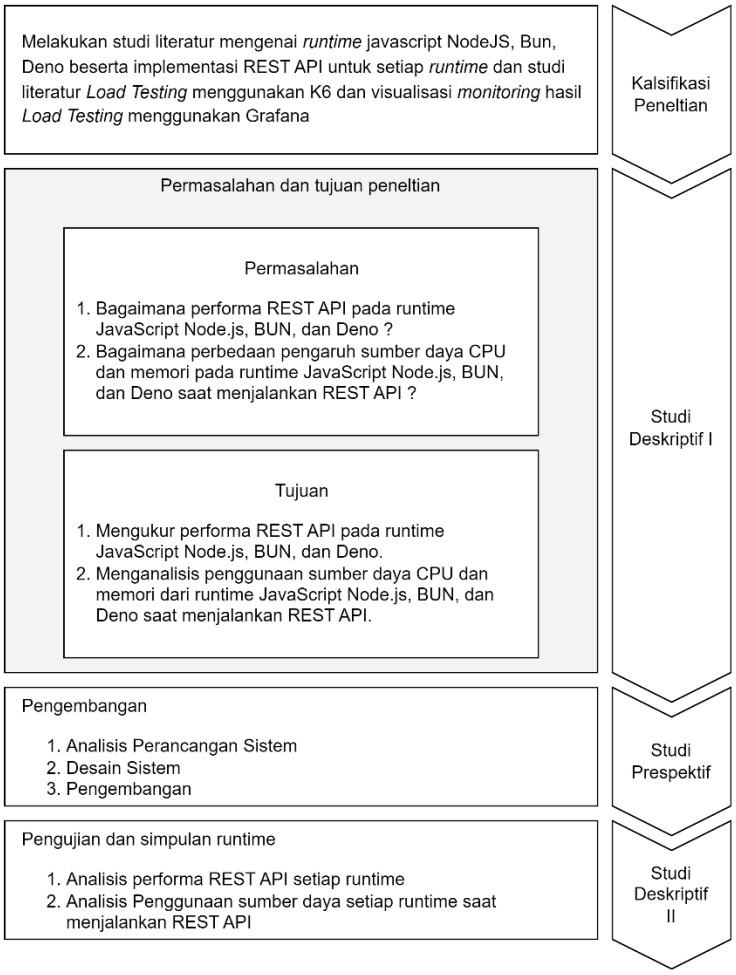
BAB III
METODE PENELITIAN

3.1 Metode Penelitian

Desain penelitian adalah suatu kerangka kerja yang berfungsi sebagai panduan dalam melaksanakan penelitian. *Design Research Methodology* (DRM) dipilih dalam penelitian ini karena DRM cocok untuk penelitian yang berfokus pada pemecahan masalah dan pengembangan perangkat lunak (Mira Orisa dkk., 2023). Skema penelitian dirancang oleh penulis sesuai dengan metode DRM, sebagaimana digambarkan pada Gambar 3.1.



Gambar 3. 1 Metode Penelitian DRM

3.1.1 Klasifikasi Penelitian

Tahap awal dalam penelitian ini adalah Melakukan studi literatur mengenai *runtime* JavaScript Node.js, Bun, Deno beserta implementasi REST API untuk setiap *runtime* dan studi literatur *Load Testing* menggunakan K6. Hasil dan saran dari Penelitian terdahulu, dapat dijadikan untuk bahan rujukan dalam pembuatan model klasifikasi teks pada penelitian ini.

3.1.2 Studi Deskriptif I

Tahap selanjutnya melibatkan analisis dan penguraian masalah serta penetapan tujuan penelitian berdasarkan temuan dari studi literatur yang telah dilakukan pada tahap sebelumnya. Analisis literatur ini menjadi landasan untuk merinci aspek-aspek penting yang perlu diatasi dan dicapai dalam penelitian ini. Informasi yang diperoleh akan menjadi dasar yang kuat dalam mengembangkan solusi untuk permasalahan penelitian yang telah diidentifikasi.

3.1.3 Studi Prespektif

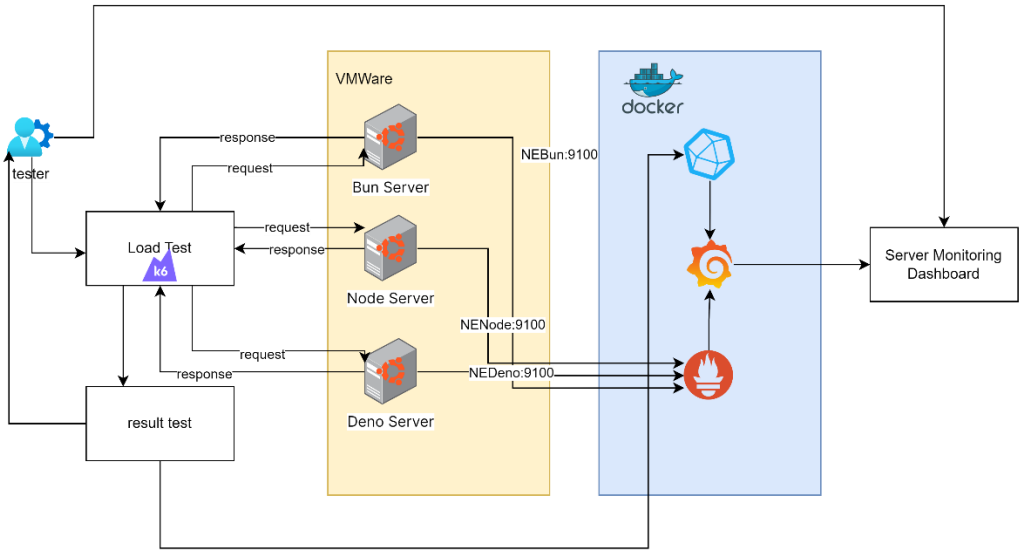
Pada tahap Studi Preskriptif ini, penelitian berfokus terhadap pengembangan dan pembangunan server. Pengembangan REST API di rancang identik sama untuk setiap *runtime* dengan spesifikasi API yang sama. Spesifikasi API dirincikan di Tabel 3.1.

Tabel 3. 1
Spesifikasi API

Endpoint	Method	Response (JSON)
{URL}/	GET	{“message” : “Hello World”}

Di tabel 3.1 menguji satu endpoint dengan *return* “Hello World” agar berfokus pada performa *runtime* terkait tanpa variable lain seperti *database query* atau kompleks logika bisnis yang mempengaruhi hasil (Kniazev & Fitiskin, 2023). REST API yang dibangun akan di-*deploy* ke server masing server *runtime* dengan spesifikasi identik sama dengan menggunakan VM. Penggunaan VM untuk menciptakan lingkungan uji yang terkontrol yang sepenuhnya terisolasi dari sistem utama. Hal ini mencegah interferensi dari variabel *external* yang mempengaruhi

hasil uji (Marzullo dkk., 2022) maka di buat 3 server identik untuk masing masing *runtime*. Arsitektur server digambarkan seperti gambar 3.2.



Gambar 3. 2 Arsitektur lingkungan pengujian server

Pada gambar 3.2 merupakan diagram arsitektur server pada penelitian ini. Setiap *virtual machine* berada di VMWare dengan masih masing VM memiliki spesifikasi yang identik. Setiap *matrix* perubahaan resource usage kirimkan oleh Node Exporter (NE) yang dikumpulkan oleh Prometheus. Grafana akan membuat dasbor untuk menampilkan *matrix* yang sudah dikumpulkan menjadi satu di dalam Prometheus menjadi grafik *realtime time-series*. Untuk Melakukan *load test* Penguji menggunakan K6 sebagai alat dengan *hit* ke IP server setiap *runtime* dengan *port* 8001. Ketika melakukan uji beban K6 akan mengeluarkan hasilnya dan divisualisasikan oleh Grafana.

Ketika membandingkan performa tiga *runtime* pada REST API menggunakan JavaScript, sangat penting untuk melakukan standarisasi spesifikasi server untuk memastikan bahwa perbandingannya valid dan dapat diandalkan. Penelitian lain mengotomatiskan pengujian REST API dan menyoroti perlunya mengendalikan variabel seperti spesifikasi server untuk mendapatkan metrik kinerja yang akurat (Baniaș dkk., 2021). Studi ini menggarisbawahi pentingnya mempertahankan pengaturan server yang konsisten untuk memastikan bahwa setiap perbedaan

kinerja yang diamati disebabkan oleh *runtime* itu sendiri, bukan oleh perangkat keras atau konfigurasi server yang mendasarinya.

3.1.4 Studi Deskriptif II

Tahap terakhir guna mendapatkan tujuan dari penelitian yang dilakukan, diperlukan proses identifikasi terhadap aplikasi uji yang telah dikembangkan. Proses identifikasi berupa pengujian performa dengan metrik *response time*, *throughput*, penggunaan memori, penggunaan CPU. Pengujian akan dilakukan menggunakan aplikasi K6. Pengujian dilakukan dengan kondisi awal 1 pengguna meningkat setiap 1 menit hingga pengguna mencapai 1000 dalam satu skenario uji (Prayogi dkk., 2020). Hasil pengujian akan dianalisa untuk mengetahui perbedaan performa antara ketiga *runtime*. Selanjutnya, dari hasil penelitian akan dibuat kesimpulan serta diuraikan kelebihan dan kekurangannya. Bagian kesimpulan akan mencakup jawaban dari rumusan masalah yang telah dirumuskan. Selain itu, peneliti juga akan memberikan saran kepada peneliti berikutnya agar penelitian dapat diperbaiki dan ditingkatkan.

3.2 Alat dan Bahan Penelitian

Di bawah ini tercantum alat yang digunakan untuk mendukung jalannya penelitian. Peralatan penelitian akan dibagi menjadi dua kelompok, yakni perangkat keras dan perangkat lunak.

1. Perangkat Keras
    - Prosessor : I7-8850U 8 CPUs 1.80 GHz
    - Kartu grafis : NVIDIA GeForce MX 150
    - Kartu memori : 16GB RAM LPDDR3 (*dual channel*)
    - Penyimpanan data : SSD 512GB NVME
2. Server *Runtime*

Untuk melakukan standarisasi spesifikasi server untuk setiap *runtime*, pada tabel 3.2 merupakan detail spesifikasi yang akan di bangun.

Tabel 3. 2
Spesifikasi server runtime

Spesifikasi server VM runtime	
OS	ubuntu 20.04 (focal)
RAM	1.5 GB (GigaByte)
CPU	2 Core
Storage	20 GB (Giga Byte)

3. Perangkat Lunak

Dalam menunjang penelitian ini diperlukan beberapa perangkat lunak yang akan digunakan. Tabel 3.3 merupakan rincian dan kegunaan masing masing perangkat lunak.

Tabel 3. 3
Perangkat lunak yang digunakan

Perangkat Lunak	Deskripsi Kegunaan
Windows 11 Pro	Sistem operasi untuk menjalankan penelitian
Google Chrome	Peramban untuk menjalankan penelitian
Visual Studio Code (IDE)	*Integrated Development Environment* (IDE) digunakan untuk menulis, mengedit, dan men-debug kode
K6 (v0.51.0)	Alat pengujian beban *open source*
VMWare (WorkStations 17 Player)	Perangkat lunak virtualisasi untuk menjalankan mesin virtual untuk menjalankan beberapa sistem operasi pada satu komputer fisik
Node.js (20.15)	Javascript *runtime*
BUN (1.1.10)	Javascript *runtime*
Deno (1.43.6)	Javascript *runtime*
Ubuntu Server (20.04 focal)	Distribusi Linux yang digunakan untuk server

Grafana (11.1.0)	Alat visualisasi data *open source* untuk membuat dasbor interaktif yang menampilkan data dari berbagai sumber secara real-time.
Prometheous (2.45.5)	Sistem *monitoring* dan *alerting open-source* untuk mengumpulkan metrik dari berbagai sumber dalam sistem.
Node Exporter (1.8.0)	Alat untuk mengumpulkan metrik dari sistem operasi berhubungan dengan Prometheus untuk menyediakan metrik penting dari *host* yang dipantau.
InfluxDB (v1.8.5)	Basis data *time-series open-source* untuk menyimpan dan menganalisis data yang berhubungan dengan waktu, seperti metrik *monitoring* dan *log*.
Docker (24.0.6)	Platform untuk mengembangkan, mengirim, dan menjalankan aplikasi dalam kontainer. Memungkinkan pengembang untuk mengemas aplikasi beserta semua dependensinya dalam satu unit yang dapat dijalankan di mana saja.
Pyhton	Python digunakan dalam penelitian ini untuk melakukan analisis statistik.

3.3 Instrumen Penelitian

Instrumen penelitian yang akan digunakan pada penelitian ini adalah alat uji berupa aplikasi K6. Aplikasi K6 dapat melakukan pengujian performa REST API. K6 dipilih karena dapat mendefinisi perilaku pengguna virtual dengan *scripting* menggunakan JavaScript. Parameter atau metrik pengujian performa pada penelitian ini berdasarkan efisiensi kinerja sesuai dengan ISO 25010. meliputi aspek perilaku waktu (*time response*), kapasitas (*throughput*) dan penggunaan sumber daya (*utilization*) CPU dan memori juga didasari dengan penelitian (Feroj, 2023).

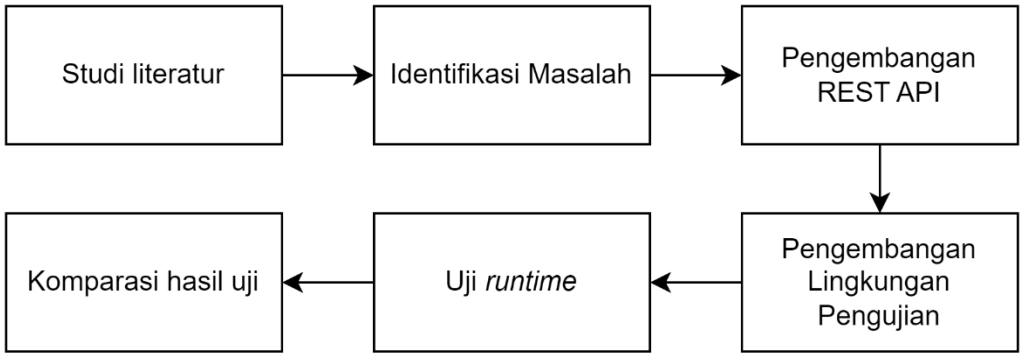
Response time merupakan waktu yang diperlukan server untuk menerima permintaan dari klien, memproses permintaan tersebut, dan mengirimkan respons kembali (Prayogi dkk., 2020). Terdapat faktor-faktor yang mempengaruhi *response time* adalah beban server, *latency* jaringan, dan kompleksitas permintaan itu sendiri. *response time* paling rendah merupakan yang paling optimal. REST API memiliki waktu respon yang optimal antara 200ms hingga 500ms, tergantung pada kompleksitas dan beban aplikasi (Ehsan dkk., 2022b).

Throughput merupakan jumlah permintaan yang dapat diolah oleh server dalam jangka waktu tertentu. *Throughput* yang tinggi menunjukkan kemampuan server untuk menangani lebih banyak permintaan dengan efisien (Nurwasito & Rahmawati, 2021). Penelitian (Prayogi dkk., 2020) menjelaskan *throughput* optimal yaitu 100% tanpa adanya *request* yang gagal.

Penggunaan sumber daya merupakan aspek penting untuk menilai efisiensi dan kapasitas server dalam menangani permintaan (Lawi dkk., 2021). Sumber daya yang di ukur adalah CPU dan *memory*. Seberapa banyak penggunaan CPU dalam persentase ketika dilakukan uji beban dan Seberapa banyak penggunaan memory dalam *Byte* ketika dilakukan uji beban. Pengukuran penggunaan *memory* memungkin memiliki perbedaan yang tidak signifikan karena tidak terlalu kompleksnya REST API atau tidak adanya operasi yang menyimpan terhadap *memory* seperti *session server-side* (Ardiansyah & Fatwanto, 2022).

3.4 Prosedur Penelitian

Prosedur penelitian akan dilakukan berdasarkan desain penelitian yang telah dijelaskan sebelumnya. Adapun secara garis besar penelitian ini dapat di gambarkan pada gambar 3.3 berikut.



Gambar 3. 3 Prosedur Penelitian

3.5 Analisis Data

Penelitian ini dirancang untuk membandingkan performa REST API pada *runtime* JavaScript Node.js, Bun, dan Deno dengan mengukur parameter pengujian performa seperti *response time*, *throughput*, dan *utilization* (CPU dan memori). Nilai-nilai parameter *response time*, *throughput* akan diambil dari aplikasi uji dan parameter *utilization* penggunaan sumber data CPU dan memory di ambil dari Grafana dengan memanfaatkan REST API *endpoint*. Pengujian akan dilakukan di lingkungan lokal untuk menghindari masalah koneksi dan sebagainya. Penelitian akan dengan lingkungan yang terkontrol dan spesifikasi yang di tentukan. Metode pengujian yang digunakan adalah *load testing* dengan memberikan beban secara bertahap pada aplikasi uji. Nilai parameter yang diperoleh dilakukan *Games-Howel Test* (Penkov dkk., 2020) untuk menentukan kesimpulan dari setiap *runtime* yang paling performa.

3.6 Hipotesis

Analisis of variance (ANOVA) merupakan salah satu teknik analisis multivariate yang berfungsi untuk membedakan rerata lebih dari dua kelompok data dengan cara membandingkan variansinya (Riadi dkk., 2020). Tujuannya untuk mengukur signifikansi perbedaan antara rata-rata kelompok metrik performa yang diuji. Uji anova yang digunakan yaitu uji anova jenis satu arah karena hanya terdapat satu faktor yang memperngaruhi variabel, yaitu *runtime*. Untuk menjalankan uji anova, terdapat syarat salah satunya yaitu variansi dari setiap kelompok harus sama. Maka dari itu perlu dilakukan uji homogenitas terlebih

dahulu untuk mengetahui apakah memenuhi asumsi uji anova. Jika variansi anova tidak teruji homogenitas *p-value* lebih kecil dari 0,05 maka menggunakan Welch's anova (Jan & Shieh, 2020).

Dalam uji ANOVA ini, terdapat beberapa elemen penting, yaitu faktor, variabel respon, dan kelompok. Faktor dalam pengujian ini adalah *runtime* yang digunakan, yaitu Node.js, Bun, dan Deno. Ketiga *runtime* ini berfungsi sebagai variabel independen yang mempengaruhi performa eksekusi kode JavaScript. Variabel respon adalah metrik yang diukur untuk menilai performa dari setiap runtime (*response time, throughput, memory usage, dan CPU load*). Variabel respon ini sebagai variabel dependen yang dipengaruhi oleh *runtime*. Sedangkan kelompok merupakan hasil dari pengujian metrik dari setiap *runtime*. Misalnya, untuk metrik response time, terdapat tiga kelompok data yang masing-masing berasal dari pengukuran performa ketiga *runtime*. Demikian pula, untuk metrik *throughput, memory usage*, dan *CPU load*, masing-masing memiliki tiga kelompok data yang terpisah untuk setiap *runtime*. Hipotesis pada penelitian ini adalah bahwa ada perbedaan signifikan dalam performa di antara ketiga *runtime* tersebut.

- H_0 : Tidak ada perbedaan signifikan dalam rata-rata performa antara *runtime* Node.js, Bun, dan Deno.
- H_1 : Ada perbedaan signifikan dalam rata-rata performa antara *runtime* Node.js, Bun, dan Deno.