INSECT: Jurnal Teknik Informatika
INFORMATICS AND SECURITY e-ISSN. 2614-431X

Desain dan Implementasi Cron Gateway Berbasis Web Menggunakan PHP-CLI dengan Penyimpanan Berbasis JSON

Joko Supriyanto*[1] , Wicaksono Yuli Sulistyo[2]
[12]Prodi Informatika, Universitas Siber Muhammadiyah
E-mail: ***[1]**joko@sibermu.ac.id, [2]wicaksono@sibermu.ac.id

Abstrak

Konfigurasi cron tradisional masih bergantung pada antarmuka baris perintah (CLI) yang rumit, rawan kesalahan, dan tidak memiliki pemantauan terintegrasi. Hal ini menyulitkan administrator, terutama yang kurang mahir menggunakan shell. Cron Gateway adalah aplikasi yang menyederhanakan dan memusatkan pengelolaan cron jobs Linux dengan menghadirkan antarmuka web visual dan interaktif sebagai pengganti perintah teks di command-line. Cron Gateway berbasis web yang dikembangkan dengan PHP, memisahkan antarmuka manajemen (web UI) dan mesin eksekusi (CLI). Konfigurasi disimpan dalam satu berkas JSON terpusat sehingga portabel tanpa butuh database eksternal. Sistem mendukung penjadwalan hibrida (interval sederhana & format cron) serta menyediakan parser untuk menerjemahkan ekspresi cron ke bahasa alami secara realtime. Pengujian penggunaan Cron Gateway di aplikasi https://kaldik.sibermu.ac.id diantaranya mampu menjalankan tugas asinkronus untuk mengirimkan kegiatan di google kalender berinterval 10 detik hingga lebih dari 15.300 kali, menjalankan Auto sinkron kirim Google kalender berinterval 10 detik hingga lebih dari 4.753 kali dan mampu mengirimkan agenda kegiatan di google kalender ke 4399 mahasiswa, menjalan proses penghapusan log berinterval 1 hari sekali setiap jam 00:00 dijalankan 2 kali, menjalankan ringkasan harian menggunakan AI untuk kalender akademik berinterval setiap hari jam 00:01 dijalankan 2 kali semua berjalan dengan stabil. Aplikasi mampu menjalankan tugas harian tepat waktu sesuai jadwal. Semua tugas tercatat "Success" dengan log eksekusi akurat, membuktikan sistem ini andal, presisi, dan efektif di lingkungan produksi. Sistem ini memudahkan penjadwalan, menyediakan log & pemantauan terpusat, serta mengurangi beban kognitif administrator dengan umpan balik interaktif sehingga manajemen tugas server menjadi lebih efisien.

***Kata Kunci**:Cron Gateway, Cron, JSON*

1. PENDAHULUAN

Penjadwalan tugas otomatis merupakan salah satu komponen penting dalam pengelolaan server berbasis sistem operasi Linux. Umumnya, penjadwalan ini dilakukan menggunakan *cron*, yaitu layanan yang memungkinkan eksekusi perintah secara terjadwal berdasarkan interval waktu tertentu[1]. Meskipun telah lama menjadi standar, konfigurasi *cron* tradisional masih mengandalkan antarmuka baris perintah (*Command Line Interface*/CLI) yang memiliki sejumlah keterbatasan[2]. Kompleksitas sintaks, potensi kesalahan manusia (*human error*), serta ketiadaan fitur pemantauan terpusat sering kali menjadi kendala, terutama bagi administrator yang kurang berpengalaman dalam operasi shell[3].

Seiring meningkatnya kebutuhan akan sistem yang lebih andal, efisien, dan mudah digunakan, dibutuhkan sebuah solusi yang mampu menyederhanakan pengelolaan *cron jobs* sekaligus memberikan visibilitas yang lebih baik terhadap proses penjadwalan. *Cron Gateway* hadir sebagai jawaban atas permasalahan tersebut. Aplikasi ini dirancang untuk memusatkan manajemen *cron*

jobs melalui antarmuka web yang visual, interaktif, dan ramah pengguna. Dengan pendekatan ini, administrator tidak lagi harus bergantung sepenuhnya pada CLI yang rawan kesalahan.
Selain menyederhanakan konfigurasi, *Cron Gateway* juga menawarkan arsitektur berbasis web yang dikembangkan dengan PHP 8.4 [4] dan menggunakan berkas JSON[5] terpusat sebagai media penyimpanan konfigurasi. Hal ini membuat sistem lebih portabel karena tidak membutuhkan *database* eksternal. Lebih lanjut, fitur penjadwalan hibrida serta parser ekspresi *cron* yang mampu menerjemahkan *sintaks* menjadi bahasa alami secara *real-time* menambah nilai guna aplikasi ini. Penelitian ini juga menekankan pada aspek fungsionalitas dan keandalan sistem dalam lingkungan operasional nyata.

2. METODE PENELITIAN

Tahap awal pengembangan aplikasi *Cron Gateway* adalah identifikasi masalah yang didasarkan pada observasi langsung di lapangan. Observasi ini bertujuan untuk memahami secara mendalam alur kerja (*workflow*) yang ada, mengidentifikasi tantangan operasional, serta menganalisis kebutuhan fungsional sistem secara empiris. Data dan informasi yang terkumpul dari proses ini menjadi landasan krusial untuk perancangan dan pengembangan sistem. Dari hasil pengamatan di lapangan *cron* bawaan Linux tidak bisa membuat jadwal di bawah 1 menit, di satu sisi untuk mempercepat proses sinkronisasi data beberapa pengembang aplikasi berbasis web membutuhkan penjadwalan dalam hitungan detik contohnya pengiriman kalender akademik ke *4399* mahasiswa perlu penjadwalan hitungan detik. Jika di buat 1 menit sekali maka proses pengirimannya bisa jauh lebih lama, sehingga aplikasi *Cron Gateway* ini menjadi kebutuhan wajib harus ada dikarenakan kemampuannya bisa membuat penjadwalan dalam hitungan detik yang tidak bisa dilakukan *cron* bawaan Linux. Metodologi yang diterapkan dalam penelitian ini adalah *System Development Life Cycle* (SDLC)[6] dengan model *waterfall*. Model *waterfall* membagi tahapan pengembangan sistem ke dalam alur yang bersifat linier, di mana setiap tahap harus diselesaikan terlebih dahulu sebelum melanjutkan ke tahap berikutnya[7].

2.1. Pendekatan Penelitian

Penelitian ini menggunakan pendekatan Rekayasa Perangkat Lunak (Software Engineering) dengan fokus pada Desain dan Kreasi (*Design and Creation*). Pendekatan ini dipilih karena tujuan utama penelitian adalah untuk menghasilkan berupa perangkat lunak (*Cron Gateway*) yang fungsional sebagai solusi atas permasalahan yang teridentifikasi, yaitu kompleksitas dan rendahnya usabilitas manajemen *cron job* tradisional. Penelitian ini berfokus pada proses konstruksi dan evaluasi aplikasi tersebut untuk membuktikan kelayakan dan efektivitasnya.

2.2. Analisis Kebutuhan Sistem

Tahap awal ini berfokus pada identifikasi masalah dan penentuan kebutuhan fungsional serta non-fungsional sistem. Tahap awal adalah mengidentifikasi masalah yaitu manajemen *cron job* melalui CLI (misalnya, crontab -e) bersifat teknis, tidak memiliki antarmuka visual, rawan kesalahan sintaks, dan sulit untuk dipantau secara terpusat. Selanjutnya melakukan explorasi kebutuhan fungsional:

a. Sistem harus dapat melakukan operasi CRUD (*Create, Read, Update, Delete*) untuk tugas terjadwal melalui antarmuka web.
b. Sistem harus mendukung dua jenis penjadwalan: interval waktu sederhana (per detik) dan format cron standar.
c. Sistem harus menampilkan status eksekusi terakhir (berhasil/gagal), waktu eksekusi, dan output log dari setiap tugas.
d. Sistem harus dapat mengaktifkan dan menonaktifkan tugas tanpa menghapusnya.

Selanjutnya melakukan explorasi penjabaran kebutuhan Non-Fungsional yaitu dengan Analisis kebutuhan nonfungsional memberikan deskripsi mengenai spesifikasi sistem yang berfokus pada karakteristik perilaku yang mendukung kelancaran proses pengembangan serta pengujian. Dalam perancangan sistem, aspek yang perlu dipertimbangkan meliputi kebutuhan perangkat keras

(*hardware*), perangkat lunak (*software*), dan pengguna (*user*) sebagai dasar analisis yang harus dipenuhi sebelum sistem diimplementasikan yaitu :

a. Analisis kebutuhan *hardware* yang diperlukan pada proses pembangunan dan implementasi terdiri dari: Server processor Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz. Memory 64GB
b. Analisis kebutuhan *software* yang diperlukan pada proses pembangunan dan implementasi terdiri dari:
    a) Virtual Box 7.1.4 r165100
    b) Ubuntu Server 20
    c) NGINX 1.28
    d) PHP versi 8.4.7
    e) Browser Crome Version 138.0.7204.185 (Official Build) (64-bit)
c. Analisis Pengguna (*User*) Tahapan ini untuk mengetahui siapa saja aktor yang terlibat dalam menjalankan sistem. Pengguna sistem adalah administrator pengelola server dan developer aplikasi berbasis php.

2.3. Teknik Pengumpulan Data

Pengumpulan data dalam penelitian ini dilakukan untuk mengevaluasi kinerja dan keandalan sistem.

a. Observasi Langsung: Melakukan pengamatan langsung terhadap antarmuka web (index.php) dalam lingkungan produksi untuk memonitor status, jumlah eksekusi, dan waktu eksekusi terakhir dari setiap tugas secara *real-time*.
b. Analisis Dokumen (Log): Menganalisis konten file jobs.json dan file log individual di direktori /logs. Data yang dikumpulkan meliputi *timestamp* eksekusi, status keberhasilan, dan *output* tekstual yang dihasilkan oleh skrip.

2.4. Teknik Analisis Data

Data yang telah dikumpulkan dianalisis menggunakan teknik analisis deskriptif kualitatif.

a. Analisis Kinerja: Data kuantitatif seperti *execution_count* dan *last_run* dianalisis untuk menyimpulkan stabilitas dan keandalan sistem. Sebagai contoh, jumlah eksekusi yang mencapai ribuan untuk tugas interval menunjukkan bahwa sistem stabil untuk operasi berfrekuensi tinggi.
b. Analisis Fungsionalitas: Data kualitatif seperti *last_status* ("*Success*") dan konten *last_output* dianalisis untuk memvalidasi bahwa setiap fungsi berjalan sesuai harapan dan menghasilkan output yang benar.
c. Analisis Usabilitas: Antarmuka pada hasil cron.jpg dianalisis secara kualitatif untuk menilai sejauh mana fitur-fitur seperti penjelasan jadwal dan status berkode warna berhasil menyederhanakan proses manajemen bagi pengguna.

3. HASIL DAN PEMBAHASAN

3.1. Perancangan Sistem

Pada tahap ini, arsitektur dan desain detail sistem dirancang berdasarkan kebutuhan yang telah didefinisikan.

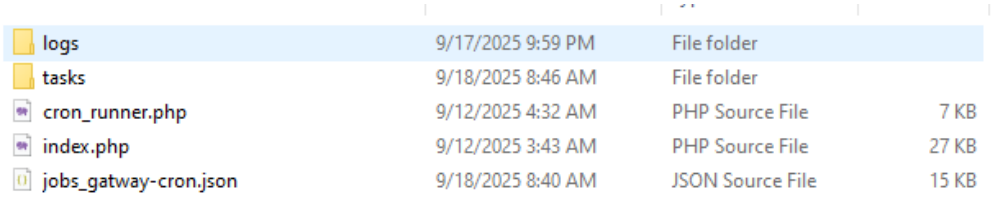
Desain Arsitektur, sistem dirancang dengan arsitektur dua komponen utama:

a. Antarmuka Manajemen (*Frontend*): Sebuah aplikasi web (index.php) yang berfungsi sebagai dasbor visual untuk interaksi pengguna.
b. Mesin Eksekutor (*Backend*): Sebuah skrip CLI (cron_runner.php) yang berjalan di latar belakang untuk mengeksekusi tugas sesuai jadwal.
c. Desain Basis Data: Sistem menggunakan file jobs.json sebagai media penyimpanan tunggal untuk semua konfigurasi tugas. Struktur data untuk setiap objek tugas mencakup

atribut seperti id, name, script_path, schedule_type, schedule, is_enabled, last_run, last_status, last_output, dan execution_count.

d. Desain Antarmuka Pengguna (UI)[8]: Antarmuka dirancang menggunakan *framework Bootstrap*[9] untuk memastikan tampilan yang responsif dan modern. Desain berfokus pada penyajian informasi yang jelas, seperti status tugas yang diberi kode warna, jadwal yang diterjemahkan, dan pratinjau log.

Desain Arsitektur Antarmuka Manajemen (*Frontend*) untuk mempermudah pemahaman mengenai implementasi *Cron Gateway,* disajikan visualisasi berupa gambar struktur berkas sebagai berikut



Gambar 1 Stuktur berkas *Cron Gateway*

Pada gambar 1 merupakan struktur berkas dari *Cron Gateway* dengan penjelasannya sebagai berikut :

File Utama

a. index.php *file* ini adalah antarmuka pengguna (UI) berbasis web. Fungsinya adalah sebagai dasbor untuk mengelola semua tugas *cron*. Halaman ini terdiri dari:
   1) Melihat semua daftar tugas yang ada beserta statusnya (aktif/nonaktif), jadwal, kapan terakhir dijalankan, dan cuplikan output terakhir.
   2) Menambah tugas baru melalui sebuah *form* modal.
   3) Mengedit konfigurasi tugas yang sudah ada.
   4) Menghapus tugas.
   5) Mengaktifkan atau menonaktifkan tugas secara cepat.
   6) Melihat log eksekusi lengkap dari setiap tugas.

b. cron_runner.php *file* Ini adalah mesin eksekutor di latar belakang. *File* ini tidak untuk diakses melalui browser, melainkan dijalankan secara otomatis oleh sistem server (melalui cron job sistem yang diatur untuk berjalan setiap menit). Tugas utamanya adalah:
   1) Membaca *file jobs*_gatway-cron.json untuk mendapatkan daftar semua tugas.
   2) Memeriksa jadwal setiap tugas yang aktif.
   3) Jika sebuah tugas sudah waktunya dijalankan (baik berdasarkan interval detik atau ekspresi *cron*), cron_runner.php akan mengeksekusi file skrip PHP yang sesuai.
   4) Mencatat *output*, status (berhasil/gagal), dan waktu eksekusi ke dalam *file* log di folder /*logs*.
   5) Memperbarui status terakhir di jobs_gatway-cron.json.

c. jobs_gatway-cron.json f*ile* ini berfungsi sebagai *database* sederhana dalam format JSON. Isinya adalah sebuah *array* yang menyimpan semua konfigurasi dan status dari setiap tugas yang Anda buat. Informasi yang disimpan meliputi nama tugas, *path* skrip, jadwal, status aktif/nonaktif, dan histori eksekusi terakhir. Baik index.php maupun cron_runner.php membaca dan menulis ke *file* ini.

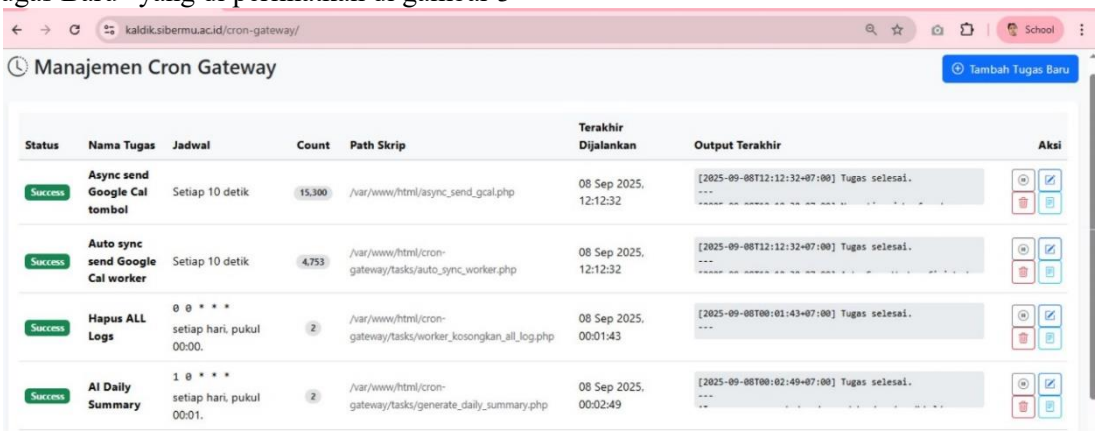
Folder

a. /logs
   1) Fungsi: Folder ini adalah tempat penyimpanan semua catatan (log) eksekusi. Setiap kali cron_runner.php menjalankan sebuah tugas, akan membuat atau menambahkan catatan ke dalam file log di dalam folder ini. Nama file log sesuai

dengan ID unik dari tugas (contoh: job_68ba50436dab8.log). Ini sangat berguna untuk melacak riwayat eksekusi dan melakukan *troubleshooting* jika terjadi kesalahan.

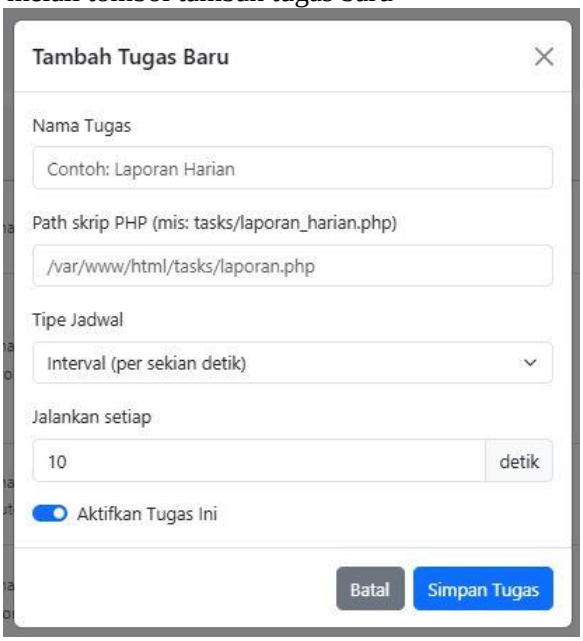
b. /tasks fungsi folder ini berisi kumpulan skrip PHP yang sebenarnya akan dijalankan secara terjadwal. Ini adalah "pekerjaan" atau "tugas" inti. Contohnya, di dalam folder ini bisa terdapat file seperti cron_sync_moodle.php atau generate_daily_summary.php. Sistem Cron Gateway ini hanya bertugas sebagai penjadwal dan eksekutor, sementara logika atau pekerjaan spesifik yang harus dilakukan berada di dalam *file-file* di folder /tasks ini.

Tampilan Antarmuka Manajemen (*Frontend*) bisa Anda lihat di gambar 2 sudah dilakukan penginputan *task* dan bisa disimpan dengan sempurna, memasukkan *task* melalui modal "Tambah Tugas Baru" yang di perlihatkan di gambar 3



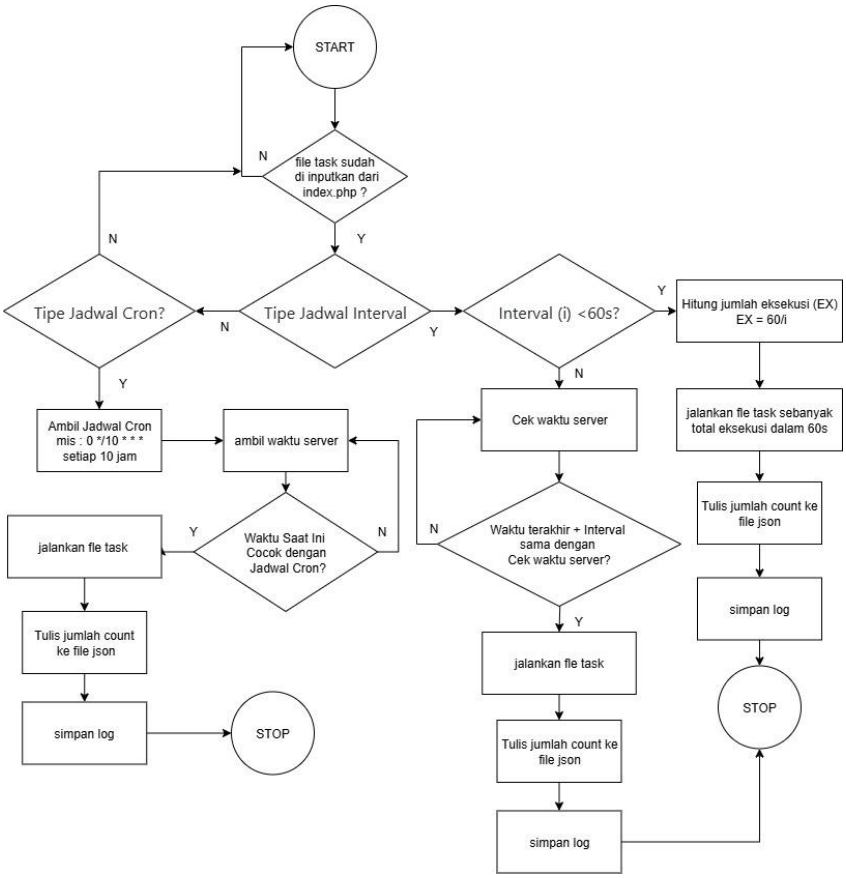
Gambar 2 Tampilan UI index.php untuk menginputkan task

Sedangkan untuk menginputkan *task* bisa melihat di gambar 3 dan merupakan bagian dari index.php yang diakses melali tombol tambah tugas baru



Gambar 3 Tampilan modal "Tambah Tugas Baru" bagian dari index.php

Desain arsitektur mesin eksekutor (*Backend*)[10] dari dari cron_runner.php yang merupakan jantung *Cron Gateway* bisa bekerja diperlihatkan gambar 4



Gambar 4 Arsitektur berkas cron_runner.php jantung dari *Cron Gateway*

Indeks.php hanya di gunakan sebagai User Inteface untuk mempermudah memasukkan *task*, sedangkan jantung dari *Cron Gateway* ini ada di berkas cron_runner.php yang di eksekusi setiap 1 menit sekali oleh cron sistem operasi Linux, sehingga administrator server hanya membuat *cron* tradisional cukup sekali saja.

Desain Basis Data *file jobs*_gateway-cron.json adalah *database*[11] konfigurasi sederhana untuk sistem *Cron Gateway*. File JSON tersebut berisi sebuah *array* (daftar) dari objek-objek tugas[12]. Setiap objek dalam *array* tersebut merepresentasikan satu tugas cron yang telah didefinisikan dari index.php. struktur *file jobs*_gateway-cron.json diperlihatkan di gambar 5


```
[
    {
        "name": "Async send Google Cal",
        "script_path": "\/mnt\/disktambahanweb\/www\/kaldik.jokode.my.id\/async_send_gcal.php",
        "is_enabled": true,
        "schedule_type": "interval",
        "interval_seconds": 10,
        "schedule": null,
        "id": "job_68ba50436dab8",
        "created_at": "2025-09-05T09:51:47+07:00",
        "last_run": "2025-09-18T08:40:32+07:00",
        "last_status": "success",
        "last_output": "[2025-09-18T08:40:32+07:00] Starting async worker...\n[2025-09-18T08:40:32+07:00] No active jobs found. Exiting.\n",
        "execution_count": 34231
    },
    {
        "name": "sync user LMS solusi",
        "script_path": "\/mnt\/disktambahanweb\/www\/kaldik.jokode.my.id\/cron-gateway\/tasks\/cron_sync_moodle.php",
        "is_enabled": false,
        "schedule_type": "cron",
        "schedule": "0 0 * * 5",
        "interval_seconds": null,
        "id": "job_68ba5087e3684",
        "created_at": "2025-09-05T09:52:55+07:00",
        "last_run": null,
        "last_status": "pending",
        "last_output": "",
        "execution_count": 0
    }
]
```


Gambar 5 Struktur *file* JSON sebagai media penyimpanan *task* pada *Cron Gateway*

Penjelasan Setiap Atribut (*Key-Value Pair*) pada gambar 5 sebagai berikut :

a. *id (String):* Pengenal unik untuk setiap tugas.
b. *name (String):* Nama tugas yang deskriptif untuk ditampilkan.
c. *path (String):* Lokasi *file* skrip PHP yang akan dijalankan.
d. *schedule (String):* Jadwal eksekusi, baik dalam format *cron* ("0 2 * * *") maupun interval detik ("300").
e. *active (Boolean):* Menentukan apakah tugas aktif (*true*) atau tidak (*false*).
f. *created_at (String):* Waktu dan tanggal tugas dibuat.
g. *last_run_time (String):* Waktu dan tanggal tugas terakhir kali dijalankan.
h. *last_output (String):* Ringkasan hasil atau *output* dari eksekusi terakhir, berguna untuk debugging.
i. *last_status (String):* Status eksekusi terakhir, misalnya "*success*" atau "*failed*".
j. *next_run_display (String):* Perkiraan waktu eksekusi berikutnya dalam format yang mudah dibaca.

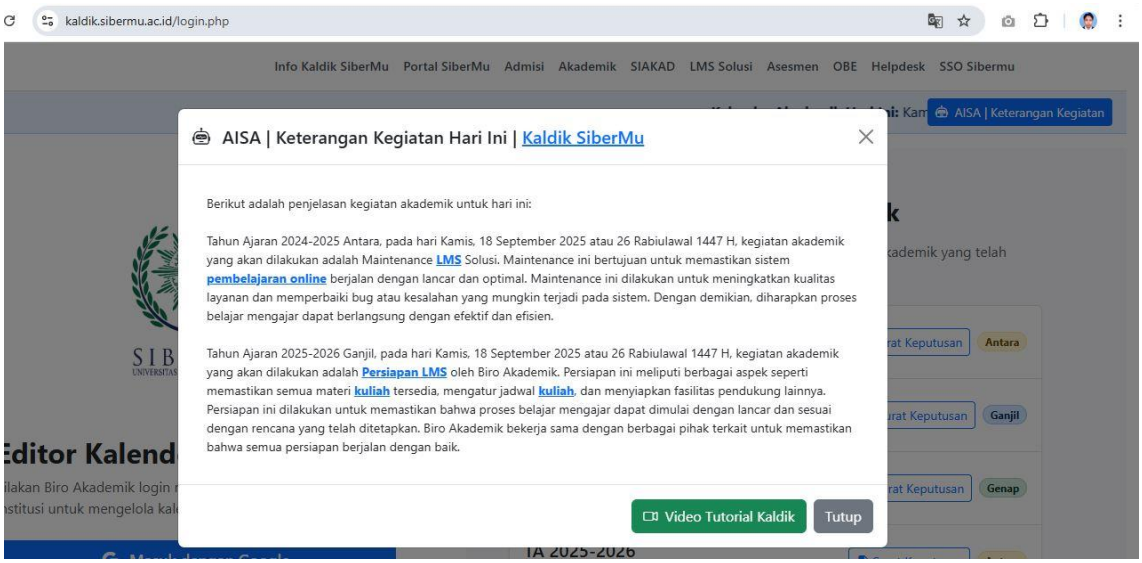
3.2 Pengujian Sistem

Tahap pengujian dilakukan untuk memverifikasi bahwa sistem berjalan sesuai dengan kebutuhan dan bebas dari eror. Metode Pengujian menggunakan metode Black Box Testing, yaitu pengujian difokuskan pada fungsionalitas sistem dari perspektif pengguna tanpa perlu mengetahui detail implementasi internal[13], dengan skenario pengujian sebagai berikut:

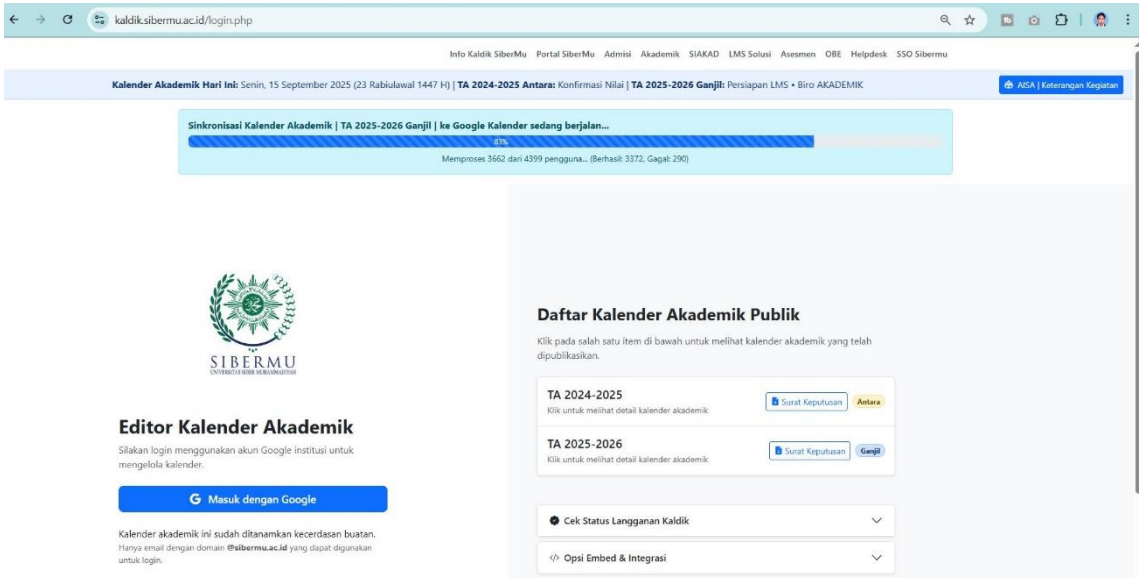
a. Membuat tugas dengan penjadwalan interval (misal, setiap 10 detik) dan memverifikasi bahwa *execution_count* serta *last_run* diperbarui secara berkala.
b. Membuat tugas dengan penjadwalan *cron* (misal, setiap hari pukul 00:00) dan memverifikasi bahwa tugas hanya dieksekusi sekali sehari pada waktu yang tepat.
c. Menguji semua fungsi CRUD (*create, read update, delete*)[14] melalui antarmuka web.
d. Memverifikasi bahwa status "*Success*" atau "*Error*" dan *output* log tercatat dengan benar setelah eksekusi.

Hasil pengujian *Cron Gateway* yang di gunakan di https://kaldik.sibermu.ac.id yang merupakan portal informasi kalender akademik di tunjukkan pada gambar 2 hasil pengujian menunjukkan bahwa :

1. Proses penyimpanan penjadwalan *task* bisa berjalan dengan sempurna.
2. Menjalankan tugas asinkronus[15] dengan nama tugas Async send Google Cal Tombol untuk mengirimkan kegiatan di google kalender menggunakan tombol kirim berinterval 10 detik dijalankan hingga lebih dari 15.300 kali
3. Menjalankan tugas asinkronus dengan nama Auto Sync send google Cal Worker yang merupakan Auto sinkron kirim Google kalender berinterval 10 detik dijalankan hingga lebih dari 4.753 kali. Bukti pengujian penggunaan *Cron Gateway* ini berjalan di tunjukkan di gambar 7 yang mampu mengirimkan data berupa agenda kegiatan di google kalender dengan total pengguna 4399 tanpa kesalahan.
4. Menjalan proses penghapusan log dengan nama tugas Hapus *All Logs* berinterval 1 hari sekali setiap jam 00:00 sudah berhasil dijalankan 2 kali
5. Menjalankan ringkasan harian menggunakan AI (Kecerdasan Buatan) API di hubungkan dengan https://openrouter.ai menggunakan model AI llama 4 Maverick[16] dengan nama tugas AI Daily Summary untuk kalender akademik berinterval setiap hari jam 00:01 sudah berhasil dijalankan 2 kali dan di perlihatkan di gambar 6



Gambar 6 C*ron Gateway* AI Daily Summary di kalender akademik dihasilkan AI



Gambar 7 Hasil *Cron Gateway* nama *task* Auto Sync send google Cal Worker

Dari hasil pengujian tersebut kesimpulannya bahwa semua pengujian berjalan lancar dan stabil.

4. KESIMPULAN

Sistem "*Cron Gateway*" telah berhasil dirancang, diimplementasikan, dan diuji untuk menyediakan antarmuka manajemen tugas terjadwal yang lebih mudah digunakan daripada cron tradisional pada sistem operasi Linux.

Ringkasan Hasil

1. Arsitektur Sistem: Sistem ini dibangun dengan arsitektur dua komponen utama yang solid: antarmuka manajemen (*frontend*) berbasis web (index.php) untuk interaksi pengguna dan mesin eksekutor (*backend*) berbasis CLI (cron_runner.php) yang berjalan di latar belakang untuk mengeksekusi tugas.
2. Penyimpanan Data: Sistem secara efektif menggunakan satu *file* json sebagai basis data sederhana untuk menyimpan semua konfigurasi dan status tugas, menyederhanakan pengelolaan tanpa memerlukan *database* server terpisah.

3. Fungsionalitas: *Frontend* yang dirancang dengan Bootstrap menyediakan dasbor visual yang responsif dan intuitif untuk mengelola tugas (menambah, mengedit, menghapus, dan melihat log). Sementara itu, *backend* berfungsi sebagai inti sistem yang secara andal memeriksa dan menjalankan setiap tugas sesuai jadwalnya.
4. Hasil Pengujian: Pengujian *Black Box* pada sistem yang diterapkan di https://kaldik.sibermu.ac.id menunjukkan bahwa *Cron Gateway* berjalan dengan sukses, stabil, dan sesuai rencana. Bukti keberhasilannya meliputi:
    a. Eksekusi dengan jumlah 15.300 kali dan 4.753 kali tugas asinkron dengan interval 10 detik untuk sinkronisasi Google Calendar tanpa kesalahan.
    b. Penjadwalan tugas harian, seperti penghapusan log dan pembuatan ringkasan berbasis AI, yang berjalan tepat waktu.
    c. Semua fungsi manajemen tugas melalui antarmuka pengguna berfungsi dengan sempurna.

Melalui integrasi dengan aplikasi akademik seperti https://kaldik.sibermu.ac.id, *Cron Gateway* diuji untuk menangani berbagai skenario otomatisasi, seperti sinkronisasi Google Kalender secara asinkron, penghapusan log terjadwal, hingga pembuatan ringkasan harian menggunakan AI. Hasil pengujian menunjukkan bahwa sistem mampu bekerja dengan stabil, tepat waktu, serta mendukung ribuan eksekusi berulang tanpa kegagalan.

Dengan demikian, keberadaan *Cron Gateway* tidak hanya menawarkan kemudahan bagi administrator dalam mengelola *cron jobs,* tetapi juga memberikan kontribusi nyata terhadap peningkatan efisiensi, akurasi, dan skalabilitas manajemen server. Implementasi ini diharapkan dapat menjadi alternatif praktis sekaligus inovatif dalam modernisasi pengelolaan tugas terjadwal di lingkungan produksi.

5. SARAN

1. Idempoten: Idealnya, menjalankan cron_runner.php berkali-kali dalam waktu singkat tidak boleh menyebabkan tugas berjalan berulang kali secara tidak sengaja. Mekanisme *last_run_time* dan jadwal harus mencegah ini dan di C*ron Gateway* ini belum mampu menangani hal tersebut.
2. Konkurensi: Jika cron_runner.php dijalankan terlalu sering atau jika ada tugas yang berjalan sangat lama, masih ada risiko beberapa *instance* cron_runner.php berjalan bersamaan dan mencoba memodifikasi jobs_gateway-cron.json secara bersamaan, menyebabkan korupsi data. Sehingga perlu adanya mekanisme *file locking* (misalnya, flock()) bisa digunakan untuk mencegah ini.
3. Logging Detail: saat ini cron gateway belum mencatat Log *error* internal cron_runner sendiri (misalnya, gagal membaca JSON)[17].

DAFTAR PUSTAKA

[1] A. Cimino, “Master ’ s Degree programme Computer Science and Information Technology - CM90 CronFrame : A Macro Annotation Cron Job Framework with Web Server and CLI Tool written in Rust,” no. 897613, pp. 1–168, 2024.

[2] L. Terecia and S. Kosasi, “Automatic Daily Scheduling System Using Cron Jobs for Handicraft Business Management,” no. September, pp. 547–552, 2025.

[3] N. Fajar and F. F. Adiwijaya, “Sistem Otomatisasi Manajemen Akses Internet Menggunakan API Mikrotik di PT . Media Akses Data Automation System for Internet Access Management Using Mikrotik API at PT . Media Akses Data,” vol. 5, 2025.

[4] F. Sinlae, I. Maulana, F. Setiyansyah, and M. Ihsan, “Pengenalan Pemrograman Web: Pembuatan Aplikasi Web Sederhana Dengan PHP dan MYSQL,” *J. Siber Multi Disiplin*,

vol. 2, no. 2, pp. 68–82, 2024, doi: 10.38035/jsmd.v2i2.156.

[5] A. A. Simatupang, "Implementasi Restful Web Service Dengan Json WebToken Di Pt. Lestari Adil Makmur," *Semin. Nas. Mhs. Fak. Teknol. Inf.*, vol. 2, no. 2, pp. 2183–2192, 2023, [Online]. Available: https://senafti.budiluhur.ac.id/index.php/senafti/article/download/870/499/11067

[6] M. Melinda, S. R. R. Na, Y. Nurdin, and Y. Yunidar, "Implementation of System Development Life Cycle (SDLC) on IoT-Based Lending Locker Application," *J. RESTI*, vol. 7, no. 4, pp. 982–987, 2023, doi: 10.29207/resti.v7i4.5047.

[7] B. S. Nagara, D. Oetari, Z. Apriliani, and T. Sutabri, "Penerapan Metode Sdlc (System Development Life Cycle) Waterfall Pada Perancangan Aplikasi Belanja Online Berbasis Android Pada Cv Widi Agro Application of the Waterfall Sdlc (System Development Life Cycle) Method in Designing Android-Based Online Shopping," *J. Inf. Technol. Comput. Sci.*, vol. 6, no. 2, 2023.

[8] R. Ramadani and D. Mahdiana, "Systematic Literature Review on the Application of Ui/Ux Design Methods in System Development," *J. Tek. Inform.*, vol. 5, no. 4, pp. 103–111, 2024, doi: 10.52436/1.jutif.2024.5.4.2073.

[9] C. Perdana, Maharani, and M. Angga Wijaya, "Implementasi Framework Bootstrap 5 Pada Perancangan Front-End Website MC BRO di PT X," *J. Sist. Inf. Galuh*, vol. 2, no. 1, pp. 30–43, 2024, doi: 10.25157/jsig.v2i1.3634.

[10] M. Y. Putra, "Responsive Web Design Menggunakan Bootstrap Dalam Merancang Layout Website," *Inf. Syst. Educ. Prof. J. Inf. Syst.*, vol. 5, no. 1, pp. 61–70, 2020.

[11] Tiara Nurul Syahida, Nurul Hadianti, Yusromuin Munthe, Shara Jumiati Siregar, and Nurbaiti Sirait, "Analisis Penggunaan Database Dalam Meningkatkan Kualitas Sistem Informasi," *J. Sist. Inf. dan Ilmu Komput.*, vol. 1, no. 3, pp. 20–26, 2023, doi: 10.59581/jusiik-widyakarya.v1i3.729.

[12] A. R. Ismail, I. Labolo, and Y. Handayani, "SISTEMASI: Jurnal Sistem Informasi Implementasi JSON Parsing pada Aplikasi Pembelajaran Produktif Pertanian Implementation of JSON Parsing in Agricultural Productive Learning Applications," *Januari*, vol. 12, no. 1, pp. 269–281, 2023, [Online]. Available: http://sistemasi.ftik.unisi.ac.id

[13] I. Permatasari, F. Adhania, S. A. Putri, and S. R. C. Nursari, "Pengujian Black Box Menggunakan Metode Analisis Nilai Batas pada Aplikasi DANA," *KONSTELASI Konvergensi Teknol. dan Sist. Inf.*, vol. 3, no. 2, pp. 373–387, 2023, doi: 10.24002/konstelasi.v3i2.8289.

[14] N. A. Adha, A. Rofiq, and R. Basatha, "Implementasi CRUD (Create, Read, Update, Delete) pada Aplikasi Toko Sembako Berbasis Visual Basic.NET dan MySQL," *Al-DYAS*, vol. 4, no. 1, pp. 279–291, 2024, doi: 10.58578/aldyas.v4i1.4456.

[15] A. Hermanto, A. N. Utama, Y. Muflihah, A. Januantoro, and G. Kusnanto, "Pemanfaatan Metode Sinkron-Asinkron Komunikasi Data Dengan Flutter Sebagai Solusi Layanan Aplikasi Untuk Daerah Dengan Keterbatasan Sinyal Internet," *J. Sist. dan Teknol. Inf.*, vol. 12, no. 3, p. 405, 2024, doi: 10.26418/justin.v12i3.75164.

[16] meta, "Model Cards & Prompt formats Llama 4." Accessed: Sep. 10, 2025. [Online]. Available: https://www.llama.com/docs/model-cards-and-prompt-formats/llama4/

[17] D. B. Santoso and Y. Wahyuni, "Sestem Log Web Server Sebagai Pendeteksi Anomali Menggunakan Isolation Forest," *J. Apl. Bisnis dan Komput.*, vol. 4, no. 3, pp. 2807–5986, 2024, [Online]. Available: http://www.jubikom.unpak.ac.id