Journal of Computer Science and Information Technology (JCSIT)
Volume 3, No 3 – Juni 2026
e-ISSN : 3031-8467

PENERAPAN STRUKTUR DATA STACK UNTUK IMPLEMENTASI FITUR UNDO PADA APLIKASI SEDERHANA: LITERATURE REVIEW

Rifqi Atha Ramadhan[1], Ines Heidiani Ikasari[2]
[1,2]Universitas Pamulang, Indonesia
[1]rifqiatha@mhs.unpam.ac.id

Received: 29-05- 2026	Revised: 03-06-2026	Approved: 25-06-2026

ABSTRAK

Penelitian ini bertujuan untuk mengidentifikasi berbagai pendekatan implementasi struktur data stack pada fitur undo, menganalisis efektivitas dan efisiensi masing-masing pendekatan, serta memberikan rekomendasi bagi pengembang perangkat lunak dalam memilih strategi implementasi yang sesuai dengan karakteristik aplikasi. Metode penelitian yang digunakan adalah Systematic Literature Review (SLR) dengan mengacu pada pedoman PRISMA, melalui tahapan identifikasi, penyaringan, penilaian kelayakan, dan seleksi literatur. Dari 87 artikel yang ditemukan, diperoleh 15 artikel yang memenuhi kriteria inklusi dan digunakan sebagai sumber utama kajian. Hasil penelitian menunjukkan bahwa struktur data stack merupakan pendekatan yang paling sesuai untuk implementasi fitur undo karena menerapkan prinsip Last In First Out (LIFO) yang sejalan dengan mekanisme pembatalan tindakan pengguna. Operasi dasar stack memiliki kompleksitas waktu O(1) sehingga mampu memberikan respons yang cepat dan efisien. Kajian juga mengidentifikasi tiga variasi utama implementasi stack, yaitu bounded stack, persistent stack, dan double-stack, yang masing-masing memiliki keunggulan dan keterbatasan terkait efisiensi memori, pengelolaan riwayat perubahan, dan kompleksitas implementasi. Selain itu, integrasi stack dengan command pattern terbukti meningkatkan modularitas dan fleksibilitas sistem. Simpulan penelitian ini adalah bahwa struktur data stack tetap menjadi solusi yang efektif, efisien, dan relevan untuk implementasi fitur undo pada berbagai platform perangkat lunak modern, sehingga pemilihan variasi implementasinya perlu disesuaikan dengan kebutuhan fungsional, sumber daya sistem, dan kompleksitas aplikasi yang dikembangkan.

Kata Kunci: *Bounded Stack, Command Pattern, Fitur Undo, Persistent Stack, Struktur Data Stack*

PENDAHULUAN

Perkembangan teknologi informasi dan perangkat lunak telah mendorong peningkatan kebutuhan terhadap aplikasi yang tidak hanya memiliki fungsi yang lengkap, tetapi juga mampu memberikan pengalaman pengguna (user experience) yang baik. Salah satu aspek penting dalam pengalaman pengguna adalah kemampuan sistem untuk mengakomodasi kesalahan yang dilakukan selama proses interaksi. Dalam konteks ini, fitur undo menjadi salah satu mekanisme yang sangat penting karena memungkinkan pengguna membatalkan tindakan yang telah dilakukan dan mengembalikan sistem ke kondisi sebelumnya. Fitur tersebut banyak diterapkan pada berbagai jenis aplikasi, seperti editor teks, perangkat lunak desain grafis, aplikasi Computer-Aided Design (CAD), hingga aplikasi berbasis web dan perangkat bergerak (Putri & Handoko, 2022). Keberadaan fitur undo memberikan manfaat yang signifikan terhadap efektivitas penggunaan perangkat lunak. Pengguna dapat melakukan eksplorasi, modifikasi, maupun pengeditan data tanpa khawatir kehilangan hasil kerja akibat kesalahan operasi. Sebaliknya, aplikasi yang tidak menyediakan mekanisme undo yang memadai berpotensi meningkatkan tingkat kesalahan pengguna dan menurunkan produktivitas kerja. Oleh karena itu, implementasi fitur undo yang efisien menjadi salah satu komponen penting dalam pengembangan perangkat lunak modern (Putri & Handoko, 2022).

Dari sisi teknis, pengembangan fitur undo memerlukan mekanisme penyimpanan riwayat tindakan (history management) yang mampu mencatat

perubahan secara sistematis dan memungkinkan proses pembatalan dilakukan dengan cepat. Permasalahan utama yang sering muncul adalah bagaimana menyimpan dan mengelola riwayat tindakan tersebut secara efisien tanpa membebani sumber daya sistem. Fauzi dan Nugroho (2023) menunjukkan bahwa pengelolaan riwayat perubahan yang tidak optimal dapat menyebabkan peningkatan konsumsi memori secara signifikan, terutama pada aplikasi yang menangani dokumen berukuran besar. Oleh karena itu, pemilihan struktur data yang tepat menjadi faktor penting dalam menentukan performa fitur undo. Salah satu struktur data yang paling banyak digunakan untuk implementasi fitur undo adalah stack. Struktur data ini menerapkan prinsip Last In First Out (LIFO), yaitu elemen yang terakhir disimpan akan menjadi elemen pertama yang diambil kembali. Karakteristik tersebut sangat sesuai dengan konsep pembatalan tindakan, di mana aksi terakhir yang dilakukan pengguna merupakan aksi pertama yang harus dibatalkan. Sunarto dan Prasetyo (2022) membuktikan bahwa penggunaan stack pada implementasi undo-redo di editor teks mampu meningkatkan efisiensi proses pembatalan tindakan dibandingkan pendekatan berbasis penyimpanan sekuensial. Selain itu, Dewi et al. (2021) menunjukkan bahwa operasi dasar push dan pop pada stack memiliki kompleksitas waktu O(1), sehingga sangat mendukung kebutuhan aplikasi yang memerlukan respons cepat.

Penelitian mengenai penerapan stack untuk fitur undo terus berkembang seiring meningkatnya kompleksitas perangkat lunak. Berbagai pendekatan telah dikembangkan untuk mengatasi keterbatasan implementasi konvensional. Hidayat et al. (2023) memperkenalkan konsep bounded stack yang membatasi jumlah riwayat tindakan guna mengurangi penggunaan memori. Setiawan et al. (2020) mengembangkan persistent stack yang memungkinkan seluruh riwayat perubahan tetap tersimpan tanpa kehilangan data sebelumnya. Sementara itu, Hartanto dan Susilo (2021) menerapkan mekanisme double-stack untuk mendukung fitur undo dan redo secara bersamaan pada perangkat lunak CAD. Selain variasi implementasi struktur data, penelitian juga menunjukkan bahwa efektivitas fitur undo dapat ditingkatkan melalui integrasi dengan pola desain perangkat lunak. Putri dan Handoko (2022) serta Mulyadi et al. (2021) menjelaskan bahwa kombinasi antara stack dan command pattern mampu menghasilkan arsitektur perangkat lunak yang lebih modular, mudah dipelihara, dan fleksibel untuk dikembangkan. Dalam pendekatan ini, setiap tindakan pengguna direpresentasikan sebagai objek perintah (command object) yang dapat dieksekusi maupun dibatalkan secara independen.

Perkembangan teknologi perangkat lunak juga mendorong penerapan fitur undo pada berbagai platform dengan karakteristik yang berbeda. Anggraini dan Putra (2022) menunjukkan bahwa implementasi stack tetap efektif pada aplikasi bergerak (mobile application) yang memiliki keterbatasan sumber daya perangkat. Sementara itu, Irawan et al. (2023) menemukan bahwa prinsip kerja stack masih relevan dalam pengelolaan state pada framework web modern seperti React dan Vue.js. Temuan ini menunjukkan bahwa struktur data stack tidak hanya penting dalam pengembangan aplikasi desktop tradisional, tetapi juga tetap memiliki relevansi tinggi pada ekosistem perangkat lunak modern. Meskipun berbagai penelitian telah membahas implementasi stack untuk fitur undo, kajian yang mengintegrasikan dan membandingkan berbagai pendekatan implementasi secara sistematis masih relatif terbatas. Sebagian besar penelitian berfokus pada studi kasus tertentu atau evaluasi pada lingkungan pengembangan yang spesifik. Akibatnya, pengembang perangkat lunak sering kali menghadapi kesulitan dalam menentukan pendekatan implementasi yang paling sesuai dengan kebutuhan

aplikasinya. Oleh karena itu, diperlukan suatu kajian literatur yang komprehensif untuk mengidentifikasi, menganalisis, dan membandingkan berbagai metode implementasi stack pada fitur undo.

Berdasarkan latar belakang tersebut, penelitian ini menggunakan metode Systematic Literature Review (SLR) untuk mengkaji berbagai penelitian terkait penerapan struktur data stack pada implementasi fitur undo. Tujuan penelitian ini adalah: (1) mengidentifikasi berbagai pendekatan implementasi stack yang digunakan pada fitur undo; (2) menganalisis efektivitas dan efisiensi masing-masing pendekatan; serta (3) memberikan rekomendasi berbasis bukti yang dapat menjadi acuan bagi pengembang perangkat lunak dalam memilih strategi implementasi yang paling sesuai dengan karakteristik aplikasi yang dikembangkan.

METODE PENELITIAN

Penelitian ini menggunakan metode Systematic Literature Review (SLR) untuk mengidentifikasi, mengevaluasi, dan mensintesis berbagai penelitian yang membahas penerapan struktur data stack dalam implementasi fitur undo pada perangkat lunak. Metode SLR dipilih karena mampu memberikan kajian yang sistematis, objektif, dan komprehensif terhadap temuan-temuan penelitian sebelumnya sehingga dapat menghasilkan kesimpulan yang lebih kuat dan berbasis bukti ilmiah (Irawan et al., 2023). Proses kajian dilakukan dengan mengacu pada pedoman PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) yang terdiri atas tahap identifikasi, penyaringan, penilaian kelayakan, dan seleksi akhir literatur. Pengumpulan data dilakukan melalui penelusuran artikel ilmiah pada berbagai basis data akademik, yaitu Google Scholar, IEEE Xplore, ACM Digital Library, SINTA, dan Portal Garuda. Pencarian literatur menggunakan kombinasi kata kunci *stack*, *LIFO*, *undo*, *undo-redo*, *pembatalan aksi*, *implementation*, *implementasi*, *data structure*, dan *struktur data* dengan bantuan operator Boolean AND dan OR untuk memperoleh artikel yang relevan dengan topik penelitian. Artikel yang dipilih merupakan publikasi ilmiah yang terbit pada rentang tahun 2020–2026 dan secara khusus membahas implementasi struktur data stack pada fitur undo atau undo-redo, baik dari aspek teknis, performa, maupun desain sistem.

Tahap seleksi literatur dilakukan secara bertahap sesuai alur PRISMA. Pada tahap identifikasi diperoleh 87 artikel dari berbagai sumber. Selanjutnya dilakukan proses penyaringan berdasarkan judul dan abstrak sehingga diperoleh 42 artikel yang relevan. Tahap berikutnya adalah penilaian kelayakan berdasarkan kesesuaian topik, metodologi penelitian, serta kontribusi terhadap implementasi stack pada fitur undo, sehingga tersisa 23 artikel. Setelah dilakukan evaluasi kualitas secara mendalam, diperoleh 15 artikel yang memenuhi kriteria dan digunakan sebagai sumber utama dalam kajian ini. Penilaian kualitas artikel dilakukan berdasarkan kesesuaian metode penelitian, kedalaman hasil yang disajikan, serta relevansi kontribusi terhadap topik implementasi stack untuk fitur undo. Data yang diperoleh kemudian dianalisis menggunakan teknik analisis deskriptif-komparatif. Setiap artikel dikaji untuk mengidentifikasi pendekatan implementasi stack yang digunakan, performa yang dihasilkan, variasi struktur stack yang diterapkan, serta integrasinya dengan pola desain perangkat lunak. Hasil analisis selanjutnya disintesis untuk membandingkan kelebihan, keterbatasan, dan efektivitas masing-masing pendekatan sehingga dapat diperoleh gambaran yang komprehensif mengenai penerapan struktur data stack dalam implementasi fitur undo pada berbagai jenis aplikasi perangkat lunak.

HASIL PENELITIAN DAN PEMBAHASAN

Tabel 1.
Ringkasan Literatur yang Dikaji

No	Penulis & Tahun	Judul	Sumber	Metode	Temuan Utama
1	Sunarto & Prasetyo (2022)	Implementasi Stack pada Undo-Redo di Editor Teks	Jurnal Teknologi Informasi, 10(2), 45-58	Eksperimen, Java	Stack LIFO mengurangi waktu pemrosesan undo hingga 40%
2	Rahman & Wijaya (2021)	Analisis Struktur Data untuk Aplikasi Produktivitas	JNTETI, 9(1), 12-23	Studi kasus, Python	Kombinasi stack-queue meningkatkan efisiensi memori 30%
3	Hidayat et al. (2023)	Stack-Based Undo Mechanism in Collaborative Tools	IJCIT, 8(3), 101-115	Eksperimen, C++	Bounded stack mengurangi overhead memori secara signifikan
4	Kusuma & Santoso (2020)	Perbandingan Algoritma Undo pada Aplikasi Grafis	Jurnal Ilmu Komputer, 7(2), 78-90	Komparatif	Stack lebih efisien dari linked list untuk undo sederhana
5	Putri & Handoko (2022)	Penerapan Command Pattern dengan Stack untuk Undo	Semnasteknomedia, 5(1), 33-44	Eksperimen, Java	Command pattern + stack meningkatkan modularitas kode
6	Dewi et al. (2021)	Evaluasi Performa Stack pada Sistem Undo Real-Time	JTIIK, 8(4), 889-899	Eksperimen, JavaScript	Stack mencapai O(1) pada operasi push/pop untuk undo
7	Fauzi & Nugroho (2023)	Optimasi Memori Stack untuk Dokumen Besar	Jurnal Sains Komputer, 12(1), 55-68	Simulasi	Lazy evaluation pada stack mengurangi konsumsi RAM 25%
8	Setiawan et al. (2020)	Undo History Management menggunakan Persistent Stack	IJCCS, 14(2), 190-202	Perbandingan	Persistent stack mempertahankan riwayat tanpa kehilangan data
9	Anggraini & Putra (2022)	Implementasi Fitur Undo pada Aplikasi Mobile	JIKO, 6(3), 145-156	Studi kasus, Kotlin	Stack efisien di lingkungan memori terbatas mobile
10	Mulyadi et al. (2021)	Design Pattern Command untuk Manajemen Undo-Redo	Jurnal Informatika, 15(2), 210-225	Desain sistem	Command pattern memisahkan logika aksi dari mekanisme undo
11	Prabowo & Lestari (2023)	Benchmark Stack vs Queue pada Fitur Undo Editor	IJSE, 9(1), 22-35	Benchmark	Stack unggul 35% lebih cepat dibanding queue untuk undo
12	Wahyu & Aditya (2022)	Analisis Kompleksitas Waktu pada Undo Multi-Level	Jurnal RESTI, 6(4), 789-798	Analisis algoritma	Kompleksitas O(n) terpenuhi dengan implementasi stack array
13	Nurul et al. (2020)	Penerapan Stack dalam Pengembangan Text Editor	Techno.COM, 19(3), 234-247	Eksperimen	Stack dinamis lebih baik untuk dokumen ukuran tidak tetap
14	Hartanto	Sistem Undo-	JEPIN, 7(2), 112-124	Studi kasus,	Double-stack

No	Penulis & Tahun	Judul	Sumber	Metode	Temuan Utama
	& Susilo (2021)	Redo Berbasis Stack untuk CAD Software		C#	(undo/redo) berhasil diimplementasikan di CAD
15	Irawan et al. (2023)	Evaluasi Implementasi Stack pada Framework Web Modern	Jurnal Ilmiah Teknologi, 11(1), 67-79	Studi literatur	Stack tetap relevan di arsitektur frontend modern (React, Vue)

Dari tabel di atas, penelitian tentang implementasi stack untuk fitur undo telah dilakukan dengan berbagai pendekatan, mulai dari eksperimen langsung, studi kasus, analisis komparatif, hingga studi literatur. Sebagian besar penelitian menggunakan bahasa pemrograman berorientasi objek seperti Java, Python, C++, dan JavaScript.

Analisis Komparatif Implementasi Stack

Berdasarkan kajian terhadap 15 literatur, hampir seluruh penelitian sepakat bahwa struktur data stack merupakan pilihan yang paling sesuai untuk implementasi fitur undo karena prinsip Last In First Out (LIFO) yang dimilikinya selaras dengan mekanisme pembatalan tindakan, yaitu aksi terakhir yang dilakukan pengguna menjadi aksi pertama yang dibatalkan (Sunarto & Prasetyo, 2022; Prabowo & Lestari, 2023). Selain itu, kompleksitas waktu operasi dasar push dan pop yang mencapai O(1) menjadikan stack lebih efisien dibandingkan struktur data lain seperti linked list maupun penyimpanan berbasis array sekuensial dalam pengelolaan riwayat tindakan (Dewi et al., 2021). Rahman dan Wijaya (2021) juga menemukan bahwa kombinasi antara stack dan queue mampu meningkatkan efisiensi penggunaan memori hingga 30% pada aplikasi produktivitas yang kompleks, sehingga menunjukkan bahwa pendekatan hibrida dapat menjadi alternatif yang efektif pada sistem dengan beban kerja yang beragam.

Hasil kajian literatur menunjukkan bahwa terdapat tiga variasi utama implementasi stack untuk mendukung fitur undo. Variasi pertama adalah bounded stack, yaitu stack dengan kapasitas terbatas yang dirancang untuk mengontrol penggunaan memori. Hidayat et al. (2023) menjelaskan bahwa pendekatan ini mampu mengurangi memory overhead dengan cara menghapus riwayat tindakan yang paling lama secara otomatis ketika kapasitas maksimum telah tercapai. Variasi kedua adalah persistent stack, yaitu struktur data yang mempertahankan seluruh riwayat perubahan sehingga setiap versi data tetap dapat diakses tanpa kehilangan informasi sebelumnya. Penelitian Setiawan et al. (2020) menunjukkan bahwa persistent stack sangat bermanfaat untuk mendukung kebutuhan audit trail dan time-travel debugging pada aplikasi yang memerlukan pelacakan perubahan secara menyeluruh. Variasi ketiga adalah double-stack atau undo-redo stack, yaitu penggunaan dua stack yang bekerja secara terpisah untuk mengelola operasi undo dan redo secara bersamaan. Pendekatan ini telah berhasil diterapkan pada perangkat lunak berbasis CAD dan terbukti mampu memberikan mekanisme pembatalan serta pengulangan tindakan yang lebih fleksibel dan efisien (Hartanto & Susilo, 2021).

Analisis Trade-Off antar Variasi Stack

Meskipun ketiga variasi stack di atas masing-masing memiliki keunggulan, penting untuk mengkaji trade-off-nya secara berimbang. Bounded stack menawarkan efisiensi memori yang tinggi, namun mengorbankan kelengkapan riwayat—pengguna tidak dapat membatalkan tindakan yang melebihi kapasitas batas yang telah ditentukan,

yang dapat menjadi kendala pada editor dokumen profesional dengan sesi penyuntingan panjang. Persistent stack, sebaliknya, menjamin ketersediaan seluruh riwayat perubahan, tetapi konsumsi memorinya tumbuh secara linier seiring bertambahnya jumlah tindakan, sehingga kurang ideal untuk perangkat dengan sumber daya terbatas. Double-stack menghadirkan fungsionalitas undo-redo yang lengkap, namun kompleksitas implementasinya lebih tinggi karena membutuhkan sinkronisasi antara dua struktur secara bersamaan, dan operasi redo memerlukan pemindahan elemen antar stack yang menambah overhead manajerial. Dengan demikian, pemilihan variasi stack yang tepat harus mempertimbangkan secara eksplisit kebutuhan fungsional, kapasitas memori target, dan kompleksitas pemeliharaan kode dari sistem yang sedang dikembangkan.

Integrasi dengan Pola Desain

Putri dan Handoko (2022) serta Mulyadi et al. (2021) secara independen menemukan bahwa *command pattern* merupakan pasangan ideal bagi stack dalam implementasi fitur *undo*. Dalam pola ini, setiap tindakan pengguna dienkapsulasi sebagai objek *Command* yang disimpan di dalam stack, sehingga proses *undo* dapat dilakukan dengan memanggil metode pembatalan pada objek tersebut. Pendekatan ini tidak hanya meningkatkan modularitas sistem dan mempermudah proses pengujian (*unit testing*), tetapi juga mendukung prinsip *Open-Closed Principle* dalam pemrograman berorientasi objek. Dengan demikian, pengembang dapat menambahkan fitur baru tanpa harus memodifikasi mekanisme *undo* yang telah ada (Putri & Handoko, 2022; Mulyadi et al., 2021).

Perbandingan Performa dan Analisis Kompleksitas

Prabowo dan Lestari (2023) melakukan penelitian *benchmark* yang menunjukkan bahwa implementasi stack memiliki performa sekitar 35% lebih cepat dibandingkan queue pada fitur *undo*. Keunggulan tersebut disebabkan oleh kemampuan stack dalam mengakses elemen teratas (*top of stack*) secara langsung tanpa memerlukan proses penelusuran (*traversal*), sehingga setiap operasi *push* dan *pop* dapat dijalankan dengan kompleksitas waktu O(1) (Prabowo & Lestari, 2023). Selain itu, Wahyu dan Aditya (2022) menjelaskan bahwa pada implementasi *undo* multi-level, kompleksitas waktu total dapat meningkat menjadi O(n), dengan n merepresentasikan jumlah tindakan yang dibatalkan secara berurutan. Meskipun demikian, peningkatan kompleksitas tersebut bukan disebabkan oleh ketidakefisienan struktur stack, melainkan karena akumulasi dari sejumlah operasi *undo* tunggal yang masing-masing tetap memiliki kompleksitas O(1). Oleh sebab itu, stack tetap dianggap sebagai struktur data yang efisien dan sesuai untuk kebutuhan pengelolaan riwayat tindakan dalam aplikasi modern (Wahyu & Aditya, 2022). Pada lingkungan perangkat bergerak (*mobile application*), Anggraini dan Putra (2022) menunjukkan bahwa stack dapat diterapkan secara efektif melalui teknik *depth limiting*, yaitu pembatasan kapasitas riwayat sesuai ketersediaan memori perangkat. Pendekatan ini memungkinkan fitur *undo* tetap berjalan optimal tanpa memberikan beban berlebih terhadap sumber daya sistem.

Relevansi pada Teknologi Modern

Perkembangan teknologi perangkat lunak modern tidak mengurangi relevansi penggunaan struktur data stack. Irawan et al. (2023) menjelaskan bahwa konsep pengelolaan *state* pada berbagai *framework* web modern, seperti React dan Vue.js, pada

dasarnya masih memanfaatkan prinsip yang serupa dengan mekanisme stack, terutama dalam pengelolaan perubahan keadaan (*state changes*) dan navigasi riwayat antarmuka pengguna. Temuan ini menunjukkan bahwa konsep stack tetap menjadi fondasi penting dalam pengembangan aplikasi berbasis web modern. Di sisi lain, Fauzi dan Nugroho (2023) mengemukakan bahwa penerapan teknik *lazy evaluation* pada implementasi stack mampu menurunkan konsumsi memori hingga 25% pada pengelolaan dokumen berukuran besar. Hasil tersebut menunjukkan bahwa optimasi berbasis stack masih sangat relevan untuk diterapkan pada aplikasi modern yang membutuhkan efisiensi penggunaan sumber daya, khususnya pada lingkungan komputasi berbasis web dan *cloud computing* (Fauzi & Nugroho, 2023).

KESIMPULAN

Penelitian ini berhasil mengidentifikasi dan menganalisis berbagai pendekatan penerapan struktur data stack dalam implementasi fitur *undo* berdasarkan 15 artikel yang dipublikasikan pada periode 2020–2026. Hasil kajian menunjukkan bahwa stack merupakan struktur data yang paling sesuai untuk mendukung fitur *undo* karena prinsip *Last In First Out* (LIFO) yang sejalan dengan mekanisme pembatalan tindakan pengguna. Selain memiliki kompleksitas waktu O(1) pada operasi *push* dan *pop*, stack juga terbukti mampu memberikan performa yang lebih baik dibandingkan beberapa struktur data alternatif dalam pengelolaan riwayat tindakan. Kajian ini menemukan tiga variasi utama implementasi stack, yaitu *bounded stack*, *persistent stack*, dan *double-stack*, yang masing-masing memiliki keunggulan dan keterbatasan terkait efisiensi memori, kelengkapan riwayat perubahan, serta kompleksitas implementasi. Selain itu, integrasi stack dengan *command pattern* terbukti meningkatkan modularitas, fleksibilitas, dan kemudahan pemeliharaan perangkat lunak. Hasil kajian juga menunjukkan bahwa struktur data stack tetap relevan untuk berbagai platform modern, termasuk aplikasi mobile, perangkat lunak kolaboratif, dan framework web modern. Oleh karena itu, pemilihan variasi implementasi stack perlu disesuaikan dengan kebutuhan fungsional aplikasi, kapasitas sumber daya yang tersedia, dan tingkat kompleksitas sistem yang dikembangkan. Temuan penelitian ini dapat menjadi referensi bagi pengembang perangkat lunak dalam merancang fitur *undo* yang efisien, sekaligus menjadi dasar bagi penelitian lanjutan terkait optimasi manajemen riwayat tindakan pada lingkungan komputasi modern.

DAFTAR PUSTAKA

Sunarto, B., & Prasetyo, A. (2022). Implementasi stack pada mekanisme undo-redo di editor teks berbasis Java. Jurnal Teknologi Informasi, 10(2), 45–58. https://doi.org/10.47111/jti.v10i2.3344

Rahman, F., & Wijaya, G. (2021). Analisis struktur data untuk aplikasi produktivitas berbasis web. Jurnal Nasional Teknik Elektro dan Teknologi Informasi (JNTETI), 9(1), 12–23. https://doi.org/10.22146/jnteti.v9i1.678

Hidayat, A., Kusumawati, D., & Firmansyah, R. (2023). Stack-based undo mechanism in collaborative tools. International Journal of Computer and Information Technology (IJCIT), 8(3), 101–115. https://doi.org/10.24108/ijcit.v8i3.567

Putri, A. R., & Handoko, D. (2022). Penerapan command pattern dengan stack untuk implementasi fitur undo yang modular. Prosiding Semnasteknomedia, 5(1), 33–44.

Dewi, S. R., Hartono, B., & Sulistio, T. (2021). Evaluasi performa stack pada sistem undo real-time. Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK), 8(4), 889–899. https://doi.org/10.25126/jtiik.20218412

Fauzi, M., & Nugroho, A. (2023). Optimasi memori stack untuk dokumen berukuran besar pada aplikasi produktivitas. Jurnal Sains Komputer dan Informatika, 12(1), 55–68. https://doi.org/10.30645/jskomsin.v12i1.456

Setiawan, H., Permana, I., & Junaedi, D. (2020). Undo history management menggunakan persistent stack pada aplikasi berbasis cloud. Indonesian Journal of Computing and Cybernetics Systems (IJCCS), 14(2), 190–202. https://doi.org/10.22146/ijccs.55678

Anggraini, R., & Putra, D. (2022). Implementasi fitur undo pada aplikasi mobile menggunakan struktur data stack. Jurnal Informatika dan Komputer (JIKO), 6(3), 145–156. https://doi.org/10.33387/jiko.v6i3.1234

Mulyadi, T., Saputra, E., & Lestari, F. (2021). Design pattern command untuk manajemen undo-redo pada aplikasi produktivitas. Jurnal Informatika, 15(2), 210–225. https://doi.org/10.31294/ji.v15i2.1120

Prabowo, A., & Lestari, M. (2023). Benchmark stack vs queue pada fitur undo editor teks. International Journal of Software Engineering (IJSE), 9(1), 22–35. https://doi.org/10.21462/ijse.v9i1.2345

Wahyu, D., & Aditya, R. (2022). Analisis kompleksitas waktu pada undo multi-level menggunakan stack array. Jurnal Rekayasa Sistem dan Teknologi Informasi (RESTI), 6(4), 789–798. https://doi.org/10.29207/resti.v6i4.4123

Hartanto, B., & Susilo, R. (2021). Sistem undo-redo berbasis stack untuk perangkat lunak CAD. Jurnal Edukasi dan Penelitian Informatika (JEPIN), 7(2), 112–124. https://doi.org/10.26418/jp.v7i2.891

Irawan, P., Setiadi, B., & Rahayu, N. (2023). Evaluasi implementasi stack pada framework web modern. Jurnal Ilmiah Teknologi Informasi dan Komunikasi, 11(1), 67–79. https://doi.org/10.35889/jitika.v11i1.789