Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id

E - ISSN: 2723-7079 | P - ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May

OPEN ACCESS

https://doi.org/10.35870/jimik.v5i2.737

Optimisasi Pengecekan Anomali pada Proses Job: Analisis Waktu dan Data untuk Identifikasi Anomali yang Efisien

Michael Rio Aditya [1], Christine Dewi [2*]

[1,2*] Program Studi Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana, Kota Salatiga, Provinsi Jawa Tengah, Indonesia.

Email: michaelrio89@gmail.com [1], christine.dewi@uksw.edu [2*]

Histori Artikel:
Dikirim 30 Maret 2024; *Diterima dalam bentuk revisi* 19 April 2024; *Diterima* 1 Mei 2024; *Diterbitkan* 10 Mei 2024.

Abstrak

Penggunaan layanan dalam berbagai konteks program perangkat lunak telah menjadi hal yang vital. Anomali baik dalam bentuk skala kecil maupun besar dapat mengganggu fungsi suatu perangkat lunak dan berdampak pada proses bisnis perusahaan. Oleh karena itu, proses identifikasi anomali menjadi hal penting dalam memastikan kinerja suatu perangkat lunak optimal. Penelitian ini bertujuan untuk mengoptimalkan proses identifikasi anomali pada layanan perangkat lunak di PT. XYZ Tbk. Saat ini, informasi terjadinya anomali diperoleh melalui pemeriksaan manual oleh tim sistem operator dengan melakukan pengecekan pada job secara satu persatu yang membutuhkan waktu lama dan berpotensi menyebabkan beberapa job anomali yang terabaikan. Penelitian ini menyajikan sebuah solusi untuk mengatasi masalah ini dengan membuat sistem optimasi pemantau identifikasi anomali pada job untuk identifikasi yang lebih cepat sehingga layanan sistem tetap terjaga stabilitasnya. Sistem optimasi pemantau identifikasi anomali merupakan aplikasi berbasis website yang dibangun menggunakan metodologi Prototyping, bahasa pemrograman PHP, framework Yii2, basis data PostgreSQL, WebSocket, dan API Control-M. Hasil dari penelitian ini adalah sistem optimasi pemantau identifikasi anomali yang dibuat mampu mempercepat proses identfikasi dan mempercepat proses transfer antar informasi dengan WebSocket.

Kata Kunci: Sistem Pemantau; Identifikasi; Anomali Layanan Perangkat Lunak; Websocket.

Abstract

The use of services in various software program contexts has become vital. Anomalies, whether in small or large scale, can disrupt the function of a software and impact a company's business processes. Therefore, the process of anomaly identification is crucial to ensure optimal software performance. This research aims to optimize the anomaly identification process in software services at PT. XYZ Tbk. Currently, information about anomalies is obtained through manual checks by the system operator team, which involves checking each job one by one, consuming a lot of time and potentially leading to overlooked anomalies. This research presents a solution to address this issue by developing an optimization system for monitoring anomaly identification in job, enabling faster identification for maintaining system services stability.. The anomaly identification monitoring system is a web-based application built using the Prototyping methodology, PHP programming language, Yii2 framework, PostgreSQL database, WebSocket, and Control-M API. The result of this research is the developed anomaly identification monitoring system, capable of accelerate the identification process and accelerate information transfer using WebSocket.

Keyword: Monitoring System; Identification; Software Service Anomalies; WebSocket.

Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id
E-ISSN: 2723-7079 | P-ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May
OPEN ACCESS
https://doi.org/10.35870/jimik.v5i2.737

1. Pendahuluan

PT. XYZ Tbk merupakan perusahaan yang bergerak di bidang layanan perbankan. Penggunaan layanan *software* menjadi suatu hal yang penting dalam mendukung proses bisnis pada PT. XYZ Tbk. Kelancaran *software* dalam menjalankan layanan nya menjadi sangat penting. Adanya kegagalan atau penyimpangan sekecil apapun dalam software menjalankan layanan nya akan memberikan dampak pada proses bisnis PT. XYZ Tbk terganggu. Kegagalan atau penyimpangan ini sering disebut dengan anomali dalam perusahaan. Deteksi anomali, juga dikenal sebagai deteksi *outlier*, merujuk pada proses mendeteksi data yang secara signifikan menyimpang dari mayoritas data (Pang *et al.*, 2021). Deteksi anomali memiliki beragam implementasi dalam berbagai bidang. Sebagai contoh, deteksi anomali telah banyak digunakan untuk menemukan intrusi jaringan. Selain itu, deteksi anomali juga telah digunakan dalam berbagai implementasi lain seperti mengidentifikasi praktik medis yang salah atau penipuan kartu kredit (Taha & Hadi, 2019). Pada penelitian ini akan membahas tentang identifikasi anomali yang terjadi pada layanan *software* yang digunakan oleh PT. XYZ Tbk. Anomali *software* merupakan suatu keganjilan, keanehan atau penyimpangan dari kondisi mayoritas yang terjadi pada program yang digunakan untuk mengoperasikan layanan pada *software*. Anomali pada layanan *software* dapat mencakup berbagai bentuk, mulai dari *bug* kecil hingga masalah yang lebih serius yang dapat menghambat fungsi utama suatu layanan *software*. Oleh karena itu, proses identifikasi anomali menjadi langkah penting dalam memastikan kinerja *software* optimal. Efisiensi dalam proses identifikasi anomali merupakan suatu hal yang dibutuhkan agar tindak lanjut terhadap job yang mengalami anomali terhadap *Person In Charge* (PIC) dapat berjalan lebih cepat. Kejelasan dan efisiensi proses ini akan berdampak langsung pada kelancaran operasional perusahaan.

Batch processing merupakan sebuah proses beruntun otomatis yang dijalankan oleh *software*. *Batch processing* digunakan dalam perusahaan untuk mengeksekusi sebuah *task* seperti penghitungan kalkulasi keuangan, penurunan gaji, dan lain–lain. *Batch processing* ini disebut dengan *job*. *Job* berada pada mainframe dari *software Control-M* yang proses nya dapat terlihat melalui sebuah API. Job memiliki beberapa elemen untuk menjalankan fungsi nya sebagai *batch processing*, yakni proses apa yang harus dijalankan dan kapan proses dijalankan (Ding, 2012). *Job* dapat berjalan kapanpun sesuai dengan kondisi *estimated start time* saat job tersebut pertama kali dibuat. Pada kondisi tertentu *job mainframe* dapat berjalan tidak sesuai kondisi seharusnya, dalam arti *job* yang seharusnya berjalan sesuai *estimated start time* dapat mengalami anomali dengan kondisi *job* tersebut belum berjalan (*wait condition*) meski sudah melewati batas waktu *estimated start time*. Kondisi saat ini pada PT. XYZ Tbk, informasi terjadinya job anomali merupakan hasil pemeriksaan manual yang dilakukan oleh tim sistem operator. Proses penelusuran *job* secara manual untuk memastikan tidak adanya *job* yang anomali membutuhkan waktu yang lama untuk sekali sesi pengecekan. Hal ini disebabkan karena tim sistem operator melakukan pengecekan secara manual terhadap daftar *job mainframe* pada *software Control-M*, dalam arti bahwa tim sistem operator melakukan identifikasi terhadap kumpulan *job mainframe* secara satu-persatu. Hal ini menyebabkan proses tindak lanjut dalam penanganan *job* yang mengalami anomali kepada *Person In Charge* (PIC) menjadi lebih lama karena proses menunggu dari selesainya hasil penelusuran *job*. Beberapa pembahasan mengenai identifikasi anomali telah dilakukan, pada penelitian yang berjudul "Deteksi Dini Identifikasi Insiden Pada Kejadian Anomali Perangkat Lunak Dengan Sistem Pendeteksi Anomali Perangkat Lunak Studi Kasus Di Astra Life" telah melakukan perancangan sistem pendeteksi anomali perangkat lunak bernama SIMPEL.

Pada penelitian tersebut sistem yang dirancang mampu memangkas waktu identifikasi adanya job anomali sebesar 53% dari waktu identifikasi *job* anomali sebelum adanya sistem tersebut dibuat (Sasmito Budi Utomo, Mela Hidayah, 2022). Selain itu pada penelitian yang berjudul "*A Semisupervised Autoencoder-based Approach for Anomaly Detection in High Performance Computing Systems*" telah menganalisis penggunaan metode autoencoder-based pada kinerja komputasi komputer tingkat tinggi dan mendapatkan tingkat akurasi deteksi terhadap anomali sebesar 12% (Borghesi et al., 2019). Kemudian pada penelitian yang berjudul "*A Real-Time Detection Method of Software Configuration Errors Based on Fine-Grained Configuration Item Types*" telah mengembangkan sebuah metode untuk mendeteksi adanya

Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id
E-ISSN: 2723-7079 | P-ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May
OPEN ACCESS
https://doi.org/10.35870/jimik.v5i2.737

anomali pada *syntax* yang ada pada sistem serta membuat konfigurasi dengan pendekatan rule base untuk memonitor konfigurasi file pada perangkat lunak dengan *real-time*. Penelitian tersebut menghasilkan tingkat akurasi klasifikasi dan efektivitas deteksi mencapai 87% dan 82% (Zhang *et al.*, 2022). Selanjutnya pada penelitian yang berjudul "*Software Error Detection And Correction* (SEDC) *Using Layer Based Approach In Expert System*" telah melakukan perancangan sistem untuk mendeteksi kesalahan dan koreksi dalam aplikasi, dengan sistem yang dibuat kesalahan yang pernah terjadi dijadikan sebagai referensi dalam mendeteksi kesalahan baru sehingga rasio kesalahan dapat lebih rendah (Josephine & Subramanian, 2010). Lalu pada penelitian yang berjudul "*A Pattern-Based Framework for Software Anomaly Detection*" telah melakukan perancangan sistem menggunakan *pattern-based framework* untuk mendeteksi anomali pada perangkat lunak. Dengan penerapan sistem menggunakan *framework* tersebut, penulis dapat mendeteksi ratusan pattern yang mengalami anomali yang jika dilakukan dengan pendekatan konvensional akan mustahil (Kothari et al., 2004). Selanjutnya pada penelitian yang berjudul "*Software Operation Anomalies Diagnosis Method Based on a Multiple Time Windows Mixed Model*" telah mengembangkan metode berbasis *multiple time windows mixed* model yang digunakan untuk memperkuat akurasi dalam mendeteksi anomali pada perangkat lunak dibandingkan dengan pendekatan tradisional (Shi *et al.*, 2023). Kemudian pada penelitian yang berjudul "*Software Errors Analysis and Detection:A Survey*" mengemukakan penting nya analisis *error* dalam perangkat lunak untuk mendeteksi anomali yang terjadi sehingga dapat mengurangi beban biaya dalam *software development lifecycle* (SDLC)(Lazić, 2006). Dan pada penelitian yang berjudul "*Recent Progress of Anomaly Detection*" memberikan tinjauan terbaru tentang kemajuan yang telah dicapai dalam deteksi anomali dengan fokus pada tipe-tipe data yang berdimensi tinggi dan data campuran (Xu *et al.*, 2019).

Selain permasalahan tersebut, pengguna yakni sistem operator membutuhkan sebuah fitur *real-time* untuk melakukan aksi *acknowledge* terhadap daftar job anomali yang tertampil hasil dari identifikasi. *Acknowledge* merupakan fitur untuk memberikan pesan bahwa data job dapat diabaikan sehingga data job tersebut dapat dipindahkan dari tampilan *monitoring* sistem tabel *alert*. Dengan adanya beberapa anggota sistem operator yang memantau identifikasi anomali, tentunya dibutuhkan proses transfer informasi antar anggota dengan cepat. Ajax merupakan kependekan dari Asynchronous Javascript and XML. XMLHttpRequest (XHR) adalah API yang dapat digunakan oleh *Javascript* dan bahasa skrip web lainnya untuk mentransfer dan memanipulasi data XML dari *server web* untuk ke *server web* menggunakan HTTP, menciptakan saluran koneksi independen antara sisi klien dan sisi server pada halaman *web* (Gupta *et al.*, 2020). Namun penggunaan ajax dalam sistem berbasis *web* memiliki kelemahan seperti kendala validasi data yang mengandalkan penyegaran halaman ketika pengguna membutuhkan data pada *server*, segala aktivitas pengguna menjadi terhenti hingga *server* memberikan tanggapan dan *browser* melakukan penyegaran halaman (Kadir, 2009). Dengan kondisi kasus PT. XYZ Tbk yang membutuhkan proses transfer data secara *real-time* untuk memantau identifikasi, penulis mengajukan *WebSocket* sebagai metode transfer data secara *real-time*. WebSocket menawarkan hasil yang lebih baik dibandingkan dengan pendekatan konvensional yang dianggap sebagai solusi dalam menyediakan informasi secara *real-time* serta menawarkan komunikasi yang cepat dan *stateful* antara server dan pengguna web (Ogundeyi & Yinka-Banjo, 2019). Secara dasar, dengan menggunakan *WebSocket*, pengguna dapat menerima data dari server tanpa perlu melakukan permintaan data terlebih dahulu, berbeda dengan Ajax yang hanya dapat menerima data saat pengguna sudah melakukan permintaan kepada server terlebih dahulu (Maulana, 2021). Dalam hal ini metode *WebSocket* lebih cocok diaplikasikan pada proses transfer data dari aksi *acknowledge* sehingga informasi yang didapat oleh masing-masing anggota tim sistem operator merupakan informasi secara *real-time*. Berdasarkan latar belakang yang telah dipaparkan, penelitian ini bertujuan untuk membangun sistem optimasi pemantau identifikasi job yang mengalami anomali untuk meningkatkan efisiensi identifikasi dan dengan implementasi WebSocket pada sistem optimasi untuk proses transfer data secara *real-time* sehingga proses identifikasi job yang mengalami anomali pada PT. XYZ Tbk dapat menjadi lebih cepat.

Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id
E-ISSN: 2723-7079 | P-ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May
OPEN ACCESS
https://doi.org/10.35870/jimik.v5i2.737

2. Metode Penelitian

Model *prototype* merupakan teknik untuk mengumpulkan informasi berdasarkan kebutuhan pengguna dengan cepat, dan aplikasi yang disajikan akan berfokus pada pengguna (Suryn, 2014). Dengan menggunakan metode pengembangan *prototype*, pengguna dapat memiliki pemahaman yang lebih baik tentang sistem yang sedang dikembangkan walaupun pengguna tidak sepenuhnya memiliki penguasaan terhadap teknologi yang diimplementasikan, pengguna dapat memberikan umpan balik secara langsung berdasarkan *prototype* dari sistem yang telah dibuat sehingga pada hasil akhir adalah pengujian sistem yang telah sesuai dengan kebutuhan pengguna. Untuk mengantisipasi adanya perubahan yang signifikan, penggunaan metode prototype dinilai cocok untuk pengambangan sistem identifikasi anomali. Selain itu, metode prototype memiliki kesesuaian penerapan dalam lingkungan kantor. Metode *prototype* memiliki 5 tahapan pengembangan yaitu *communication*, *quick plan*, *modelling quick design*, *construction*, dan *deployment, delivery & feedback*.

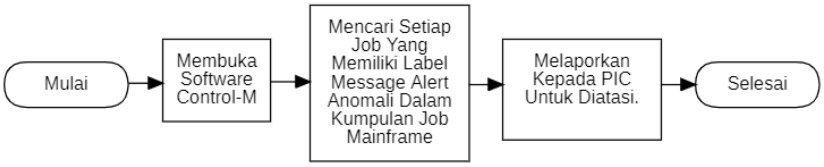
Gambar 1. Metode *Prototype*

2.1 *Communication*

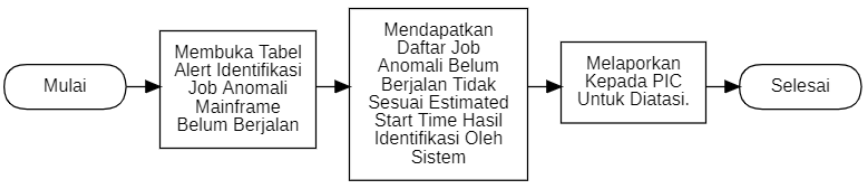
Pada tahap *communication* penulis mengumpulkan segala kebutuhan yang diperlukan untuk merancang sistem yang dibuat dengan melakukan diskusi dengan tim sistem operator. Pada proses diskusi yang dilakukan, penulis mengumpulkan layanan apa saja yang dibutuhkan seperti API dari *Control-M* untuk memberikan akses secara langsung kepada sistem yang dibuat terhadap proses *job* yang ada pada API. Penggunaan *WebSocket* dinilai cocok untuk digunakan pada sistem yang dibuat untuk memenuhi kebutuhan pengguna dalam proses pemantauan *job* secara *real-time*.

2.2 *Quick Plan*

Pada tahap *quick plan* dilakukan dengan perancangan sistem secara garis besar. Perancangan dilakukan dengan cepat dan mewakili seluruh aspek sistem.



Gambar 2. Proses Identifikasi *Job Mainframe* belum berjalan sebelum adanya sistem

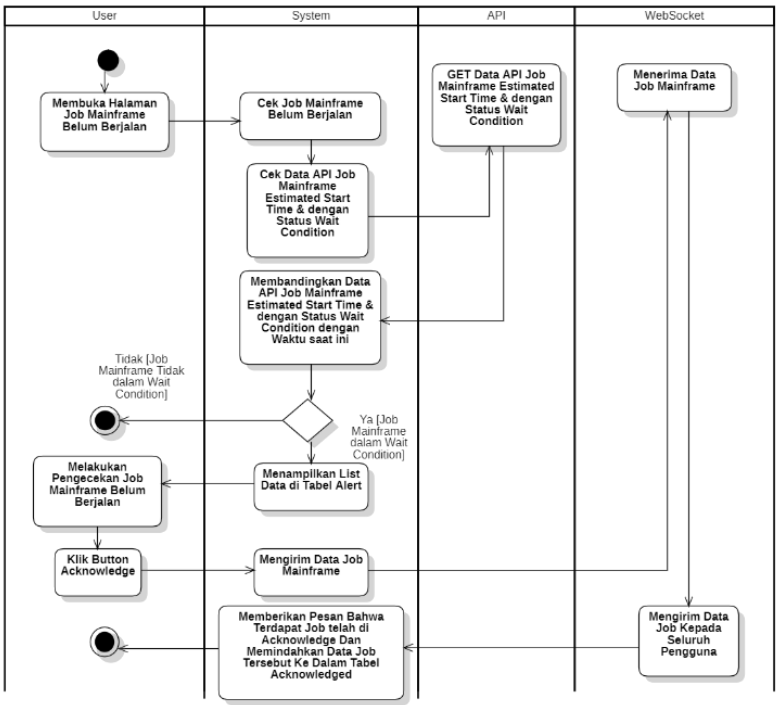


Gambar 3. Proses Identifikasi *Job Mainframe* belum berjalan sesudah adanya sistem

Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id
E - ISSN: 2723-7079 | P - ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May
OPEN ACCESS
https://doi.org/10.35870/jimik.v5i2.737

2.3 *Modelling Quick Design*

Pada tahap *modelling quick design* berfokus pada penyajian hasil dari *quick plan* dengan pembuatan alur kerja sistem yang dibuat.



Gambar 4. *Activity Diagram* Identifikasi *Job Mainframe* belum berjalan

2.4 *Construction*

Pada tahap *construction* dilakukan pengkodean terhadap *prototype* yang telah dibuat oleh penulis dan disetujui oleh pengguna yakni tim sistem operator. Proses pengkodean dilakukan dengan pengerjaan durasi selama 1 bulan hingga sistem optimasi pemantau identifikasi anomali dapat berjalan sesuai dengan rancangan *prototype*.

2.5 *Deployment, Delivery & Feedback*

Pada tahap *deployment, delivery & feedback* sistem optimasi pemantau idenfitikasi anomali yang telah dibuat dilakukan pengujian untuk menguji fungsionalitas dari sistem yang dibuat. Pengujian dilakukan dari 2 sisi, yakni pengujian fungsionalitas menggunakan metode *blackbox* dan pengujian perbandingan waktu antara proses mengirim dan menerima data. Setelah pengujian dilakukan, sistem akan dikirimkan kepada pengguna sehingga pengguna dapat memberikan umpan balik kepada sistem dalam memberikan evaluasi jika diperlukan untuk *maintenance*.

3. Hasil dan Pembahasan

3.1 Perancangan Sistem

Penelitian ini menciptakan aplikasi sistem optimasi pemantau identifikasi anomali dengan implementasi *WebSocket* dan bahasa pemrograman PHP menggunakan layanan API dari *Control-M*. Penulis menyiapkan link API untuk akses data *job* kemudian ditampung masuk kedalam sistem. Dengan ini sistem terhubung dengan data *job*. Identifikasi dilakukan dengan mengolah data *job* pada API.

Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id
E-ISSN: 2723-7079 | P-ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May
OPEN ACCESS
https://doi.org/10.35870/jimik.v5i2.737

Tabel 1. Data *Job* API *Control-M*

Data	Tipe Data	Keterangan
Jobname	*String*	Nama *job*
Status	*String*	Status *job* dapat berupa *wait condition* atau *executing*
Estimated Start Time	*Timestamp*	Waktu *job* akan berjalan

Pada tabel 1 merupakan data job yang diambil pada API untuk digunakan sistem dengan kondisi job yang memiliki *estimated start time* dan status *wait condition*. Properti *estimated start time* yang berbentuk *timestamp* digunakan untuk analisis perbandingan dengan waktu saat ini. Data-data tersebut akan digunakan untuk mengidentifikasi proses *job* yang mengalami anomali. Setelah mendapatkan data *job* dengan *estimated start time* dan status *wait condition*, selanjutnya dilakukan analisis perbandingan dengan waktu saat ini (*current time*) dengan ketentuan jika data job dengan *estimated start time* dan status *wait condition* lebih kecil dari waktu saat ini, maka *job* teridentifikasi sebagai anomali.

Tabel 2. Tabel Database *Acknowledged* Data

Kolom	Tipe Data	Keterangan
Jobname	*String*	Nama *job*
Status	*String*	Status *job* dapat berupa *wait condition* atau *executing*
Estimated Start Time	*Timestamp*	Waktu *job* akan berjalan
Message	*String*	Keterangan *job acknowledged*

Pada tabel 2 merupakan tabel *database* untuk menampung data *job* yang telah di *acknowledge* oleh pengguna. Pada kondisi tertentu dalam arti bahwa job dapat memiliki kondisi normal pada proses nya bagi tim sistem operator. *Job* tersebut dapat diabaikan dengan memindahkan data *job* dari tampilan *monitoring* sistem tabel *alert* kedalam tabel *acknowledged* data. Dengan ini, tim sistem operator dapat berfokus pada data *job* yang memang mengalami anomali.

Kode Program 1. Identifikasi *Job* belum berjalan

```
//API
$endpoint = 'https://10.40.3.148:8443/automation-api';
//GET JOB HEADER
$url = $endpoint.'/run/jobs/status?limit=10000&status=Wait%20Condition';
//GET JOB DATA
$curl_data = curl_init();
curl_setopt($curl_login, CURLOPT_URL, $url);
$result2 = json_decode(curl_exec($curl_data));
$finalData = [];
    foreach ($result2->statuses as $value){
        if (isset($value->estimatedStartTime)){
            foreach ($value->estimatedStartTime as $estimatedstart) {
                // GET CURRENT TIME
                $get_current_time = date('Y-m-d H:i:s', time());
                $current_of_time = new DateTime($get_current_time);
                // GET ESTIMATE START TIME
                $get_estimate_start_time = date('Y-m-d H:i:s',
strtotime($estimatedstart));
                $estimate_start_time = new DateTime($get_estimate_start_time);
                // GET JOB NAME
                $thejobname = $value->name;
                // FILTER STATUS
                $thestatus = $value->status;
                // LABEL MESSAGE
                $label = 'Anomaly detected';
                // CEK DATA DB
                $is_acknowledged = false;
                $ackstatus_job = ApiAcknowledge::find()->where
                (["api_ack_status" => "acknowledged", "api_job_id" => $thejobid]);
                    if($ackstatus_job->exists()){
                        $is_acknowledged = true;
```

Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id
E-ISSN: 2723-7079 | P-ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May
OPEN ACCESS
https://doi.org/10.35870/jimik.v5i2.737

```
                    }
                // IDENTIFICATION PROCESS
if ($is_acknowledged != true && $estimate_start_time->format('Y-m-d H:i:s') <
$current_of_time->format('Y-m-d H:i:s')) {
                        $rowData = array(
                            'jobname' => $thejobname,
                            'status' => $thestatus,
                            'estimated_start_time' => $ estimate_start_time->format('Y-
m-d H:i:s'),
                            'current_time' => $current_of_time->format('Y-m-d H:i:s'),
                            'label' => $label
                        );
                        $finalData[] = $rowData;
                    }
                    break;
                }
            }
        }
```

Kode program 1 merupakan fungsi yang digunakan untuk identifikasi anomali terhadap job mainframe belum berjalan. Pada baris 1-2 menentukan link API yang digunakan. Baris 3-4 menggunakan URL untuk mengambil data *job* berdasarkan kondisi *wait condition*. Baris 5-8 mengambil data job berdasarkan link dalam variabel $url menggunakan cURL, kemudian data ditampung dalam variabel $result2. Pada baris 9-47 mengolah data pada $result2. Pada baris 9-12 melakukan pengecekan kondisi apakah data *job* memiliki *estimated start time*, ketika kondisi terpenuhi maka pengolahan dilakukan pada baris 13-31 agar data dapat digunakan pada proses identifikasi pada baris 32-47. Selanjutnya, baris 32-47 proses identifikasi dilakukan dengan analisis perbandingan jika $estimated_start_time yang berarti perkiraan waktu saat *job* berjalan lebih kecil dari waktu saat ini $current_of_time maka job dianggap telah berlalu namun masih dengan kondisi *wait condition*. Dengan ini data *job* dapat teridentifikasi sebagai anomali. $finalData merupakan data hasil identifikasi terhadap *job mainframe* belum berjalan yang kemudian digunakan untuk ditampilkan pada tabel *alert*.

Kode Program 2. *Insert* Data Pada Database *Acknowledged* Data

```
function ackFunc(event) {
    var rowData = $('#message-dialog').data('ackData');
    var message = $('#message-input').val();
    $.ajax({
        url: '".Url::to(['report/insertacknowledge']).."',
        type: 'POST',
        data: {
            rowData: rowData,
            message: message,
        }
    });
}
```

Kode program 2 merupakan fungsi yang digunakan untuk melakukan *insert* data terhadap job yang telah di *acknowledge*. Pada baris 2 dan 3 variabel rowData dan variabel message berisi baris data yang terpilih untuk di *acknowledge* beserta pesan yang di masukkan oleh pengguna. Pada baris 4-11 digunakan *ajax* untuk mengirim data kepada *controller* insertacknowledge yang terdapat pada variabel url. Dengan function ackFunc dijalankan, maka data akan masuk kedalam database melalui url dengan perintah POST.

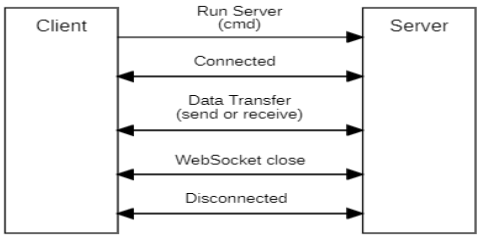
Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id
E-ISSN: 2723-7079 | P-ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May
OPEN ACCESS
https://doi.org/10.35870/jimik.v5i2.737

Name	Status	Estimated Start Time	Current Time	Label	Acknowledge
CIDATLDI	Wait Condition	2024-02-20 19:03:10	2024-02-20 20:06:58	Anomaly detected	Acknowledge
DBPFILGF	Wait Condition	2024-02-18 14:05:00	2024-02-20 20:06:58	Anomaly detected	Acknowledge
CDCURLOV	Wait Condition	2024-02-20 16:30:00	2024-02-20 20:06:58	Anomaly detected	Acknowledge
CDCVDSPS	Wait Condition	2024-02-18 13:00:30	2024-02-20 20:06:58	Anomaly detected	Acknowledge
CDIPSC	Wait Condition	2024-02-19 18:00:00	2024-02-20 20:06:58	Anomaly detected	Acknowledge

Gambar 5. Hasil Identifikasi Anomali Job Mainframe Belum Berjalan pada Tabel Alert

Gambar 5 merupakan 5 sampel job yang berhasil teridentifikasi mengalami anomali. Berdasarkan contoh nama job CIDATLDI memiliki waktu perkiraan job mulai berjalan (Estimated Start Time) pada tanggal 2024-02-20 dengan waktu pukul 19:03:10, namun berdasarkan waktu saat sampel diambil yakni pada kolom Current_Time, tanggal menunjukan 2024-02-20 dengan waktu pukul 20:06:58 yang dapat diartikan bahwa job tersebut belum berjalan. *Job* masih dalam status *wait condition* yang dimana seharusnya job tersebut telah berjalan dengan status *executing* pada tanggal 2024-02-20 pukul 19:03:10 sesuai dengan waktu perkiraan job berjalan pada kolom *Estimated Start Time*. Sistem dapat mendeteksi atau mengidentifikasi *job mainframe* yang mengalami anomali dalam kondisi belum berjalan tidak sesuai *estimated start time*. Hal ini merupakan kebutuhan bisnis yang diperlukan oleh PT. XYZ Tbk dalam memantau adanya proses *job* yang anomali, dengan ini tim sistem operator tidak perlu melakukan identifikasi terhadap *job* anomali secara manual karena telah teridentifikasi oleh sistem berdasarkan proses pada kode program 1.

3.2 Implementasi *WebSocket*



Gambar 6. Proses Kerja *WebSocket*

Proses kerja *WebSocket* dimulai dengan menjalankan *server WebSocket* pada sisi *server*. Setelah server dijalankan maka koneksi antara sisi *client* dan server akan terhubung, proses ini disebut dengan *handshake*. Koneksi ini akan terus terhubung hingga pengguna menutup halaman sistem. Selanjutnya pengguna dapat melakukan aksi *acknowledge* terhadap data job dengan mengirimkan data melalui *button acknowledge*. Data dikirimkan dalam bentuk JSON menggunakan *javascript* pada *WebSocket*. Kemudian pada sisi server akan menerima data secara cepat dan memberi respon pada seluruh pengguna yang terhubung dengan melakukan *update* data pada tabel *alert* secara otomatis melalui *WebSocket*. Dengan ini pengguna dapat memperoleh *update* terhadap data pada sisi server tanpa harus melakukan permintaan pada sisi *server* terlebih dahulu. Berbeda dengan pendekatan konvensional dimana perlu membangun koneksi berbeda untuk melayani *request* secara terpisah. Setelah *request* selesai, koneksi akan terputus secara otomatis sehingga perlu melakukan *request* berulang kali pada sisi *server*, tentunya hal ini dapat membuat performa sistem menurun karena melayani banyaknya *request* tersebut. Implementasi *WebSocket* dilakukan dengan beberapa tahap. Instalasi WebSocket dilakukan melalui *composer*. Setelah *WebSocket* terinstal maka *library* yang telah terinstal dapat digunakan untuk menjalankan koneksi yang menghubungkan sisi server dan sisi *client*.

Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id
E-ISSN: 2723-7079 | P-ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May
OPEN ACCESS
https://doi.org/10.35870/jimik.v5i2.737

Kode Program 3. Kode program untuk implementasi *WebSocket* pada sisi *server*

```
use Ratchet\Server\IoServer;
use MyApp\Socket;
use Ratchet\Http\HttpServer;
use Ratchet\WebSocket\WsServer;
require dirname(__DIR__) . '/vendor/autoload.php';
$server = IoServer::factory(
            new HttpServer(
                new WsServer(
                    new Socket()
                )
            ),
            8080
            );
$server->run();
```

Kode program 3 merupakan *file run_socket_server*. PHP yang digunakan untuk menjalankan koneksi pada sisi server. Beberapa *library* seperti IoServer, HttpServer, dan WsServer digunakan untuk menjalankan koneksi dari *instance Socket* pada sisi *server* yang telah dibuat. *Instance* Socket merupakan *class* file yang berfungsi untuk menerima dan mengelola koneksi. Server kemudian dijalankan dengan memanggil *method* run melalui perintah *run_socket_server*. PHP pada *command prompt* untuk memulai *broadcast* server WebSocket pada *port* 8080. Dengan ini maka beberapa pengguna dapat terhubung melalui *broadcast* server yang telah berjalan dan dapat mengirim dan menerima data melalui sisi *client*.

Kode Program 4. Implementasi *WebSocket* mengirim data pada sisi *client*

```
// send data to websocket
var conn = new WebSocket('ws://localhost:8080/websocket/');
var rowData = $('#message-dialog').data('ackData');
var dataToSend = {
    title: 'JOB ACKNOWLEDGED',
    msg: rowData.jobname + ' ' + rowData.status
};
conn.send(JSON.stringify(dataToSend));
```

Kode program 4 merupakan *script* untuk melakukan pengiriman data pada WebSocket. Saat pengguna menjalankan function ackFunc pada kode program 2, maka kode program 4 akan berjalan secara bersamaan dengan mengirim data yang dikirimkan melalui function tersebut pada WebSocket. Pada baris 2 merupakan inisiasi permintaan koneksi untuk terhubung dengan server yang telah berjalan melalui *composer* sebelumnya. Koneksi akan terhubung sesuai dengan port yang telah berjalan pada *server* yakni localhost 8080. Pada baris 3-8 digunakan untuk melakukan pengiriman data yang pengguna ingin kirimkan kepada seluruh pengguna yang terhubung. Variabel rowData berisi data dari hasil tampilan daftar job yang teridentifikasi anomali pada tabel *alert* yang dikirimkan oleh pengguna melalui aksi *acknowledge*. Variabel dataToSend berisi data yang akan dikirimkan pada *WebSocket* yang memiliki *key* title dan *key* msg, msg merupakan nama job dan status job yang telah di *acknowledge* oleh pengguna. Pada baris 8 digunakan untuk mengirim data dari variabel dataToSend dalam bentuk JSON melalui *stringify* agar dapat diakses oleh *WebSocket*.

Kode Program 5. Implementasi *WebSocket* menerima data pada sisi *client*

```
// receive data from websocket
    conn.onmessage = function(e) {
        var data = JSON.parse(e.data);
var dataname = data.title;
        var dataresult = $.ajax({
            success: function(data) {
                if(dataname === 'JOB ACKNOWLEDGED') {
                    tables.draw();
                }
            }
        });
    };
```

Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id
E-ISSN: 2723-7079 | P-ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May
OPEN ACCESS
https://doi.org/10.35870/jimik.v5i2.737

Kode program 5 merupakan *scipt* untuk menerima data yang telah dikirimkan pada *WebSocket*. Pada baris 3 variabel data berisi data yang telah diterima pada *WebSocket*. Untuk mengakses data dilakukan JSON *parse* agar data dapat diakses. Pada baris 4 variabel dataname berisi nama pesan ketika suatu data job telah di *acknowledge*. Selanjutnya pada baris 5-12, dilakukan pemeriksaan jika terdapat data yang memiliki pesan "*JOB ACKNOWLEDGED*" maka *WebSocket* akan melakukan *update* data dengan tables.draw() pada tampilan tabel *alert* kepada seluruh pengguna yang terhubung pada server secara otomatis dan cepat tanpa pengguna harus melakukan penyegaran halaman, dengan ini data yang telah di *acknowledge* akan hilang dari tampilan tabel *alert* untuk dipindahkan kedalam tabel *acknowledged* data.

3.3 Pengujian Sistem

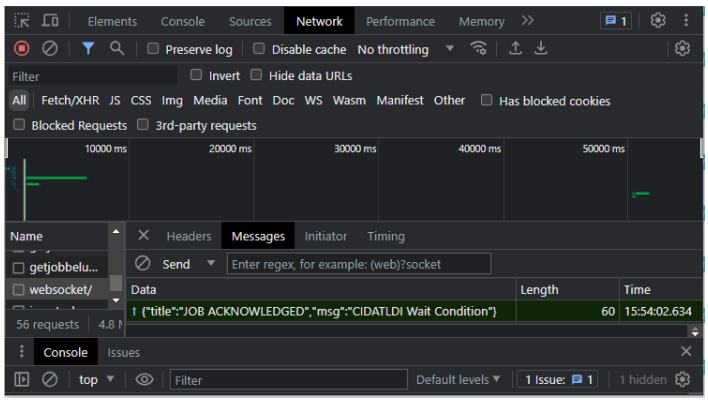
Beberapa pengujian dilakukan untuk menguji sistem agar berjalan dengan seharusnya sesuai dengan prototype yang telah dibuat berdasarkan kebutuhan pada tahap awal perancangan sistem. Pengujian dilakukan menggunakan metode *blackbox* dan pengujian performa dengan fokus *latency testing*. Dengan pengujian *blackbox* merupakan pengujian yang berfokus pada fungsional sedangkan *latency testing* berfokus pada kecepatan waktu yang diperlukan saat *WebSocket* mentransmisikan data.

Tabel 3. Pengujian Metode *Blackbox*

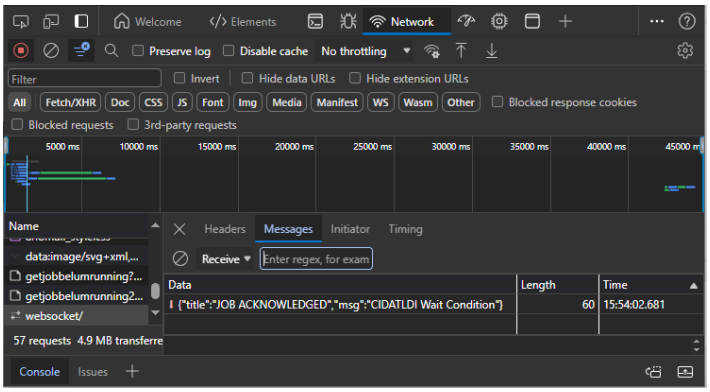
No	Pengujian	Hasil	Kesimpulan
1	Sistem dapat memproses data pada API Control-M	Data API dapat diproses dengan kondisi yang sesuai	*Valid*
2	Sistem dapat mengidentifikasi data job yang belum berjalan tidak sesuai *estimated start time*	Data yang ditampilkan merupakan data job belum berjalan tidak sesuai *estimated start time*	*Valid*
3	Tombol *acknowledge* dapat memindahkan data job dari tabel *alert* kedalam tabel *acknowledged* data	Row data yang terpilih hilang dari tabel *alert* dan masuk ke tabel *acknowledged* data	*Valid*
4	Sistem dapat mengirim dan menerima data pada WebSocket	Saat *acknowledge* dijalankan WebSocket dapat mengirim dan menerima data pengguna dan melakukan *update* pada tabel *alert* kepada seluruh pengguna	*Valid*

Tabel 3 merupakan hasil dari pengujian *blackbox*, dari hasil pengujian menunjukan bahwa seluruh kebutuhan sistem dapat memenuhi standar yang telah ditetapkan pada tahap *prototype*. Pada pengujian nomor 2 sistem mampu mengidentifikasi data yang diperoleh dan ditampilkan pada tabel *alert* dengan kondisi data job yang memang mengalami anomali pada API, dengan ketentuan kondisi data job merupakan data yang memiliki status *wait condition* belum berjalan tidak sesuai *estimated start time*, hal ini menunjukan keberhasilan sistem dalam mengidentifikasi job pada mainframe yang mengalami anomali secara otomatis. Dengan ini proses identifikasi tidak perlu dilakukan secara manual oleh tim sistem operator karena job anomali telah teridentifikasi oleh sistem yang dibuat. Selanjutnya, dari hasil pengujian menunjukan bahwa saat aksi *acknowledge* dijalankan *WebSocket* mampu mengirim dan menerima data sehingga dapat melakukan *update* melalui penyegaran data tabel kepada seluruh pengguna sekaligus.

Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id
E-ISSN: 2723-7079 | P-ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May
OPEN ACCESS
https://doi.org/10.35870/jimik.v5i2.737



Gambar 7. Pengujian *Latency Testing* Penggunaan *WebSocket* Pada Sistem Mengirim Data Dengan Chrome DevTools



Gambar 8. Pengujian *Latency Testing* Penggunaan *WebSocket* Pada Sistem Menerima Data Dengan Microsoft Edge DevTools

Tabel 4. Pengujian *Latency Testing* Sistem Mengirim Data Melalui WebSocket

Objek	Data	Length	Time (Milidetik)
Pengguna 1 mengirim data menggunakan *broswer* Chrome	*Title: Job ACKNOWLEDGED Message: CIDATLDI Wait Condition*	60	634

Tabel 5. Pengujian *Latency Testing* Sistem Menerima Data Melalui WebSocket

Objek	Data	Length	Time (Milidetik)
Pengguna 2 menerima data menggunakan *broswer Microsoft Edge*	*Title: JOB ACKNOWLEDGED Message: CIDATLDI Wait Condition*	60	681

Tabel 4 dan 5 merupakan hasil dari pengujian performa sistem dalam kecepatan waktu yang dibutuhkan saat halaman mengirim dan menerima data melalui *WebSocket*. Berdasarkan pengujian yang dilakukan pada gambar 7 melalui alat Chrome DevTools, data dikirimkan oleh pengguna 1 pada waktu pukul 15:54:02.634 yang tertera pada kolom *Time*. Selanjutnya berdasarkan pada gambar 8 melalui alat Microsoft Edge DevTools, pengguna 2 menerima data yang diperoleh dari hasil kiriman pengguna 1 pada waktu pukul 15:54:02.681. Berdasarkan hasil pengujian tersebut hasil analisis adalah

Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id
E-ISSN: 2723-7079 | P-ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May
OPEN ACCESS
https://doi.org/10.35870/jimik.v5i2.737

objek pengguna 1 pada tabel 4 mengirim data dengan satuan milidetik sebesar 634 dan objek pengguna 2 pada tabel 5 menerima data yang telah dikirimkan dengan satuan milidetik sebesar 681. Dengan hasil yang diperoleh tersebut selisih perbedaan antara pengguna 1 dalam mengirim data dan pengguna 2 menerima data hanya memiliki perbedaan sebesar 47 milidetik. Dengan ini dapat disimpulkan bahwa perbedaan waktu yang diperlukan saat WebSocket mentransmisikan data antar pengguna tercatat dibawah 1 detik. Antar pengguna dapat memperoleh data informasi yang sama tanpa harus melakukan *request* terlebih dahulu pada *server* untuk memperoleh *update* data terbaru pada tabel *alert* karena *server* dan *client* telah terhubung melalui adanya *WebSocket*. Dengan hasil tersebut penggunaan *WebSocket* berhasil dalam mentransmisikan informasi data antar pengguna secara *real-time*.

4. Kesimpulan

Dalam melakukan proses bisnisnya PT. XYZ Tbk sangat mengandalkan layanan dari *software* yang digunakan untuk keperluan proses bisnis sehari-hari. Dalam hal ini, layanan *software Control-M* yang digunakan sering mengalami anomali pada *job* dengan kondisi belum berjalan meski telah melewati *estimated start time*. Tim sistem operator yang bertugas memantau proses *job* sering mengalami hambatan dalam proses identifikasi. Pencarian terhadap *job* anomali yang ada pada *software* sering terlewatkan dikarenakan banyaknya proses *job* yang ada pada software. Selain itu kebutuhan proses transfer informasi antar anggota tim sistem operator dalam penanganan job anomali secara *real-time* dibutuhkan. Oleh karena itu, dalam artikel ini penulis mengajukan solusi dengan dibuatnya sistem identifikasi *job* anomali yang belum berjalan pada mainframe. Dengan dibuatnya sistem ini proses identifikasi menjadi lebih cepat, *job mainframe* yang mengalami anomali akan teridentifikasi secara otomatis sehingga tim sistem operator tidak perlu melakukan pencarian atau identifikasi secara manual terhadap *job* anomali pada *software Control-M*. Selain itu, dilakukanya implementasi *WebSocket* pada sistem identifikasi dapat mempercepat proses transfer data antar pengguna karena informasi yang didapatkan merupakan infromasi secara *real time*. Penelitian dalam artikel ini berhasil mengembangkan proses identifikasi yang lebih cepat dengan dibuatnya sistem identifikasi terhadap *job* anomali belum berjalan pada mainframe dengan mengambil data *job* secara langsung pada API *Control-M*, sehingga proses penanganan terhadap *job* anomali kepada PIC dapat lebih cepat dan meminimalisir gangguan pada proses bisnis perusahaan.

5. Ucapan Terima Kasih

Artikel ini disusun untuk memenuhi persyaratan tugas akhir strata 1 Program Studi Teknik Informatika Universitas Kristen Satya Wacana. Dengan penuh rasa syukur, penulis ingin mengucapkan terima kasih kepada semua pihak yang telah turut serta dalam perjalanan penyusunan tugas akhir ini:

1) Prof. Ir. Daniel H.F. Manongga, M.Sc., Ph.D. selaku Dekan Fakultas Teknologi Informasi Universitas Kristen Satya Wacana.
2) Budhi Kristianto, S.Kom., M.Sc., Ph.D. selaku Ketua Program Studi Teknik Informatika, Universitas Kristen Satya Wacana.
3) Christine Dewi, S.Kom., M.Cs., Ph.D. selaku dosen pembimbing yang telah menuntun dalam penyusunan artikel.
4) Teman-teman, senior dan mentor yang telah membantu dalam memberi pengetahuan.
5) Dukungan dan doa Ibu terkasih yang telah membantu penulis selama proses penyusunan artikel.

Tak lupa semua pihak yang terlibat langsung maupun tidak langsung dalam penyusunan artikel ini.

E-ISSN: 2723-7079 | P-ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May
OPEN ACCESS
https://doi.org/10.35870/jimik.v5i2.737

6. Daftar Pustaka

Borghesi, A., Bartolini, A., Lombardi, M., Milano, M., & Benini, L. (2019). A semisupervised autoencoder-based approach for anomaly detection in high performance computing systems. *Engineering Applications of Artificial Intelligence*, *85*, 634-644. DOI: https://doi.org/10.1016/j.engappai.2019.07.008.

Ding, Q. (2012). *BMC Control-M 7: A Journey from Traditional Batch Scheduling to Workload Automation*. Packt Publishing Ltd.

Gupta, Y., Dewan, H., & Leekha, A. (2020). Real-time monitoring using AJAX and WebSockets. *Journal of Statistics and Management Systems*, *23*(1), 125-134. DOI: https://doi.org/10.1080/09720510.2020.1714154.

Gutta, S., Shenoy, J., Kamath, S. P., Mithra, P., Baliga, B. S., Sarpangala, M., & Srinivasan, M. (2019). Light emitting diode (LED) phototherapy versus conventional phototherapy in neonatal hyperbilirubinemia: a single blinded randomized control trial from coastal India. *BioMed Research International*, *2019*.

Josephine, M., & Subramanian, D. K. S. (2009). Software error detection and correction (SEDC) using layer based approach in expert system. *International Journal of Reviews in Computing, ISSN*, 2076-3328.

Kadir, A. (2009). Telaah Ajax Untuk Mengejar Ketertinggalan Aplikasi Web Terhadap Aplikasi Desktop. *Jurnal Teknologi Ist Akprind*, *2*.

Kothari, S. C., Bishop, L., Sauceda, J., & Daugherty, G. (2004). A pattern-based framework for software anomaly detection. *Software Quality Journal*, *12*, 99-120.

Maulana, R., & Alauddin Makassar, N. (2021). Implementasi Web Socket Pada Sistem Pelayanan Pasien Rawat Jalan Pada Puskesmas Kabupaten Gowa. *J. INSTEK (Informatika Sains dan Teknol., vol. 6, no. 1, p. 130, 2021, doi: 10.24252/instek. v6i1. 20555*.

Ogundeyi, K. E., & Yinka-Banjo, C. (2019). WebSocket in real time application. *Nigerian Journal of Technology*, *38*(4), 1010-1020.

Pang, G., Shen, C., Cao, L., & Hengel, A. V. D. (2021). Deep learning for anomaly detection: A review. *ACM computing surveys (CSUR)*, *54*(2), 1-38.

Shi, C., Yuan, F., Li, Z., Zheng, Z., Yuan, C., Huang, Z., ... & Ding, Z. (2022). MSN@ IL-4 sustainingly mediates macrophagocyte M2 polarization and relieves osteoblast damage via NF-κB pathway-associated apoptosis. *BioMed Research International*, *2022*.

Shi, T., Zou, Z., & Ai, J. (2023). Software Operation Anomalies Diagnosis Method Based on a Multiple Time Windows Mixed Model. *Applied Sciences*, *13*(20), 11349.

Suryn, W. (2013). *Software quality engineering: a practitioner's approach*. John Wiley & Sons.

Taha, A., & Hadi, A. S. (2019). Anomaly detection methods for categorical data: A review. *ACM Computing Surveys (CSUR)*, *52*(2), 1-35.

Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id

E - ISSN: 2723-7079 | P - ISSN: 2776-8074
Vol. 5 No. 2 (2024) | May

OPEN ACCESS

https://doi.org/10.35870/jimik.v5i2.737

Utomo, S. B., Hidayah, M., & Anjani, N. L. (2022). DETEKSI DINI IDENTIFIKASI INSIDEN PADA KEJADIAN ANOMALI PERANGKAT LUNAK DENGAN SISTEM PENDETEKSI ANOMALI PERANGKAT LUNAK STUDI KASUS DI ASTRA LIFE: indonesia. *Technologic Politeknik Astra*, *13*(1). DOI: https://doi.org/10.52453/t.v13i1.390.