Journal of Informatics Engineering and Software Applications

Implementation of Real-Time Chat Using Pusher on Vocaject Application to Support Project-Based Learning

Saiful Kamil[1], Husaini[2*], Fachri Yanuar Rudi F[3]

[1,2,3] Jurusan Teknologi Informasi dan Komputer Politeknik Negeri Lhokseumawe
Jln. B.Aceh Medan Km.280 Buketrata 24301 INDONESIA
*Penulis Korespondensi : husaini @pnl.ac.id

INFORMASI ARTIKEL

Riwayat artikel:
Diajukan pada 00 Maret 00
Direvisi pada 00 April 00
Publikasi pada 00 Juni 00

Kata kunci:
Pusher
Pusher Channels
Realtime Chat
Black Box Testing
Quality of Service

Keywords:
Pusher
Pusher Channels
Realtime Chat
Black Box Testing
Quality of Service

ABSTRAK

Perkembangan teknologi informasi yang pesat telah melahirkan berbagai platform digital yang dirancang untuk memfasilitasi komunikasi dan kolaborasi jarak jauh. Namun, efektivitas komunikasi sering kali terhambat karena tidak semua aplikasi menyediakan fitur interaksi yang optimal secara *real-time*. *Vocational Project* (Vocaject), sebuah aplikasi pendukung metode pembelajaran *Project-Based Learning* (PBL) di lingkungan akademis, saat ini menghadapi kendala dalam memfasilitasi komunikasi instan antara dosen, mahasiswa, dan mitra industri karena belum tersedianya fitur *chat* yang andal. Penelitian ini bertujuan untuk mengatasi kesenjangan tersebut melalui implementasi fitur *real-time chat* menggunakan teknologi Pusher yang diintegrasikan ke dalam aplikasi Vocaject. Metodologi pengembangan mencakup perancangan sistem berbasis *websocket*, implementasi antarmuka, serta pengujian fungsionalitas dan kinerja jaringan. Validasi fungsional dilakukan menggunakan metode *Black Box Testing*, sementara kinerja jaringan dianalisis menggunakan parameter *Quality of Service* (QoS) yang meliputi *throughput*, *packet loss*, *delay*, dan *jitter*. Hasil pengujian fungsional menunjukkan tingkat keberhasilan 100% pada seluruh skenario penggunaan. Sementara itu, analisis QoS pada pengiriman data dari pengguna ke server menghasilkan rata-rata *throughput* sebesar 5.858 kbps, *packet loss* 8,31%, *delay* 0,992 ms, dan *jitter* 0,984 ms, yang dikategorikan sebagai kinerja jaringan yang baik. Sebaliknya, respon balik dari server ke pengguna mencatat rata-rata *throughput* 60,2 kbps dengan *delay* 102,9 ms. Implementasi ini diharapkan dapat meningkatkan efisiensi koordinasi proyek dalam ekosistem Vocaject secara signifikan.

ABSTRACT

The rapid advancement of information technology has given rise to various digital platforms designed to facilitate remote communication and collaboration. However, communication effectiveness is often hindered because not all applications provide optimal real-time interaction features. Vocational Project (Vocaject), an application supporting Project-Based Learning (PBL) in academic environments, currently faces challenges in facilitating instant communication between lecturers, students, and industrial partners due to the lack of a reliable chat feature. This study aims to bridge this gap by implementing a real-time chat feature using Pusher technology integrated into the Vocaject application. The development methodology includes websocket-based system design, interface implementation, as well as functionality and network performance testing. Functional validation was conducted using the Black Box Testing method, while network performance was

analyzed using Quality of Service (QoS) parameters including throughput, packet loss, delay, and jitter. Functional test results showed a 100% success rate across all usage scenarios. Meanwhile, QoS analysis on data transmission from user to server resulted in an average throughput of 5.858 kbps, packet loss of 8.31%, delay of 0.992 ms, and jitter of 0.984 ms, categorized as good network performance. Conversely, the response from server to user recorded an average throughput of 60.2 kbps with a delay of 102.9 ms. This implementation is expected to significantly improve project coordination efficiency within the Vocaject ecosystem.

1. Pendahuluan

Dalam era digitalisasi industri 4.0, efisiensi komunikasi menjadi fondasi utama dalam keberhasilan manajemen proyek dan proses pembelajaran. Kemajuan teknologi informasi telah mengubah paradigma interaksi manusia, memungkinkan pertukaran informasi terjadi tanpa batasan ruang dan waktu. Salah satu bentuk implementasi teknologi komunikasi yang paling krusial adalah layanan pesan instan atau *chatting*. Teknologi ini tidak hanya mempercepat proses pengambilan keputusan tetapi juga meminimalkan hambatan administratif yang sering terjadi pada komunikasi konvensional. *Chatting* memungkinkan pertukaran pesan teks, suara, maupun video secara langsung (*real-time*) melalui jaringan internet, menjadikannya elemen vital dalam aplikasi kolaboratif.

Aplikasi *Vocational Project* atau Vocaject merupakan sebuah platform inovatif yang dikembangkan untuk mendukung implementasi kurikulum *Project-Based Learning* (PBL) di perguruan tinggi vokasi. Aplikasi ini berfungsi sebagai jembatan antara dunia akademik (kampus) dan dunia usaha/dunia industri (DUDI). Melalui Vocaject, industri dapat mengunggah proyek nyata yang kemudian dikerjakan oleh mahasiswa di bawah bimbingan dosen. Fitur-fitur yang ada saat ini meliputi pengelolaan proyek, pengajuan proposal, serta pengawasan logbook kegiatan mahasiswa. Meskipun Vocaject telah memiliki fitur manajemen proyek yang komprehensif, terdapat kekurangan fundamental yang menghambat efektivitas kolaborasi, yaitu ketiadaan fitur komunikasi langsung (*live chat*) yang terintegrasi di dalam aplikasi.

Saat ini, komunikasi antara dosen pembimbing, mahasiswa, dan mitra industri sering kali dilakukan melalui aplikasi pihak ketiga yang tidak terintegrasi dengan sistem manajemen proyek. Hal ini menyebabkan fragmentasi informasi dan kesulitan dalam melacak riwayat diskusi terkait proyek tertentu. Selain itu, tantangan teknis dalam pengembangan fitur *chat* mandiri adalah masalah latensi atau *delay*. Layanan *chat* konvensional sering mengalami penundaan pengiriman pesan yang signifikan seiring dengan bertambahnya jumlah pengguna aktif secara bersamaan. Fenomena ini dapat menurunkan *User Experience* (UX) dan menghambat penyelesaian masalah proyek yang bersifat mendesak. Oleh karena itu, diperlukan sebuah solusi teknis yang mampu menjamin *Quality of Service* (QoS) yang tinggi, terutama dalam aspek kecepatan pengiriman dan penerimaan pesan.

Sebagai solusi atas permasalahan tersebut, penelitian ini mengusulkan implementasi teknologi *Websocket* menggunakan layanan Pusher sebagai *message broker*. Pusher dipilih karena kemampuannya dalam menyediakan infrastruktur komunikasi *real-time* yang dapat diskalakan (*scalable*) dan andal. Berbeda dengan protokol HTTP standar yang mengharuskan klien untuk terus-menerus meminta pembaruan data (*polling*), Pusher memanfaatkan koneksi *full-duplex* melalui *Websockets*. Teknologi ini memungkinkan

server untuk mengirimkan data (pesan) secara proaktif ke klien segera setelah data tersedia, sehingga latensi komunikasi dapat ditekan seminimal mungkin. Penggunaan *Pusher Channels* memfasilitasi komunikasi dua arah yang stabil antara server dan aplikasi berbasis Android yang digunakan oleh pengguna Vocaject.

Penelitian terdahulu menunjukkan bahwa penggunaan *framework* pengembangan aplikasi *mobile* seperti Flutter yang dikombinasikan dengan layanan *backend real-time* mampu menghasilkan aplikasi dengan performa tinggi. Flutter, sebagai *Software Development Kit* (SDK) besutan Google, memungkinkan pengembangan aplikasi *multi-platform* (Android dan iOS) dengan satu basis kode, yang mempercepat proses pengembangan fitur baru pada Vocaject. Integrasi antara Flutter di sisi klien dan Pusher di sisi *backend* diharapkan dapat menciptakan ekosistem komunikasi yang responsif.

Tujuan utama dari penelitian ini adalah merancang bangun fitur *chat* pada aplikasi Vocaject dan melakukan analisis mendalam terhadap kinerjanya. Analisis tidak hanya terbatas pada keberhasilan fungsi (fungsionalitas), tetapi juga mencakup pengukuran parameter kualitas jaringan (QoS) seperti *throughput*, *packet loss*, *delay*, dan *jitter*. Dengan demikian, penelitian ini berkontribusi dalam memberikan bukti empiris mengenai kelayakan teknis penggunaan Pusher dalam mendukung komunikasi *real-time* pada aplikasi manajemen pembelajaran berbasis proyek.

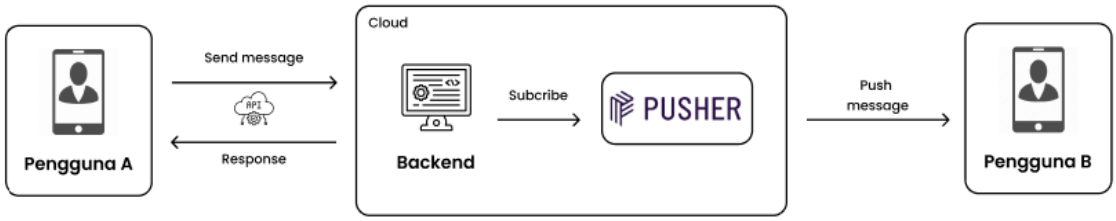
2. Metode

2.1 Arsitektur dan Perancangan Sistem

Infrastruktur *backend* sistem ini dibangun di atas layanan *Google Cloud Platform* (GCP). Spesifikasi server yang digunakan adalah *Compute Engine* tipe n1-standard-1 yang dilengkapi dengan 1 vCPU, RAM sebesar 3,8 GB, dan penyimpanan HDD 10 GB. Server ini beroperasi menggunakan sistem operasi Ubuntu 22.04 LTS dan ditempatkan di zona *asia-southeast1* untuk meminimalkan latensi fisik dengan pengguna di Indonesia. Sebagai inti dari komunikasi *real-time*, penelitian ini memanfaatkan layanan *Pusher Channel* (paket *sandbox free*) yang mampu menangani hingga 200.000 pesan per hari dengan batas 100 koneksi konkuren (bersamaan), yang dinilai cukup untuk kebutuhan pengembangan dan pengujian tahap awal.

Alur komunikasi data dirancang sebagai berikut:

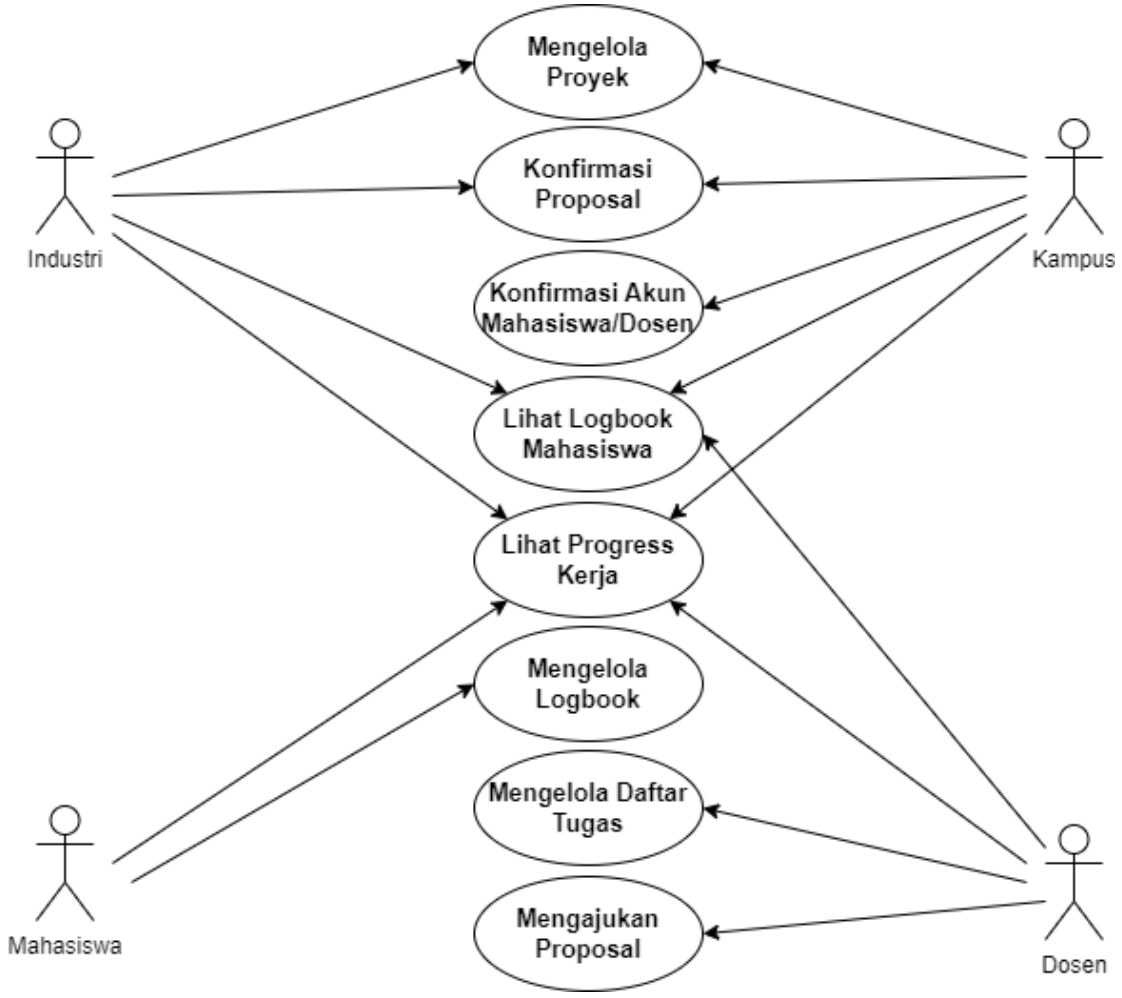
1. Pengguna (Dosen/Industri/Mahasiswa) berinteraksi melalui aplikasi *mobile* Vocaject (dibangun dengan Flutter).
2. Saat pengguna mengirim pesan, aplikasi melakukan *HTTP Request* (POST) ke *Backend Server* melalui REST API yang diamankan dengan protokol HTTPS.
3. *Backend Server* memvalidasi sesi pengguna dan hak akses, kemudian menyimpan data pesan ke dalam basis data (*database*) untuk keperluan riwayat percakapan.
4. Setelah penyimpanan berhasil, *backend* meneruskan *payload* pesan ke layanan Pusher melalui API Pusher.
5. Pusher kemudian mem-broadcast pesan tersebut ke "Channel" spesifik yang sedang didengarkan (*subscribe*) oleh pengguna penerima.
6. Aplikasi penerima yang memiliki koneksi *Websocket* aktif ke Pusher akan menerima pesan tersebut secara instan tanpa perlu melakukan *refresh* halaman.



Gambar 1. Arsitektur Perancangan Sistem Fitur Chat

2.2 Perancangan Interaksi Pengguna (*Use Case*)

Analisis kebutuhan pengguna dipetakan melalui *Use Case Diagram*. Terdapat empat aktor utama dalam ekosistem Vocaject: Industri, Kampus, Dosen, dan Mahasiswa. Masing-masing aktor memiliki peran spesifik. Industri dan Dosen adalah pengguna utama fitur *chat* untuk koordinasi proyek. Dosen dapat mengajukan proposal dan memantau logbook, sementara Industri dapat mengonfirmasi proposal dan memantau progres. Fitur *chat* menjadi jembatan komunikasi antara kedua entitas ini untuk mendiskusikan detail teknis proyek yang tidak dapat ditampung dalam format laporan formal.



Gambar 2. Diagram Use Case Aplikasi Vocaject

2.3 Desain Alur Aktivitas (Activity Diagram)

Alur kerja fitur *chat* dirancang untuk memastikan validitas data sebelum pesan terkirim. Proses dimulai saat pengguna (User A) mengirim pesan. Sistem akan memvalidasi inputan; jika valid, pesan disimpan ke *database*. Langkah krusial berikutnya adalah sistem melakukan *trigger* ke Pusher. Pusher kemudian mendistribusikan pesan tersebut ke klien lain yang terhubung pada *channel* yang sama (User B). Diagram aktivitas ini memastikan bahwa setiap pesan yang tampil di layar pengguna penerima adalah pesan yang sudah terkonfirmasi tersimpan di server.

2.4 Skenario Pengujian dan Parameter Kualitas

Pengujian dilakukan dalam dua tahap utama:

1. **Black Box Testing:** Fokus pada validasi fungsionalitas fitur tanpa melihat kode internal. Pengujian ini memastikan input yang diberikan menghasilkan output yang diharapkan pada berbagai modul antarmuka (Login, Register, Chatting).
2. **Analisis Quality of Service (QoS):** Fokus pada kinerja jaringan. Pengukuran dilakukan menggunakan perangkat lunak *Wireshark* untuk menangkap paket data dan *Apache JMeter* untuk simulasi beban pengguna. Parameter yang diukur mengacu pada standar TIPHON (*Telecommunications and Internet Protocol Harmonization Over Networks*), meliputi:
    a. **Throughput:** Kecepatan rata-rata transfer data efektif.
    b. **Packet Loss:** Persentase paket data yang gagal mencapai tujuan.
    c. **Delay (Latensi):** Waktu yang dibutuhkan data untuk menempuh jarak dari asal ke tujuan.
    d. **Jitter:** Variasi kedatangan paket data (ketidakstabilan delay).

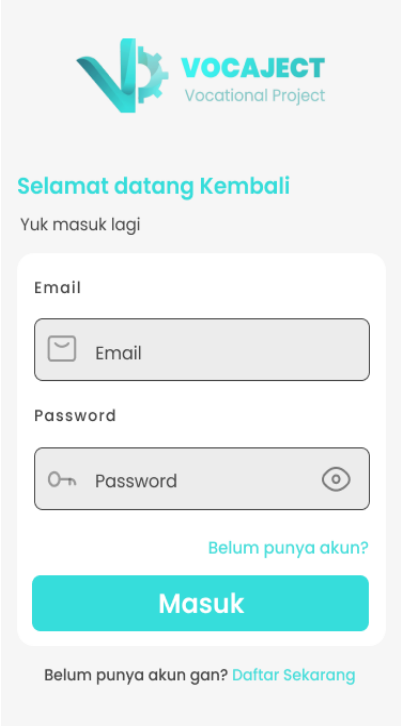
Pengujian QoS dilakukan dengan skenario pembebanan bertingkat, mulai dari 10, 50, 100, hingga 500 sampel pengguna virtual untuk melihat ketahanan sistem di bawah tekanan trafik yang berbeda.

3. Hasil Dan Pembahasan

3.1 Implementasi Antarmuka Pengguna (*User Interface*)

Implementasi antarmuka pengguna dibangun menggunakan *framework* Flutter dengan desain yang minimalis namun fungsional untuk memudahkan navigasi pengguna.

1. **Halaman Login:** Menjadi gerbang utama autentikasi. Sistem memverifikasi kredensial pengguna (email dan sandi) sebelum memberikan akses ke fitur utama.



Gambar 3. Tampilan Halaman *Login*

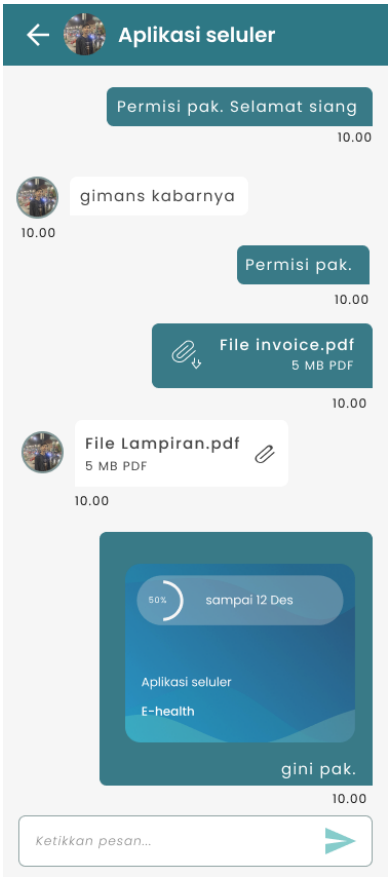
2. **Halaman Utama (Dashboard):** Menampilkan daftar proyek aktif dan status terkini. Ini memudahkan pengguna untuk memilih proyek mana yang ingin didiskusikan.

Gambar 4. Tampilan Halaman Utama

3. **Halaman Detail Proyek:** Menyediakan informasi komprehensif mengenai spesifikasi proyek, tenggat waktu, dan anggaran. Di halaman ini terdapat tombol akses cepat menuju fitur *chat* dengan pemilik proyek.

Gambar 5. Tampilan Halaman *Detail Project*

4. **Halaman Chat:** Merupakan inti dari implementasi penelitian ini. Antarmuka *chat* mendukung pertukaran pesan teks, gambar, dan dokumen. Desain balon pesan (*chat bubble*) membedakan antara pesan pengirim dan penerima untuk keterbacaan yang lebih baik. Status pengiriman pesan dipantau secara *real-time* berkat integrasi Pusher.



Gambar 6. Tampilan Halaman *Chat*

3.2 Hasil Pengujian Fungsional (*Black Box Testing*)

Pengujian *Black Box* dilakukan terhadap 7 skenario krusial yang mencakup alur registrasi, login, hingga proses *chatting* untuk berbagai tipe pengguna (Industri dan Dosen). Melibatkan 20 responden pengguna aplikasi Vocaject, pengujian ini bertujuan untuk memverifikasi bahwa logika aplikasi berjalan sesuai spesifikasi kebutuhan.

Tabel 1 merangkum hasil pengujian tersebut. Terlihat bahwa seluruh kasus uji (Test Case 01 s.d. 07) menghasilkan status "Berhasil". Tidak ditemukan *bug* kritikal yang menghambat fungsi utama aplikasi. Hal ini mengindikasikan bahwa integrasi API Pusher dengan aplikasi Android telah terimplementasi dengan logika pemrograman yang benar.

Tabel 1. Hasil Pengujian Black Box Testing

Test Case	Pengguna Berhasil	Pengguna Tidak Berhasil	Persentase	Hasil
01	20	0	20/20× 100%	100%
02	20	0	20/20× 100%	100%
03	20	0	20/20× 100%	100%
04	20	0	20/20× 100%	100%
05	20	0	20/20× 100%	100%
06	20	0	20/20× 100%	100%
07	20	0	20/20× 100%	100%
		Rata - rata		100%

3.3 Analisis *Quality of Service* (QoS): Pengiriman Data User ke Server

Kinerja jaringan saat pengguna mengirimkan data (pesan/file) ke server diuji dengan mengirimkan paket file berukuran 275 KB. Simulasi dilakukan dengan variasi beban sampel (threads) menggunakan JMeter. Data lalu lintas jaringan direkam menggunakan Wireshark.

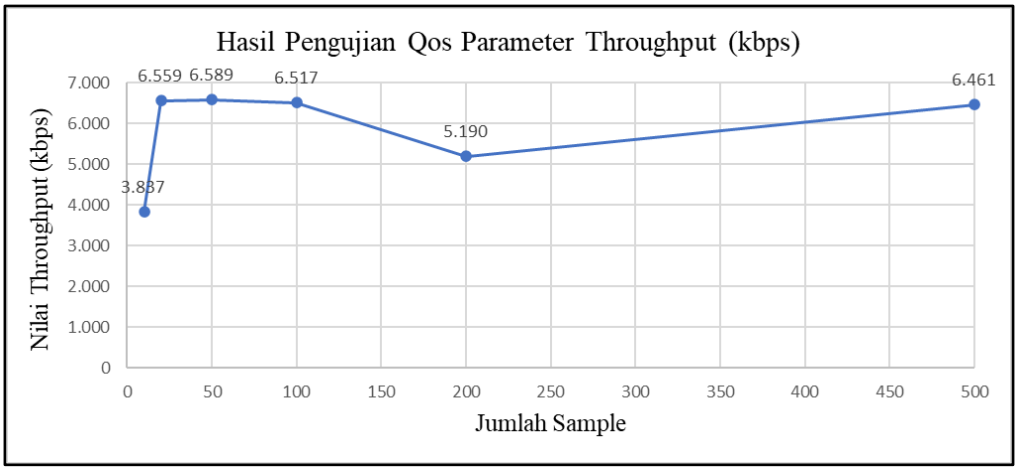
Hasil pengukuran QoS dari sisi *User to Server* dirangkum dalam Tabel 2.

Tabel 2. Hasil Pengujian Mengirim Data Dari User Ke Server

Sample	Throughput (kbps)	Packet Loss (%)	Delay (ms)	Jitter (ms)
10	3.837	0	1,554	1,520
20	6.559	0	0,915	0,901
50	6.589	0	0,922	0,921
100	6.517	1	0,839	0,839
200	5.190	13	1,426	1,428
500	6.461	35.9	1,586	1,586
Rata - rata	5.858	8,31 %	1.207	1,199

Berdasarkan Tabel 2, analisis mendalam dapat dilakukan sebagai berikut:

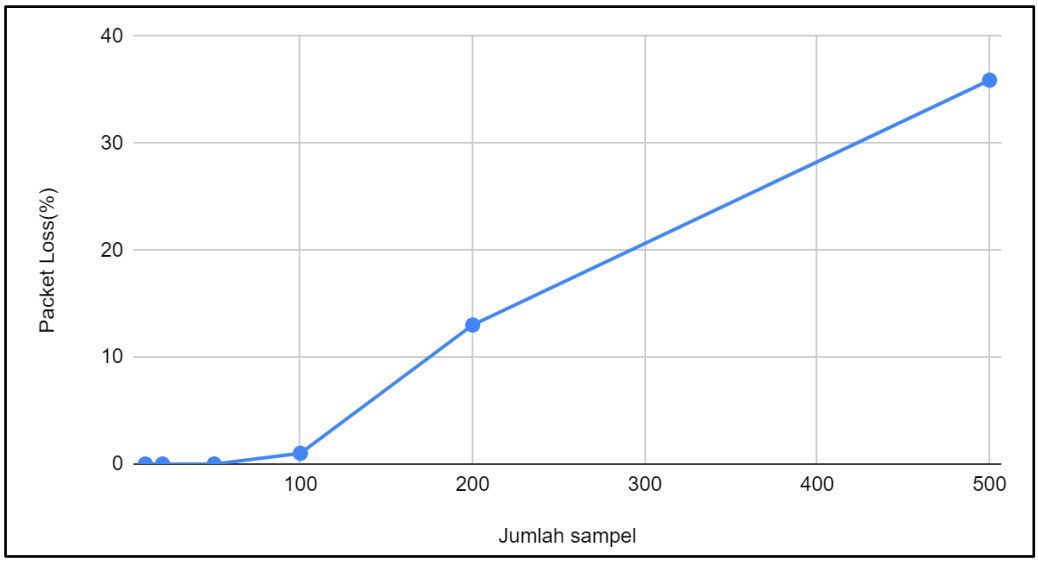
1. **Analisis Throughput:** Grafik pengujian menunjukkan *throughput* rata-rata sebesar 5.858 kbps. Nilai ini relatif stabil meskipun beban pengguna ditingkatkan hingga 500. Stabilitas *throughput* ini menunjukkan bahwa server *cloud* dan layanan Pusher mampu menangani *bandwidth* masuk dengan cukup baik.



Gambar 7. Grafik Pengujian Qos *Parameter Throughput*

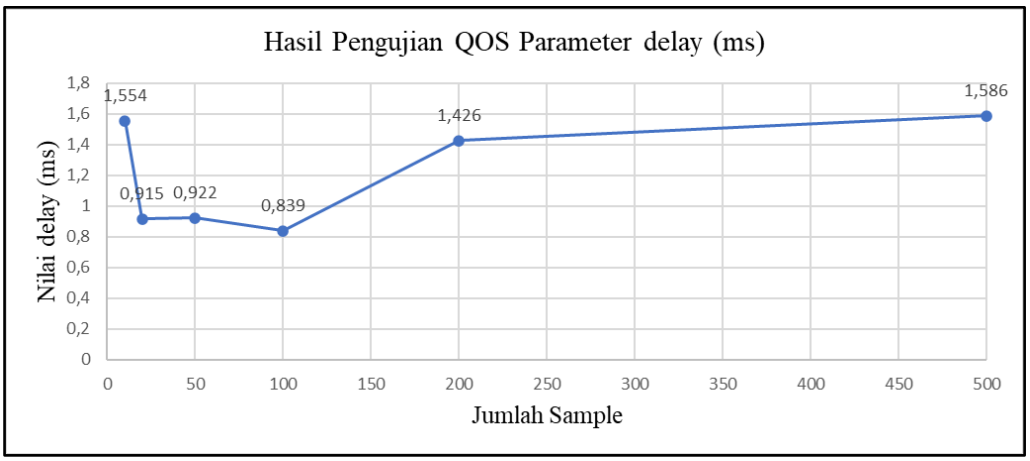
2. **Analisis Packet Loss:** Parameter *packet loss* menunjukkan tren yang menarik. Pada beban rendah (10-50 sampel), *packet loss* adalah 0% (sangat baik). Namun, terjadi lonjakan signifikan pada beban 200 sampel (13%) dan 500 sampel (35.9%). Hal ini wajar terjadi dalam pengujian beban tinggi (*stress testing*) di mana antrian proses pada server mulai penuh. Meskipun rata-rata keseluruhan (8.31%) masih dalam kategori "Bagus" menurut standar TIPHON, angka 35.9% pada beban puncak

mengindikasikan perlunya optimalisasi server jika aplikasi akan digunakan oleh ribuan pengguna secara bersamaan.

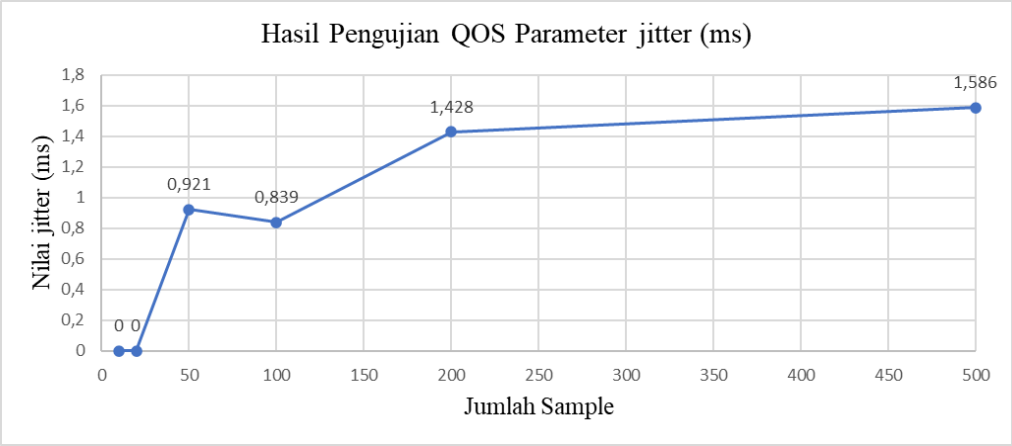


Gambar 8. Grafik Pengujian Qos Parameter *Packet Loss*

3. **Analisis Delay dan Jitter:** Nilai *delay* rata-rata sangat rendah (1.207 ms) dan *jitter* (1.199 ms). Rendahnya latensi ini adalah keunggulan utama penggunaan arsitektur *Websocket*. Pesan terkirim hampir seketika. Bahkan pada beban 500 pengguna, *delay* hanya naik sedikit menjadi 1,586 ms, yang secara persepsi pengguna masih terasa instan (*real-time*). [



Gambar 9. Grafik Pengujian Qos Parameter Delay



Gambar 10. Grafik Pengujian Qos Parameter *Jitter*

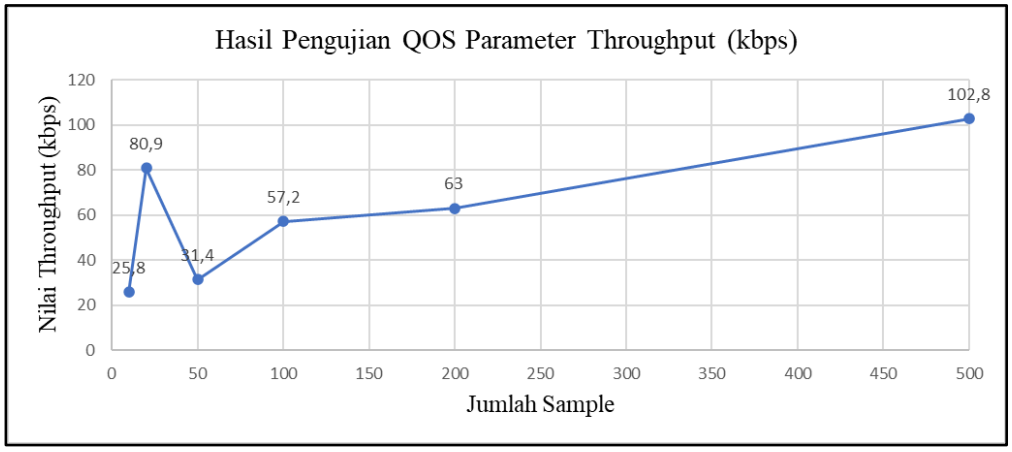
3.3 Analisis Quality of Service (QoS): Respon Server ke User

Pengujian tahap kedua mengukur kinerja saat server mengirimkan balasan atau notifikasi ke pengguna (*downlink*). Ini merepresentasikan kecepatan pesan diterima di aplikasi pengguna lain.

Tabel 3. Hasil Pengukuran Response Time Server ke User

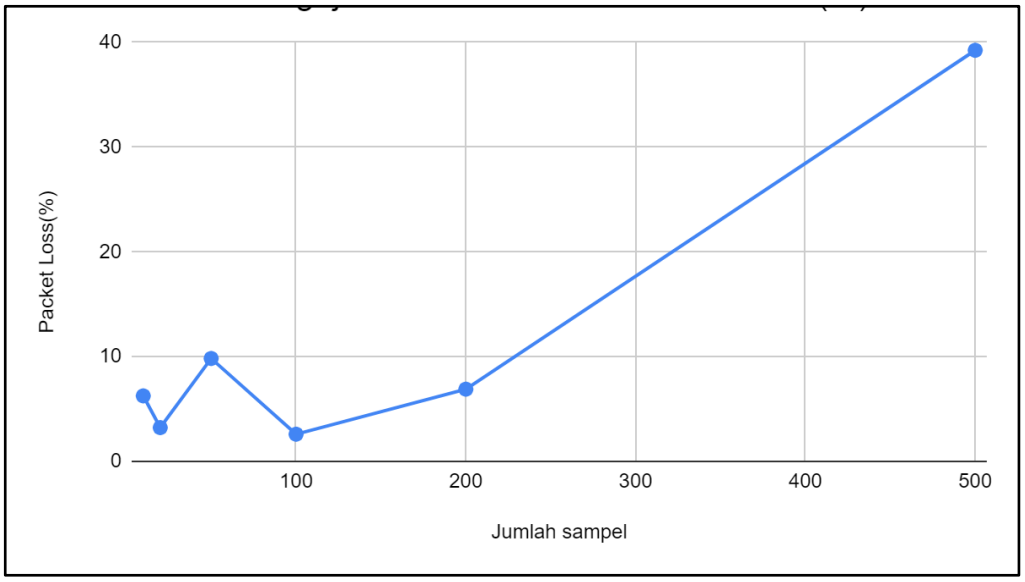
Sample	Throughput (kbps)	Packet Loss (%)	Delay (ms)	Jitter (ms)
10	25,8	6,25	151,47	134,27
20	80,9	3,22	78,47	78,47
50	31,4	9,82	96,16	96,17
100	57,2	2,59	122,20	122,19
200	63,0	6,89	79,31	78,54
500	102,8	39,24	89,79	89,73
Rata - rata	60,23	11,33	102,9	99,895

1. **Diskusi Kinerja Downlink:** Rata-rata *throughput* pada sisi *downlink* (60,23 kbps) terlihat lebih kecil dibandingkan *uplink*. Hal ini sebagian disebabkan oleh karakteristik data respon JSON yang ukurannya kecil namun frekuensinya tinggi. Meskipun secara angka *throughput* terlihat rendah dan mungkin dikategorikan "Buruk" jika hanya melihat *bandwidth* mentah, dalam konteks aplikasi *chat* teks, ini tidak menjadi masalah besar karena data yang dikirim memang kecil.

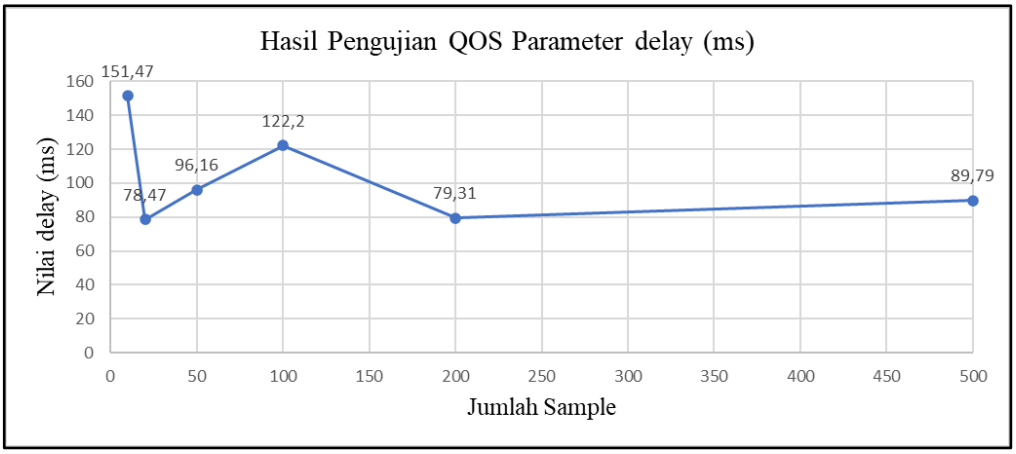


Gambar 11. Grafik Pengujian *Response Time* Qos Parameter *Throughput*

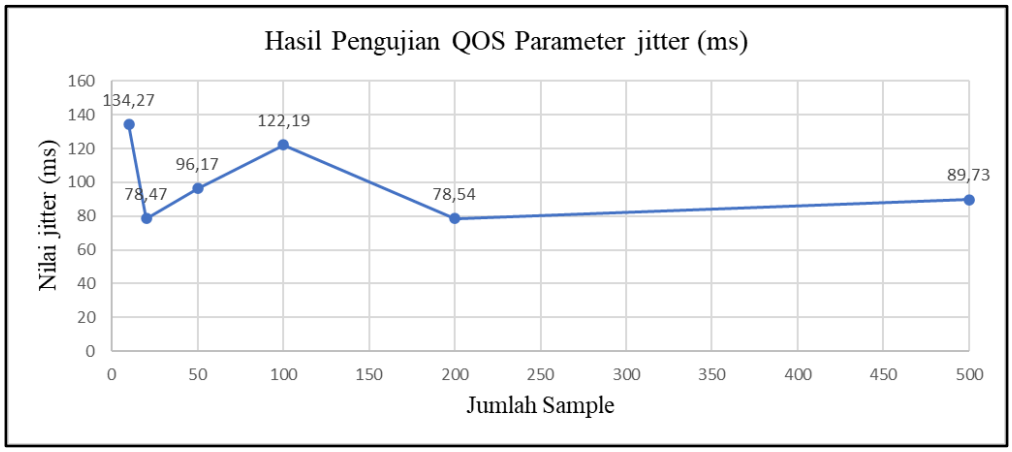
2. **Evaluasi Stabilitas Koneksi:** Rata-rata *Packet Loss* sebesar 11,33% dengan lonjakan hingga 39% pada beban 500 pengguna menunjukkan adanya kemacetan (*bottleneck*) pada sisi emulator atau koneksi internet penguji saat menangani trafik *broadcast* yang masif. Delay rata-rata sekitar 102 ms masih dapat diterima untuk komunikasi manusia (di bawah 150 ms masih dianggap responsif untuk suara/video, apalagi teks).



Gambar 12. Grafik Pengujian *Response Time* Qos Parameter *Packet Loss*



Gambar 13. Grafik Pengujian *Response Time* Qos Parameter *Delay*



Gambar 14. Grafik Pengujian *Response Time* Qos Parameter *Jitter*

Secara keseluruhan, meskipun terdapat degradasi performa pada beban ekstrem (500 pengguna konkuren dalam satu detik), sistem secara umum menunjukkan kinerja yang sangat responsif untuk skenario penggunaan normal. Penggunaan Pusher terbukti efektif menjaga latensi tetap rendah (*low latency*) yang krusial untuk pengalaman *chatting*.

4. Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan serangkaian pengujian yang telah dilakukan pada fitur *real-time chat* aplikasi Vocaject, dapat ditarik beberapa kesimpulan utama. Pertama, integrasi teknologi Pusher dengan aplikasi berbasis Flutter berhasil dilakukan dengan baik. Pengujian fungsional menggunakan metode *Black Box Testing* membuktikan bahwa fitur ini 100% berfungsi sesuai rancangan awal pada tujuh skenario utama, mencakup proses registrasi, autentikasi, hingga pertukaran pesan antar pengguna industri dan dosen.

Kedua, dari sisi kinerja jaringan, sistem menunjukkan reliabilitas yang memadai untuk mendukung komunikasi *real-time*. Pengujian parameter *Quality of Service* (QoS) pada skenario pengiriman data dari pengguna ke server menghasilkan rata-rata *delay* yang sangat rendah (1,2 ms) dan *jitter* yang stabil (1,19 ms), menjamin pengalaman pengguna yang responsif. Meskipun terdapat peningkatan *packet loss* pada pengujian beban tinggi (500 pengguna), kinerja pada beban normal hingga menengah masuk dalam kategori "Bagus" dan "Sangat Bagus".

Penelitian ini merekomendasikan adanya optimasi pada sisi infrastruktur server atau penyesuaian paket layanan Pusher jika aplikasi akan di-deploy untuk ribuan pengguna aktif guna meminimalisir *packet loss*. Namun, untuk kebutuhan lingkungan akademis saat ini, solusi yang dibangun telah memenuhi standar operasional yang diharapkan untuk mendukung metode *Project-Based Learning* di Vocaject.

Referensi

[1] L. Safitri And S. Basuki, "Analisa Dan Perancangan Sistem Informasi Text Chatting Berbasis Android Web View," Jurnal Ipsikom , Vol. 8, 2020.

[2] M. Juansen And S. Simatupang, "Integrasi Mesin Absensi Dan Pusher Notification Pada Sistem Informasi Akademik Sekolah Untuk Monitoring Absensi Real-Time," Journal Of Computer System And Informatics (Josyc), Vol. 4, No. 4, Pp. 1028–1035, Aug. 2023, Doi: 10.47065/Josyc.V4i4.3840.

[3] Y. Kurnia And G. Aditya, "Online Learning Service Application Using Flutter Framework And Laravel," 2022. Available:

Http://Bsti.Ubd.Ac.Id/E-Jurnal

[4] Y. Dwi Putri And S. Lutfi, "Penerapan Kriptografi Caesar Cipher Pada Fitur Chatting Sistem Informasi Freelance," Jurnal Informatika Dan Komputer P-Issn, Vol. 2, No. 2, Pp. 2355–7699, 2019, Doi: 10.33387/Jiko.

[5] M. Agung And I. Aplikasi, "Implementasi Aplikasi Pembuatan Chat Implementation Of The Chat Creation Application 1."

[6] A. Yusuf, B. Setiawan, And S. R. Nudin, "Pengembangan Aplikasi Mobile 'simpel' Bnnp Jatim Dengan Fitur Real-Time Chat Dan Geotagging," Journal Of Informatics And Computer Science, Vol. 04, 2022,. Available: Https://Simpelbnnpjatim.Com

[7] Pusher. (2023). Pusher Channels - realtime communication for developers. Retrieved Mei 1, 2024, From https://pusher.com/

[8] R. Riski, Husaini, And M. Nasir, "Implementasi Socket Programming Pada Aplikasi Chat Uloen Messenger Berbasis Android," Journal Of Artificial Intelligence And Software Engineering. 2023.

[9] P. R. Utami, "Analisis Perbandingan Quality Of Service Jaringan Internet Berbasis Wireless Pada Layanan Internet Service Provider (Isp) Indihome Dan First Media," Jurnal Ilmiah Teknologi Dan Rekayasa, Vol. 25, No. 2, Pp. 125–137, 2020, Doi: 10.35760/Tr.2020.V25i2.2723.

[10] M. Hasbi And N. R. Saputra, "Analisis Quality Of Service (Qos) Jaringan Internet Kantor Pusat King Bukopin Dengan Menggunakan Wireshark," 2021. Available: Https://Jurnal.Umj.Ac.Id/Index.Php/Just-It/Index

[11] R. Puspita Sari And S. Rahmayuda, "Implementasi Framework Flutter Pada Sistem Informasi Perpustakaan Masjid (Studi Kasus: Masjid Di Kota Pontianak)," Jurnal Komputer dan Aplikasi 2022.

[12] D. Gustian, Y. Fitrisia, S. Purwantoro Esgs, And W. Novayani, "Implementasi Automation Deployment Pada Google Cloud Compute Vm Menggunakan Terraform," Jurnal Inovtek Polbeng - Seri Informatika, Vol. 8, No. 2, 2023.