APLIKASI SISTEM MONITORING RESOURCE SERVER BERBASIS CLI DENGAN NOTIFIKASI REAL-TIME

Disusun untuk memenuhi tugas kelompok mata kuliah: Metode Penelitian

Dosen Pengampu: Dr. Sugiyanto M.Kom

Disusun oleh kelompok 1:

Hafizh Muhammad Hikmatiar (230511096)

Zaki Al Fattah (230511103)

Bondan Dwi Prasetya (230511079)

Kamal Erlambang (230511075)

TI23C

KABUPATEN CIREBON

UNIVERSITAS MUHAMMADIYAH CIREBON KAMPUS 2

TAHUN 2026

KATA PENGANTAR

Puji syukur kami panjatkan ke hadirat Allah SWT yang telah melimpahkan rahmat dan hidayah-Nya sehingga kami dapat menyelesaikan proposal penelitian yang berjudul "Sistem Monitoring Resource Server Berbasis Command Line Interface (CLI) Menggunakan Python" ini dengan baik dan tepat waktu.

Proposal penelitian ini disusun sebagai salah satu syarat untuk melaksanakan penelitian dalam rangka penyelesaian studi pada Program Studi Teknik Informatika. Penelitian ini bertujuan untuk merancang dan membangun sistem pemantauan sumber daya server yang ringan, efisien, dan mampu memberikan notifikasi peringatan secara real-time kepada administrator sistem ketika terjadi kondisi anomali.

Dalam proses penyusunan proposal ini, kami banyak mendapatkan bimbingan, arahan, dukungan, dan bantuan dari berbagai pihak. Oleh karena itu, pada kesempatan ini kami menyampaikan ucapan terima kasih yang sebesar-besarnya kepada:

1. Orang tua dan keluarga tercinta yang senantiasa memberikan doa, motivasi, dan dukungan moral maupun material selama kami menempuh pendidikan.
2. Bapak/Ibu Dosen Pembimbing yang telah meluangkan waktu untuk memberikan bimbingan, saran, dan arahan yang sangat berharga dalam penyusunan proposal penelitian ini.
3. Bapak/Ibu Dosen Program Studi Teknik Informatika yang telah memberikan ilmu pengetahuan dan pengalaman yang sangat berharga selama masa perkuliahan.
4. Rekan-rekan mahasiswa Teknik Informatika yang telah memberikan semangat, masukan, dan dukungan selama proses penyusunan proposal ini.
5. Semua pihak yang tidak dapat kami sebutkan satu per satu, yang telah turut membantu dalam penyelesaian proposal penelitian ini.

Kami menyadari sepenuhnya bahwa dalam penyusunan proposal penelitian ini masih terdapat banyak kekurangan dan keterbatasan, baik dari segi isi, sistematika penulisan, maupun teknis penyajian. Oleh karena itu, kami sangat mengharapkan kritik dan saran yang bersifat membangun dari semua pihak demi perbaikan dan penyempurnaan penelitian ini ke depannya.

Besar harapan kami agar proposal penelitian ini dapat diterima dan bermanfaat, tidak hanya bagi kami sendiri, tetapi juga bagi para pembaca, civitas akademika, dan para administrator sistem yang membutuhkan solusi pemantauan server yang efisien dan praktis.

Cirebon 15 Februari 2026

Penulis

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam infrastruktur teknologi informasi modern, server merupakan komponen vital yang menopang stabilitas layanan digital. Pengelolaan sumber daya utama seperti Central Processing Unit (CPU), RAM, dan Disk Storage sangat menentukan performa sistem secara keseluruhan. Namun, seiring dengan kompleksitas infrastruktur, kebutuhan akan sistem pemantauan (monitoring) yang mampu bekerja secara efisien tanpa membebani performa perangkat menjadi tantangan tersendiri bagi administrator sistem.Penelitian sebelumnya telah mengeksplorasi penggunaan alat monitoring skala besar seperti Zabbix yang diterapkan pada lembaga pemerintahan seperti Badan Pengawasan Keuangan dan Pembangunan (BPKP). Meskipun Zabbix sangat komprehensif dalam pengawasan jaringan dan server, implementasinya seringkali memerlukan konfigurasi infrastruktur yang kompleks dan sumber daya yang tidak sedikit untuk manajemen basis data serta dasbor GUI-nya (Arisanti & Prasetijo, 2022). Di sisi lain, tren teknologi saat ini mulai bergeser pada efisiensi penggunaan sumber daya. Penelitian mengenai sistem deteksi bahaya berbasis IoT menunjukkan pentingnya metode monitoring yang adaptif untuk menekan konsumsi energi dan biaya operasional namun tetap menjaga keakuratan data (Munir dkk, 2022).

Penelitian terdahulu telah mengimplementasikan alat pemantauan skala besar seperti Zabbix pada instansi pemerintahan seperti Badan Pengawasan Keuangan dan Pembangunan (BPKP). Meskipun Zabbix menawarkan fitur pengawasan jaringan dan server yang komprehensif, implementasinya memerlukan konfigurasi infrastruktur yang kompleks dan konsumsi sumber daya yang signifikan untuk pengelolaan basis data serta antarmuka grafis (GUI) (Arisanti & Prasetijo, 2022). Hal ini menjadi hambatan bagi organisasi yang memerlukan pemantauan ringan pada perangkat dengan spesifikasi terbatas.

Efisiensi penggunaan sumber daya merupakan aspek krusial dalam pemantauan modern. Sebagaimana dijelaskan dalam penelitian mengenai sistem deteksi bahaya berbasis IoT, efisiensi energi, biaya, dan sumber daya melalui metode pemantauan adaptif sangat diperlukan untuk menjaga keberlanjutan sistem (Munir dkk, 2022). Pendekatan ini relevan dalam pemantauan server, di mana sistem harus mampu memberikan informasi akurat tanpa mengganggu beban kerja utama server tersebut. Selain itu, kecepatan respon merupakan faktor kunci dalam mitigasi gangguan. Penggunaan model on-device Artificial Intelligence untuk pemantauan real-time membuktikan bahwa pemrosesan data di tingkat lokal memberikan keunggulan berupa respon yang instan tanpa ketergantungan penuh pada transfer data eksternal yang besar (Smith dkk, 2022).

Berdasarkan tinjauan tersebut, penelitian ini mengusulkan pengembangan Sistem Monitoring Resource Server Berbasis CLI. Berbeda dengan sistem berbasis GUI yang berat, sistem ini dirancang menggunakan bahasa pemrograman Python dengan antarmuka Command Line Interface (CLI) untuk memastikan penggunaan sumber daya yang minimal. Sistem ini juga mengintegrasikan fitur notifikasi real-time yang memungkinkan administrator menerima peringatan kritis secara instan melalui perangkat seluler saat terjadi anomali beban sumber daya, sehingga tindakan preventif dapat segera dilakukan untuk menjaga reliabilitas layanan.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, rumusan masalah dalam penelitian ini Adalah:

1. Bagaimana merancang sistem monitoring *resource* server yang ringan dan efisien berbasis *Command Line Interface* (CLI) berbasis Python?
2. Bagaimana membangun mekanisme notifikasi *real-time* yang mampu memberikan peringkatan instan kepada administrator saat penggunaan sumber daya server melampaui ambang batas?

1.3 Batasan Masalah

Agar penelitian tetap terarah sesuai dengan rincian isi proposal, batasan masalah ditentukan sebagai berikut:

1. Parameter monitoring terbatas pada penggunaan CPU, RAM, dan kapasitas penyimpanan (Disk).
2. Pengembangan sistem menggunakan bahasa pemrograman Python dan pustaka akses sistem psutil.
3. Output utama sistem berupa antarmuka berbasis teks (CLI) dan pesan notifikasi real-time.
4. Sistem diimplementasikan dan diuji pada lingkungan sistem operasi berbasis Linux.
5. Parameter ambang batas (threshold) peringatan ditentukan secara statis di dalam konfigurasi sistem.

1.4 Tujuan Penelitian

Tujuan yang ingin dicapai melalui penelitian ini adalah:

1. Menghasilkan aplikasi monitoring berbasis CLI yang mampu menyajikan data penggunaan sumber daya server secara akurat dengan konsumsi memori rendah.
2. Mengembangkan fitur notifikasi otomatis untuk mempercepat waktu respon administrator dalam menangani anomali pada server.
3. Memberikan alternatif solusi pemantauan yang efisien dan adaptif untuk manajemen infrastruktur server.

1.5 Manfaat Penelitian

1. Bagi Administrator Sistem: Mempermudah pengawasan kinerja server secara praktis tanpa beban konfigurasi yang rumit dan memungkinkan pemantauan jarak jauh melalui notifikasi.
2. Bagi Instansi/Organisasi: Menekan risiko terjadinya kegagalan sistem (downtime) dengan biaya operasional sumber daya yang lebih hemat.
3. Bagi Peneliti: Menambah wawasan mengenai optimasi sumber daya dalam pengembangan skrip administrasi sistem dan integrasi sistem peringatan dini.

BAB II
TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 Penelitian Terkait

Beberapa penelitian sebelumnya telah dilakukan terkait sistem monitoring server dan infrastruktur teknologi informasi. Tabel berikut merangkum penelitian-penelitian yang relevan dan menunjukkan posisi penelitian ini di antara penelitian terdahulu.

Tabel 2.1 Penelitian Terkait

No	Peneliti & Tahun	Judul Penelitian	Teknologi/Metode	Keterangan
1	Arisanti & Prasetijo (2022)	Implementasi Sistem Monitoring Jaringan Berbasis Zabbix di BPKP	Zabbix, GUI, Web	Monitoring komprehensif namun membutuhkan infrastruktur dan konfigurasi yang besar
2	Munir dkk (2022)	Sistem Deteksi Bahaya Berbasis IoT dengan Pemantauan Adaptif	IoT, Python, Sensor	Fokus pada efisiensi energi sensor IoT, tidak spesifik pada server monitoring
3	Smith dkk (2022)	On-Device AI untuk Real-Time Server Monitoring	AI, Python, Edge Computing	Pemrosesan lokal dan respons instan, namun kompleksitas implementasi AI tinggi
4	Pratama & Wijaya (2022)	Pengembangan Dashboard Monitoring Server Berbasis Web	PHP, Laravel, Web GUI	Antarmuka web modern tetapi konsumsi sumber daya lebih besar dibanding CLI
5	Haryadi dkk (2023)	Analisis Performa Tools Monitoring: Nagios vs Prometheus	Nagios, Prometheus, Grafana	Perbandingan tools enterprise dengan overhead konfigurasi yang signifikan
6	Nugroho & Santoso (2022)	Rancang Bangun Sistem Monitoring Server Menggunakan Protokol SNMP Berbasis Web	SNMP, PHP, MySQL, Web	Monitoring via protokol SNMP, namun bergantung pada layanan web server yang aktif
7	Rahmawati dkk (2022)	Implementasi Notifikasi Real-Time pada Sistem Monitoring Jaringan Menggunakan Telegram Bot	Python, Telegram Bot API, SNMP	Integrasi notifikasi Telegram sudah baik, tetapi fokus pada monitoring jaringan bukan resource server

No	Peneliti & Tahun	Judul Penelitian	Teknologi/Metode	Keterangan
8	Kusuma & Hidayat (2022)	Perancangan Sistem Monitoring Infrastruktur IT Berbasis Open Source menggunakan Prometheus dan Grafana	Prometheus, Grafana, Docker	Visualisasi data sangat kaya namun membutuhkan deployment Docker dan resources yang tidak sedikit
9	Putra & Maharani (2022)	Lightweight Server Health Monitoring Menggunakan Shell Script pada Lingkungan Linux	Bash, Shell Script, Cron	Pendekatan CLI ringan menggunakan Shell Script, namun portabilitas dan pengelolaan kode lebih terbatas dibanding Python
10	Firmansyah dkk (2023)	Pengembangan Skrip Python untuk Otomatisasi Monitoring dan Pelaporan Kinerja Server	Python, psutil, SMTP, CSV	Monitoring otomatis dengan pelaporan via email, belum mengimplementasikan notifikasi instan real-time

Berdasarkan tabel perbandingan di atas, dapat dilihat bahwa penelitian terdahulu umumnya berfokus pada tools monitoring berbasis GUI seperti Zabbix, Nagios, dan Prometheus yang memiliki fitur lengkap namun membutuhkan sumber daya sistem yang besar. Penelitian ini hadir sebagai alternatif solusi monitoring yang ringan, efisien, dan tetap mampu memberikan notifikasi real-time kepada administrator sistem.

2.2 Landasan Teori

2.2.1 Monitoring Resource Server

Monitoring resource server adalah proses pemantauan berkelanjutan terhadap penggunaan sumber daya komputasi pada sebuah server, meliputi Central Processing Unit (CPU), Random Access Memory (RAM), dan kapasitas penyimpanan disk. Tujuan utama monitoring adalah mendeteksi anomali sedini mungkin sebelum menyebabkan gangguan layanan (downtime) yang dapat merugikan organisasi.

Menurut Smith dkk (2022), pemantauan sumber daya yang efektif harus memenuhi tiga kriteria utama, yaitu akurasi data, kecepatan respons, dan efisiensi penggunaan sumber daya oleh agen monitoring itu sendiri. Sebuah sistem monitoring yang mengonsumsi terlalu banyak CPU atau RAM justru akan mengganggu performa server yang dipantau.

2.2.2 Command Line Interface (CLI)

Command Line Interface (CLI) adalah jenis antarmuka pengguna yang berbasis teks, di mana pengguna berinteraksi dengan sistem komputer melalui perintah-perintah yang diketikkan pada terminal atau konsol. Berbeda dengan Graphical User Interface (GUI) yang memerlukan rendering grafis, CLI bekerja secara langsung pada level sistem operasi sehingga konsumsi memori dan CPU-nya jauh lebih rendah.

Dalam konteks administrasi sistem, CLI menjadi pilihan utama para administrator server karena kemampuannya untuk bekerja pada lingkungan dengan spesifikasi minimal, kemudahan otomatisasi melalui scripting, serta keamanan yang lebih terjaga karena tidak memerlukan port layanan web yang terbuka.

2.2.3 Python sebagai Bahasa Pemrograman

Python adalah bahasa pemrograman interpretatif tingkat tinggi yang dirancang dengan filosofi keterbacaan kode (code readability) sebagai prioritas utama. Python saat ini menjadi salah satu bahasa pemrograman paling populer di dunia, terutama di bidang administrasi sistem, data science, dan pengembangan otomatisasi.

Keunggulan Python untuk pengembangan alat administrasi sistem antara lain: sintaks yang ringkas dan mudah dipahami, ekosistem pustaka (library) yang sangat luas, dukungan multi-platform (Windows, Linux, macOS), serta kemampuan interaksi langsung dengan sistem operasi melalui modul bawaan seperti os, subprocess, dan platform.

2.2.4 Pustaka psutil

psutil (python system and process utilities) adalah pustaka Python lintas platform yang menyediakan antarmuka untuk mengambil informasi mengenai proses yang sedang berjalan dan penggunaan sumber daya sistem. Pustaka ini berfungsi sebagai lapisan abstraksi yang memungkinkan pengambilan data sistem operasi tanpa harus menangani perbedaan implementasi antar platform secara manual.

Fungsi-fungsi utama psutil yang dimanfaatkan dalam penelitian ini meliputi psutil.cpu_percent() untuk mendapatkan persentase penggunaan CPU, psutil.virtual_memory() untuk data penggunaan RAM termasuk total, used, dan available, serta psutil.disk_usage() untuk informasi kapasitas dan penggunaan disk pada path yang ditentukan. Semua fungsi tersebut mengembalikan data secara real-time dengan overhead minimal.

2.2.5 Mekanisme Notifikasi Real-Time

Notifikasi real-time dalam konteks sistem monitoring adalah kemampuan sistem untuk mengirimkan peringatan secara otomatis dan instan kepada administrator ketika suatu kondisi abnormal terdeteksi, tanpa perlu menunggu siklus pelaporan periodik. Ini berbeda dengan sistem polling biasa yang hanya melaporkan kondisi pada interval waktu tertentu.

Implementasi notifikasi real-time umumnya memanfaatkan protokol komunikasi asinkron seperti webhook atau API messaging. Salah satu platform yang banyak digunakan untuk notifikasi sistem monitoring adalah Telegram, yang menyediakan Bot API dengan kemudahan integrasi dan kecepatan pengiriman pesan yang tinggi, bahkan dapat diakses tanpa biaya berlangganan.

2.2.6 Threshold dan Mekanisme Alert

Threshold atau ambang batas adalah nilai batas parameter sistem yang jika terlampaui akan memicu suatu aksi, dalam hal ini pengiriman notifikasi. Penentuan nilai threshold yang tepat sangat krusial dalam sistem monitoring; nilai yang terlalu rendah akan menghasilkan terlalu banyak false alarm, sedangkan nilai yang terlalu tinggi berisiko melewatkan kondisi kritis.

Praktik umum dalam administrasi sistem menetapkan threshold penggunaan CPU pada 80-90%, penggunaan RAM pada 85-90%, dan penggunaan disk pada 80-85% sebagai titik peringatan (warning), dengan level kritis di atas nilai-nilai tersebut.

BAB III
METODOLOGI PENELITIAN

3.1 Analisis Kebutuhan

3.1.1 Kebutuhan Data

Data yang dibutuhkan dalam penelitian ini meliputi data penggunaan sumber daya server secara real-time yang diperoleh langsung dari sistem operasi melalui pustaka psutil. Data tersebut terdiri dari persentase penggunaan CPU, data memori virtual (total, used, free, dan persentase), serta informasi kapasitas dan penggunaan disk storage pada direktori root sistem.

Selain data primer dari sistem operasi, penelitian juga membutuhkan data konfigurasi berupa nilai-nilai threshold peringatan yang disimpan dalam berkas konfigurasi sistem (config.py atau config.json), serta kredensial API Telegram untuk fitur notifikasi real-time.

3.1.2 Spesifikasi Perangkat Keras

Perangkat keras minimum yang dibutuhkan untuk menjalankan sistem monitoring ini adalah sebagai berikut: prosesor dengan kecepatan minimal 1 GHz, memori RAM minimal 512 MB, penyimpanan disk minimal 1 GB untuk sistem operasi dan instalasi Python, serta koneksi jaringan internet untuk fitur notifikasi real-time melalui Telegram Bot API.

Untuk lingkungan pengembangan dan pengujian, penelitian ini menggunakan server dengan spesifikasi: prosesor Intel Core i5 generasi ke-8, RAM 8 GB DDR4, penyimpanan SSD 256 GB, dan sistem operasi Ubuntu Server 22.04 LTS.

3.1.3 Spesifikasi Perangkat Lunak

Perangkat lunak yang digunakan dalam penelitian ini adalah: Python versi 3.8 atau lebih baru sebagai bahasa pemrograman utama, pustaka psutil versi 5.9 atau lebih baru untuk pengambilan data sumber daya sistem, pustaka requests untuk komunikasi dengan Telegram Bot API, sistem operasi Ubuntu Server 22.04 LTS sebagai platform pengujian utama, dan Visual Studio Code sebagai Integrated Development Environment (IDE).

3.2 Metode Pengembangan Sistem

Penelitian ini menggunakan metode pengembangan sistem Waterfall (Air Terjun) yang bersifat sekuensial dan terstruktur. Pemilihan metode Waterfall didasarkan pada pertimbangan bahwa ruang lingkup sistem telah didefinisikan dengan jelas sejak awal penelitian, sehingga model pengembangan linier cocok diterapkan. Tahapan dalam metode Waterfall yang digunakan adalah sebagai berikut.

1. Requirements Analysis: Analisis kebutuhan fungsional dan non-fungsional sistem secara menyeluruh.
2. System Design: Perancangan arsitektur sistem, alur data, dan desain modul perangkat lunak.
3. Implementation: Penulisan kode program Python berdasarkan hasil rancangan sistem.
4. Testing: Pengujian sistem menggunakan metode Black Box Testing dan pengukuran performa.
5. Deployment & Maintenance: Implementasi sistem pada lingkungan server produksi dan pemeliharaan.

3.3 Perancangan Sistem

3.3.1 Arsitektur Sistem

Arsitektur sistem monitoring resource server berbasis CLI ini terdiri dari tiga komponen utama yang

saling terintegrasi. Pertama, lapisan pengumpulan data (Data Collection Layer) yang bertugas mengambil data penggunaan sumber daya secara periodik menggunakan pustaka psutil. Kedua, lapisan pemrosesan dan tampilan (Processing & Display Layer) yang memformat data menjadi output CLI yang informatif dan mudah dibaca. Ketiga, lapisan notifikasi (Notification Layer) yang memantau nilai threshold dan mengirimkan peringatan melalui Telegram Bot API apabila terjadi anomali.

Alur data pada sistem ini dimulai dari eksekusi skrip Python oleh administrator melalui terminal. Sistem kemudian memasuki loop pemantauan berkelanjutan, di mana pada setiap iterasi data sumber daya diambil, diproses, ditampilkan pada CLI, dan diperiksa terhadap nilai threshold. Jika threshold terlampaui, sistem secara otomatis mengirimkan notifikasi ke akun Telegram yang telah dikonfigurasi.

3.3.2 Use Case Diagram

Use Case Diagram sistem monitoring resource server menggambarkan interaksi antara dua aktor utama, yaitu Administrator Sistem dan Server (sebagai sumber data), dengan sistem yang dikembangkan. Aktor Administrator Sistem dapat melakukan aksi: menjalankan sistem monitoring, menghentikan sistem monitoring, mengkonfigurasi nilai threshold, dan menerima notifikasi peringatan. Aktor Server menyediakan data sumber daya sistem yang diakses secara otomatis oleh skrip monitoring.

3.3.3 Activity Diagram

Activity Diagram menggambarkan alur kerja sistem secara keseluruhan dari awal hingga akhir eksekusi. Sistem dimulai dengan pembacaan file konfigurasi untuk mendapatkan nilai threshold dan kredensial API. Selanjutnya sistem memasuki loop utama yang dijalankan secara periodik (default setiap 5 detik). Dalam setiap siklus, sistem mengambil data CPU, RAM, dan disk, kemudian menampilkan hasilnya di terminal dengan format yang terstruktur. Sistem kemudian melakukan pengecekan threshold; jika penggunaan sumber daya melebihi batas yang ditetapkan dan cooldown notifikasi telah terlampaui, sistem mengirimkan pesan peringatan melalui Telegram.

3.3.4 Entity Relationship Diagram (ERD)

Mengingat sistem ini berbasis CLI dan tidak menggunakan database relasional untuk penyimpanan data operasional, ERD pada penelitian ini difokuskan pada struktur data konfigurasi yang disimpan dalam berkas JSON. Entitas utama dalam sistem ini adalah: entitas Config yang menyimpan atribut threshold_cpu, threshold_ram, threshold_disk, telegram_bot_token, telegram_chat_id, dan monitoring_interval; serta entitas AlertLog yang menyimpan timestamp, resource_type, nilai_saat_ini, dan threshold_value sebagai catatan riwayat peringatan yang pernah terkirim.

3.3.5 Sequence Diagram

Sequence Diagram menggambarkan urutan interaksi antar objek dalam sistem. Untuk skenario pengiriman notifikasi, urutan interaksinya adalah: Administrator mengeksekusi skrip melalui Terminal, skrip MonitoringService memanggil psutil untuk mendapatkan data sumber daya, psutil mengembalikan data ke MonitoringService, MonitoringService melakukan perbandingan dengan nilai threshold dari ConfigManager, apabila threshold terlampaui maka MonitoringService memanggil NotificationService untuk mengirim pesan, NotificationService mengirimkan HTTP POST request ke Telegram Bot API, dan Telegram Bot API merespons dengan konfirmasi pengiriman.

3.4 Teknik Pengujian

3.4.1 Black Box Testing

Pengujian Black Box dilakukan untuk memverifikasi bahwa setiap fitur sistem bekerja sesuai dengan

spesifikasi fungsional yang telah ditetapkan, tanpa mempedulikan detail implementasi kode di dalamnya. Skenario pengujian yang dirancang mencakup enam aspek utama, yaitu: pengujian akurasi pembacaan data CPU, pengujian akurasi pembacaan data RAM, pengujian akurasi pembacaan data disk, pengujian mekanisme trigger notifikasi saat threshold terlampaui, pengujian bahwa notifikasi tidak dikirim saat kondisi normal, dan pengujian format tampilan output CLI.

Setiap skenario pengujian akan didokumentasikan dalam tabel pengujian yang mencantumkan nomor pengujian, skenario, data masukan, hasil yang diharapkan (expected result), hasil aktual (actual result), dan status (Pass/Fail).

3.4.2 Pengujian Performa

Selain pengujian fungsional, penelitian ini juga melakukan pengujian performa untuk mengukur seberapa ringan sistem monitoring yang dikembangkan dibandingkan dengan tools monitoring populer lainnya. Parameter performa yang diukur meliputi: konsumsi CPU oleh proses monitoring (dalam persen), konsumsi RAM oleh proses monitoring (dalam MB), dan waktu respons sejak threshold terlampaui hingga notifikasi diterima di Telegram (dalam detik).

Pengujian performa dilakukan dengan menjalankan sistem monitoring selama 30 menit pada kondisi beban server yang bervariasi (idle, beban sedang 50%, dan beban penuh 90%). Data konsumsi sumber daya oleh proses monitoring dicatat setiap menit dan kemudian dihitung rata-ratanya sebagai hasil pengujian performa.

3.5 Jadwal Penelitian

Penelitian ini direncanakan berlangsung selama lima bulan, dari bulan Oktober 2025 hingga Februari 2026. Jadwal penelitian disusun dalam bentuk Gantt Chart sebagaimana ditampilkan pada Tabel 3.1 berikut.

Tabel 3.1 Jadwal Penelitian (Gantt Chart)

Kegiatan	Oktober	November	Desember	Januari	Februari
Studi Literatur & Pengumpulan Data	✓				
Analisis Kebutuhan Sistem	✓	✓			
Perancangan Sistem (Desain)		✓			
Implementasi / Coding		✓	✓		
Pengujian (Testing)			✓	✓	
Penulisan Laporan / Skripsi				✓	✓
Sidang / Presentasi					✓

BAB IV

IMPLEMENTASI SISTEM

4.1 Lingkungan Implementasi

Implementasi sistem monitoring ini dilakukan setelah tahapan perancangan selesai. Berdasarkan spesifikasi yang telah ditetapkan pada BAB III, lingkungan implementasi dibagi menjadi perangkat keras dan perangkat lunak.

4.1.1 Spesifikasi Perangkat Keras

Pengujian dan implementasi dilakukan pada server dengan spesifikasi sebagai berikut:

1. **Prosesor:** Intel Core i5 Generasi ke-8 (4 Core, 3.6 GHz).
2. **Memori (RAM):** 8 GB DDR4.
3. **Penyimpanan:** SSD 256 GB.
4. **Jaringan:** Koneksi Internet stabil (Minimal 1 Mbps) untuk pengiriman notifikasi Telegram.

4.1.2 Spesifikasi Perangkat Lunak

Lingkungan perangkat lunak yang digunakan untuk membangun sistem adalah:

1. **Sistem Operasi:** Ubuntu Server 22.04 LTS (Linux).
2. **Bahasa Pemrograman:** Python versi 3.10.
3. **Pustaka Utama:** `psutil` (v5.9.0) untuk monitoring sistem dan `requests` (v2.28.0) untuk API Telegram.
4. **Text Editor:** Visual Studio Code dengan ekstensi Python.
5. **Terminal:** Bash Shell.

4.2 Struktur Program

Sistem dibangun dengan struktur modul yang terpisah untuk memudahkan pemeliharaan (maintenance). Berikut adalah struktur direktori proyek:

```
/server-monitor-cli
│
├── main.py              # File utama eksekusi program
├── config.json          # File konfigurasi threshold dan API
├── requirements.txt     # Daftar dependensi pustaka
├── src/
│   ├── monitor.py       # Modul pengambilan data resource (psutil)
│   ├── notifier.py      # Modul pengiriman notifikasi (Telegram)
│   └── logger.py        # Modul pencatatan log alert
└── logs/
    └── alert_log.txt    # Penyimpanan riwayat alert lokal
```

4.3 Konfigurasi Sistem

Konfigurasi sistem disimpan dalam berkas `config.json` agar administrator dapat mengubah nilai ambang batas (threshold) dan kredensial notifikasi tanpa mengubah kode sumber. Berikut adalah contoh isi berkas konfigurasi:

```
{
  "threshold": {
    "cpu": 80,
    "ram": 85,
    "disk": 80
  },
  "telegram": {
    "bot_token": "123456789:ABCdefGHIjklMNOpqrsTUVwxyz",
    "chat_id": "987654321"
  },
  "interval": 5,
  "cooldown": 300
}
```

Parameter `cooldown` diatur selama 300 detik (5 menit) untuk mencegah *spamming* notifikasi jika kondisi anomali berlangsung terus-menerus.

4.4 Implementasi Fitur Monitoring

Inti dari sistem terletak pada modul `src/monitor.py` yang memanfaatkan pustaka `psutil`. Fungsi utama yang diimplementasikan adalah:

1. **`get_cpu_usage()`**: Mengambil persentase penggunaan CPU rata-rata per core.
2. **`get_ram_usage()`**: Mengambil informasi memori virtual (used, total, percent).
3. **`get_disk_usage()`**: Mengambil informasi penggunaan disk pada partisi root (`/`).

Data tersebut diambil dalam sebuah *looping* tak terbatas yang diatur oleh `time.sleep(interval)` sesuai konfigurasi. Setiap data yang diambil langsung dicetak ke dalam CLI dengan format tabel sederhana menggunakan library bawaan Python untuk memastikan keterbacaan.

4.5 Implementasi Notifikasi Real-Time

Modul `src/notifier.py` bertanggung jawab mengirimkan permintaan HTTP POST ke API Telegram (`https://api.telegram.org/bot<token>/sendMessage`). Notifikasi hanya akan dikirimkan jika kondisi berikut terpenuhi:

1. Nilai resource saat ini > Nilai threshold konfigurasi.
2. Waktu terakhir notifikasi dikirim + cooldown < Waktu saat ini.

Pesan notifikasi dirancang informatif, contohnya:

> **PERINGATAN SERVER** Waktu: 2026-02-10 14:30:05 Resource: CPU Usage Nilai Saat Ini: 92% Threshold: 80% Status: KRITIS

BAB V

HASIL PENGUJIAN FUNGSIONAL

5.1 Skenario Pengujian Black Box

Pengujian fungsional dilakukan menggunakan metode *Black Box Testing* sebagaimana direncanakan pada BAB III. Tujuannya adalah untuk memvalidasi apakah sistem berjalan sesuai dengan spesifikasi kebutuhan tanpa meninjau kode internal. Terdapat enam skenario pengujian utama.

5.2 Hasil Pengujian

Berikut adalah tabel hasil pengujian fungsional yang telah dilaksanakan pada lingkungan Ubuntu Server 22.04 LTS.

Tabel 5.1 Hasil Pengujian Black Box Testing

No	Skenario Pengujian	Data Masukan	Hasil yang Diharapkan	Hasil Aktual	Status
1	Pembacaan Data CPU	Menjalankan `stress --cpu 4`	CLI menampilkan % CPU naik akurat	CLI menampilkan 95-100%	Pass
2	Pembacaan Data RAM	Menjalankan aplikasi konsumsi memori tinggi	CLI menampilkan % RAM naik akurat	CLI menampilkan kenaikan sesuai `free -h`	Pass
3	Pembacaan Data Disk	Mengisi disk dengan file dummy	CLI menampilkan % Disk naik akurat	CLI menampilkan kapasitas terpakai bertambah	Pass
4	Trigger Notifikasi CPU	Membebani CPU > 80%	Notifikasi Telegram masuk	Notifikasi diterima dalam < 5 detik	Pass
5	Kondisi Normal	Server idle (beban < 50%)	Tidak ada notifikasi terkirim	Tidak ada notifikasi masuk	Pass
6	Format Output CLI	Menjalankan `python main.py`	Tampilan rapi, teks terbaca jelas	Tampilan tabel CLI rapi dan berwarna	Pass

5.3 Analisis Hasil Fungsional

Berdasarkan Tabel 5.1, seluruh skenario pengujian berstatus **Pass**.

1. **Akurasi Data:** Data yang ditampilkan pada CLI memiliki selisih yang tidak signifikan (<1%) dibandingkan dengan alat monitoring bawaan Linux seperti `top` atau `htop`. Hal ini membuktikan bahwa pustaka `psutil` mampu mengambil data sistem secara akurat.

2. **Mekanisme Alert:** Sistem berhasil mendeteksi ketika penggunaan sumber daya melampaui threshold yang ditentukan pada `config.json`. Notifikasi Telegram diterima oleh administrator secara real-time.

3. **Logika Cooldown:** Pengujian menunjukkan bahwa ketika beban tinggi berlangsung selama 10 menit, notifikasi hanya dikirimkan sekali pada menit pertama, dan tidak mengirim ulang hingga periode `cooldown` (5 menit) selesai. Ini membuktikan mekanisme anti-spam berfungsi dengan baik.

BAB VI

HASIL PENGUJIAN PERFORMA DAN PEMBAHASAN

6.1 Pengujian Konsumsi Sumber Daya Sistem Monitoring

Salah satu tujuan utama penelitian ini adalah menciptakan sistem monitoring yang "ringan". Untuk membuktikannya, dilakukan pengukuran konsumsi CPU dan RAM oleh proses `python main.py` itu sendiri selama 30 menit dalam tiga kondisi beban server (Idle, Sedang, Penuh).

Tabel 6.1 Rata-rata Konsumsi Resource oleh Script Monitoring

Kondisi Server	Rata-rata CPU Script (%)	Rata-rata RAM Script (MB)
Idle (Beban 0-10%)	0.2%	18.5 MB
Sedang (Beban 50%)	0.4%	19.1 MB
Penuh (Beban 90%+)	0.5%	20.3 MB

Analisis: Hasil pengujian menunjukkan bahwa skrip monitoring sangat efisien. Konsumsi CPU tidak pernah melebihi 0.5% bahkan saat server dalam beban penuh. Konsumsi memori juga stabil di bawah 25 MB. Hal ini jauh lebih ringan dibandingkan tools berbasis GUI atau web dashboard seperti Zabbix atau Prometheus yang disebutkan dalam Tinjauan Pustaka (BAB II), yang biasanya membutuhkan ratusan MB RAM untuk agen monitoringnya.

6.2 Pengujian Latensi Notifikasi

Pengujian latensi mengukur waktu yang dibutuhkan sejak threshold terlampaui hingga notifikasi muncul di perangkat seluler administrator. Pengujian dilakukan sebanyak 10 kali percobaan.

Tabel 6.2 Hasil Pengujian Latensi Notifikasi

Percobaan	Waktu Threshold Terlewati	Waktu Notifikasi Diterima	Durasi (Detik)
1	10:00:00	10:00:02	2
2	10:05:00	10:05:03	3
3	10:10:00	10:10:02	2
...	...	...	...
10	10:45:00	10:45:04	4
Rata-rata			**2.8 Detik**

Analisis: Rata-rata waktu respons adalah 2.8 detik. Ini memenuhi kriteria "Real-Time" yang didefinisikan dalam penelitian ini (di bawah 5 detik). Kecepatan ini dipengaruhi oleh kecepatan API Telegram dan kestabilan internet server.

6.3 Pembahasan

Secara keseluruhan, sistem yang dibangun telah memenuhi tujuan penelitian yang tertuang pada BAB I.

1. **Efisiensi:** Penggunaan CLI dan Python terbukti menjadi solusi alternatif yang efektif untuk server dengan spesifikasi terbatas dibandingkan implementasi Zabbix yang kompleks.

2. **Keandalan:** Mekanisme notifikasi melalui Telegram Bot API handal dan dapat diandalkan untuk peringatan dini tanpa biaya infrastruktur tambahan (seperti server SMTP sendiri).

3. **Keterbatasan:** Sistem ini bergantung pada koneksi internet untuk notifikasi. Jika server kehilangan koneksi internet, notifikasi tidak akan terkirim meskipun monitoring lokal tetap berjalan. Selain itu, threshold yang statis memerlukan penyesuaian manual jika pola beban server berubah.

BAB VII

PENUTUP

7.1 Kesimpulan

Berdasarkan hasil implementasi dan pengujian yang telah dilakukan, dapat ditarik beberapa kesimpulan sebagai berikut:

1. **Berhasil Dibangun:** Aplikasi Sistem Monitoring Resource Server Berbasis CLI berhasil dibangun menggunakan bahasa pemrograman Python dan pustaka `psutil`. Sistem mampu menampilkan data penggunaan CPU, RAM, dan Disk secara akurat dengan konsumsi sumber daya yang sangat rendah (rata-rata < 0.5% CPU dan < 25 MB RAM).

2. **Notifikasi Real-Time:** Fitur notifikasi real-time melalui Telegram Bot berhasil diimplementasikan dengan rata-rata latensi 2.8 detik. Mekanisme `cooldown` efektif mencegah banjir notifikasi saat terjadi anomali berkepanjangan.

3. **Efisiensi Operasional:** Sistem ini memberikan alternatif solusi monitoring yang ringan bagi administrator sistem yang membutuhkan pemantauan sederhana tanpa kompleksitas konfigurasi infrastruktur monitoring skala besar (Enterprise).

7.2 Saran

Untuk pengembangan penelitian selanjutnya, penulis menyarankan beberapa hal guna meningkatkan kualitas sistem:

1. **Threshold Dinamis:** Mengembangkan algoritma sederhana untuk menentukan threshold secara otomatis berdasarkan rata-rata beban server selama periode tertentu (misalnya 7 hari terakhir), sehingga administrator tidak perlu menentukan nilai statis secara manual.

2. **Multi-Channel Notification:** Menambahkan saluran notifikasi lain selain Telegram, seperti Email (SMTP) atau WhatsApp Gateway, sebagai cadangan jika salah satu layanan mengalami gangguan.

3. **Penyimpanan Data Historis:** Menambahkan fitur pencatatan data ke dalam database ringan (seperti SQLite) untuk memungkinkan analisis tren penggunaan resource secara berkala, bukan hanya monitoring real-time.

4. **Manajemen Proses:** Menambahkan fitur untuk melihat daftar proses yang paling banyak mengonsumsi resource secara spesifik melalui CLI, sehingga administrator dapat langsung mengetahui penyebab beban tinggi.

DAFTAR PUSTAKA

Adiputra, R., Santoso, B., & Wicaksono, T. (2022). Pengembangan Tools Monitoring CPU dan Memori Berbasis CLI untuk Server Linux. Jurnal Teknologi Sistem Informasi, 8(2), 112–121.

Arisanti, D., & Prasetijo, A. B. (2022). Implementasi Sistem Monitoring Jaringan Berbasis Zabbix di Badan Pengawasan Keuangan dan Pembangunan (BPKP). Jurnal Informatika dan Rekayasa Elektronik, 3(1), 45–53.

Firmansyah, A., Nugraha, D., & Setiawan, H. (2023). Pengembangan Skrip Python untuk Otomatisasi Monitoring dan Pelaporan Kinerja Server. Jurnal Ilmu Komputer dan Sistem Informasi, 11(1), 22–31.

Handoko, B., & Priyambodo, T. K. (2024). Efisiensi Konsumsi Memori pada Skrip Monitoring Python: Studi Kasus Penggunaan psutil vs Platform Modul Natif. Journal of Computer Science and Engineering, 5(1), 14–24.

Haryadi, F., Kurniawan, D., & Laksono, P. (2023). Analisis Performa Tools Monitoring Server: Nagios vs Prometheus. Jurnal Sistem dan Teknologi Informasi, 11(3), 201–210.

Kusuma, A., & Hidayat, R. (2022). Perancangan Sistem Monitoring Infrastruktur IT Berbasis Open Source menggunakan Prometheus dan Grafana. Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK), 8(4), 789–798.

Lestari, N., & Setiawan, G. (2023). Integrasi Platform Messaging untuk Alerting pada Sistem Monitoring Server Berbasis Microservice. Jurnal Ilmiah Informatika Komputer, 28(3), 177–188.

Munir, A., Hidayat, S., & Fauzi, R. (2022). Sistem Deteksi Bahaya Berbasis IoT dengan Pemantauan Adaptif untuk Efisiensi Energi. Jurnal Nasional Teknik Elektro dan Teknologi Informasi (JNTETI), 10(2), 88–96.

Nugroho, S., & Santoso, A. (2022). Rancang Bangun Sistem Monitoring Server Menggunakan Protokol SNMP Berbasis Web. Jurnal Teknik Informatika dan Sistem Informasi (JuTISI), 7(1), 56–65.

Pratama, R., & Wijaya, H. (2022). Pengembangan Dashboard Monitoring Server Berbasis Web Menggunakan Framework Laravel. Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 6(5), 2310–2319.

Putra, D. A., & Maharani, S. (2022). Lightweight Server Health Monitoring Menggunakan Shell Script pada Lingkungan Linux. Jurnal Rekayasa Teknologi Informasi (JURTI), 6(2), 134–142.

Rahmawati, I., Susanto, E., & Prasetya, A. (2022). Implementasi Notifikasi Real-Time pada Sistem Monitoring Jaringan Menggunakan Telegram Bot. Jurnal Informatika Upgris, 8(2), 101–110.

Sari, M. P., & Kurniawan, B. (2023). Implementasi Sistem Peringatan Dini Server Overload Berbasis Threshold Dinamis. Jurnal Ilmu Teknik Elektro Komputer dan Informatika (JITEKI), 9(1), 45–56.

Smith, J., Brown, A., & Davis, C. (2022). On-Device Artificial Intelligence for Real-Time Server Resource Monitoring. International Journal of Computer Applications, 183(12), 1–8.

Wibowo, S., & Purnama, C. (2022). Analisis Konsumsi Sumber Daya Sistem Monitoring: Perbandingan Pendekatan Berbasis Agent dan Agentless. Jurnal Sistem Informasi dan Teknologi, 4(2), 88–97.