan Penelitian

Bahan penelitian yang akan digunakan untuk mendukung penelitian ini adalah Artikel ilmiah, jurnal, buku, serta referensi teoritis lainnya yang berkaitan dengan topik penelitian ini digunakan sebagai landasan teoretis.

3.3 Instrumen Penelitian

Dalam instrumen penelitian, K6, UWebSocket.js serta WS *Library* menjadi fokus utama. K6 digunakan sebagai alat utama untuk pengujian kinerja, sementara UWebSocket dan WS *Library* digunakan sebagai perbandingan *library*. K6 adalah sebuah alat open-source yang digunakan untuk melakukan uji beban (*load test*ing) dan pengujian kinerja aplikasi (Pai dkk., 2021) , sedangkan UWebSocket dan WS *Library* memberikan kemampuan untuk mengimplementasikan komunikasi *real-time* (Dykes dkk., 2018). Grafana is a feature-rich, interactive visualization and dashboard software (Betke dan Kunkel, 2017).

3.4 Analisis data

Hasil eksperimen perbandingan dari dua *library*, yaitu UWebSocket dan WS, Perbandingan akan dilakukan dengan menggunakan sejumlah matriks evaluasi berdasarkan standar ISO/IEC 25023:2016 mengenai pengukuran kualitas sistem dan produk perangkat lunak bagian efisiensi performa yang meliputi *Time Behavior* untuk mengukur waktu respon dan latensi dari koneksi WebSocket dalam kondisi beban yang berbeda (ISO, 2016). *Resource Utilization* untuk mengukur penggunaan sumber daya CPU dan memori saat menjalankan UWebSocket dan WS *Library*, *Capacity* yang menilai jumlah koneksi yang dapat ditangani oleh masing-masing *library*, serta *Reliability* untuk mengevaluasi tingkat kegagalan dalam proses pengetesan. Berikut merupakan metrik yang masuk kedalam tiap aspek.

1. *Time Behavior*
    a. Response Time

       Waktu yang diperlukan untuk menerima respons setelah permintaan dikirim.
    b. Turnaround Time

       Total waktu yang dibutuhkan dari permintaan hingga penyelesaian, termasuk waktu pemrosesan dan waktu tunggu.

2. *Resource Utilization*

    a. Processor Usage

        Persentase rata rata penggunaan CPU selama operasi berjalan.

3. *Capacity*

    a. Throughput

        Mengukur jumlah data yang dapat diproses atau jumlah transaksi yang dapat ditangani. Throughput tidak disebutkan dalam standar ISO/IEC 25023:2016, namun Throughput adalah parameter yang penting. Hal ini menggambarkan kapasitas server dalam mengatasi beban yang besar. Semakin tinggi Throughput, semakin optimal performa server (Ghodasara dkk., 2018).

4. Reliability

    a. Status is 101

        Tingkat keberhasilan perubahan protokol dari http ke WebSocket, diukur sebagai persentase dari total sesi yang berhasil mengubah protokol ke WebSocket.

Berikut adalah tabel yang menjelaskan hubungan antar aspek, metrik, dan parameter yang akan digunakan:

Tabel 3.1

Metrik Penelitian

Metrik	Parameter	Deskripsi
Response Time	ws_connecting	Waktu yang diperlukan untuk membangun koneksi awal dengan server WebSocket.
	ws_session_duration	Durasi total dari sesi WebSocket, mulai dari saat koneksi dibuka hingga ditutup. Metrik ini mencakup waktu untuk mengirim dan menerima pesan selama sesi tersebut.

Turnaround Time	ws_message_duration	Waktu yang diperlukan untuk mengirim dan menerima pesan WebSocket
Processor Usage	average_cpu	Rata-rata persentase pemakaian CPU selama periode pengujian.
Throughput	ws_msgs_sent	Jumlah total pesan yang dikirim oleh klien ke server WebSocket
	ws_msgs_received	Jumlah total pesan yang diterima oleh klien dari server WebSocket
	ws_sessions	Jumlah total sesi WebSocket yang dibuka dan berjalan selama pengujian
Error Rate	status is 101	Tingkat keberhasilan perubahan protokol dari http ke WebSocket yang terjadi selama pengujian WebSocket

3.5 Hipotesis

Untuk mengidentifikasi perbedaan dalam penelitian ini, penggunaan uji statistik t-test akan digunakan untuk menganalisis perbedaan rata-rata antara kelompok atau sampel. T-test adalah metode statistik parametrik yang memerlukan data yang terdistribusi secara normal (Tae Kyun Kim, 2015). Untuk disebut data parametrik, sampel harus memenuhi beberapa kriteria tertentu (Usmadi, 2020), seperti

a. Sampel data tidak saling mempengaruhi atau harus independen satu sama lain.
b. Populasi data yang dikumpulkan harus terdistribusi secara normal, sehingga memerlukan uji normalitas.
c. Varians populasi antar kelompok harus sama (homogenitas varians).

Jika dataset memenuhi kriteria yang diperlukan, maka analisis akan dilakukan menggunakan Independent t-test. Namun, jika kriteria tersebut tidak terpenuhi, analisis akan dilakukan secara non-parametrik dengan menggunakan Mann-Whitney U-Test (atau uji Wilcoxon rank-sum).

Independent t-test adalah metode statistik yang digunakan untuk menentukan apakah terdapat perbedaan yang signifikan secara statistik antara rata-rata dari dua kelompok yang tidak saling berhubungan (independen). Jika tidak ada perbedaan antara rata-rata kedua sampel, maka selisihnya akan mendekati nol (Tae Kyun Kim, 2015). Berikut merupakan rumus yang digunakan

$$t_{hitung} = \frac{\bar{X}_1 - \bar{X}_2}{S_p\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$

$$S_p = \sqrt{\frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}}$$

$$t_{table} = \alpha; df = n_1 + n_2 - 2$$

Dimana:

- $\bar{X}_1$ dan $\bar{X}_2$ = Rata-rata dari dua sampel
- n_1 dan n_2 = Ukuran dari dua sampel
- S_p = Standar deviasi gabungan
- s_1 dan s_2 = Standar deviasi dari dua sampel
- a = nilai signifikansi
- df = derajat kebebasan (*degree of freedom*)

Keputusan:

- $t_{hitung} > t_{table}$ maka H_0 ditolak

Sementara Mann-Whitney U Test, yang juga dikenal sebagai uji Wilcoxon rank-sum, adalah alternatif non-parametrik yang sering digunakan untuk menggantikan t-test. Uji Mann-Whitney U memanfaatkan data yang diurutkan

dalam peringkat sebagai pengganti nilai asli, sehingga lebih tahan terhadap outlier dan distribusi yang tidak normal (Perme & Manevski, 2019).

$$U_1 = n_1 n_2 + \frac{n_1(n_1 + 1)}{2} - R_1$$

$$U_2 = n_1 n_2 + \frac{n_2(n_2 + 1)}{2} - R_2$$

Dimana :

- n_1 dan n_2 = Jumlah sampel 1 dan 2
- R_1 dan R_2 = Jumlah rangking pada sampel n_1 dan n_2
- U_1 dan U_2 = Jumlah peringkat 1 dan 2

Ukuran sampel yang besar dalam pengujian statistik menawarkan berbagai keuntungan, seperti estimasi yang lebih akurat dan peningkatan keakuratan statistik. (Asiamah, 2017). Penelitian ini akan menggunakan skenario beban terbesar untuk sampel yang akan di uji yaitu 3000 pengguna virtual.