TUGAS AKHIR

RANCANG BANGUN SISTEM *GENERATOR* ANTARMUKA PENGGUNA BERBASIS *ATOMIC DESIGN*

DIAJUKAN OLEH:

MUFID BAHAUDIN NUGROHO

22106050021

PROGRAM STUDI INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI SUNAN KALIJAGA

YOGYAKARTA

2026

LEMBAR PENGESAHAN TUGAS AKHIR

KEMENTERIAN AGAMA
UNIVERSITAS ISLAM NEGERI SUNAN KALIJAGA
FAKULTAS SAINS DAN TEKNOLOGI
Jl. Marsda Adisucipto Telp. (0274) 540971 Fax. (0274) 519739 Yogyakarta 55281

PENGESAHAN TUGAS AKHIR

Nomor : B-1359/Un.02/DST/PP.00.9/06/2026

Tugas Akhir dengan judul : Rancang Bangun Sistem Generator Antarmuka Pengguna Berbasis Atomic Design

yang dipersiapkan dan disusun oleh:

Nama : MUFID BAHAUDIN NUGROHO
Nomor Induk Mahasiswa : 22106050021
Telah diujikan pada : Kamis, 04 Juni 2026
Nilai ujian Tugas Akhir : A-

dinyatakan telah diterima oleh Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta

TIM UJIAN TUGAS AKHIR

Ketua Sidang

Muhammad Mustakim, S.T. M.T.
SIGNED

Valid ID: 6a38d31c1e3b8

Penguji I

Dr. Agung Fatwanto, S.Si., M.Kom.
SIGNED

Valid ID: 6a337c997b284

Penguji II

Ir. Muhammad Didik Rohmad Wahyudi, S.T., MT.
SIGNED

Valid ID: 6a38c1800d195

Yogyakarta, 04 Juni 2026
UIN Sunan Kalijaga
Dekan Fakultas Sains dan Teknologi

Prof. Dr. Dra. Hj. Khurul Wardati, M.Si.
SIGNED

Valid ID: 6a3b6346a8ab0

LEMBAR PERNYATAAN

SURAT PERNYATAAN KEASLIAN

Yang bertanda tangan dibawah ini:

Nama	: Mufid Bahaudin Nugroho
NIM	22106050021
Program Studi	: Informatika
Fakultas	: Sains dan Teknologi

Menyatakan dengan sesungguhnya, bahwa skripsi saya yang berjudul "Rancang Bangun Sistem Generator Antarmuka Pengguna Berbasis Atomic Design" merupakan hasil pekerjaan penulis sendiri sepanjang pengetahuan penulis, bukan duplikasi atau saduran dari karya orang lain kecuali bagian tertentu yang penulis ambil sebagai bahan acuan. Apabila terbukti pernyataan ini tidak benar, sepenuhnya menjadi tanggung jawab penulis.

Yogyakarta, 25 Juni 2026

Mufid Bahaudin Nugroho
22106050021

LEMBAR PERSETUJUAN

SURAT PERSETUJUAN SKRIPSI/TUGAS AKHIR

Hal : Persetujuan Skripsi / Tugas Akhir
Lamp : -

Kepada
Yth. Dekan Fakultas Sains dan Teknologi
UIN Sunan Kalijaga Yogyakarta
di Yogyakarta

Assalamu'alaikum wr. wb.

Setelah membaca, meneliti, memberikan petunjuk dan mengoreksi serta mengadakan perbaikan seperlunya, maka kami selaku pembimbing berpendapat bahwa skripsi Saudara:

Nama	:	Mufid Bahaudin Nugroho
NIM	:	22106050021
Judul Skripsi	:	Rancang Bangun Sistem Generator Antarmuka Pengguna Berbasis Atomic Design

sudah dapat diajukan kembali kepada Program Studi Informatika Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta sebagai salah satu syarat untuk memperoleh gelar Sarjana Strata Satu dalam Program Studi Informatika.

Dengan ini kami mengharap agar skripsi/tugas akhir Saudara tersebut di atas dapat segera dimunaqasyahkan. Atas perhatiannya kami ucapkan terima kasih.

Wassalamu'alaikum wr. wb.

Yogyakarta, 25 Juni 2026
Pembimbing

Dr. Muhammad Mustakim, S.T., M.T.
NIP. 19790331 200501 1004

INTISARI

Perkembangan industri perangkat lunak modern menuntut antarmuka pengguna (*User Interface/UI*) yang berkualitas, konsisten, dan mudah dikembangkan. Meskipun teknologi *Generative Artificial Intelligence* (AI) telah membuka peluang otomatisasi dalam pembuatan kode UI, kode yang dihasilkan seringkali tidak terstruktur, sulit dipelihara, dan tidak mengikuti pola komponen yang terstandarisasi. Penelitian ini bertujuan merancang dan membangun sistem generator antarmuka pengguna berbasis *Atomic Design* dengan memanfaatkan *Large Language Model* (LLM) sebagai mesin pembangkit kode. Sistem dikembangkan menggunakan framework Next.js yang mengintegrasikan lapisan *frontend* dan *backend* dalam satu framework terpadu, serta memanfaatkan Google Gemini API sebagai model LLM. Pendekatan Atomic Design diterapkan melalui teknik *prompt engineering* berlapis yang menanamkan instruksi hierarki *Atom, Molecule, Organism, Template,* dan *Page* secara eksplisit pada setiap prompt. Sistem mendukung tiga mode input, yaitu *Standard Prompt*, *Template Builder*, dan *Structured Form*, untuk mengakomodasi berbagai kebutuhan pengguna mulai dari eksplorasi kreatif hingga pengujian terukur. Pengujian dilakukan melalui *output test* dan *black-box testing* terhadap 12 skenario fungsional. Hasil penelitian menunjukkan bahwa sistem berhasil menghasilkan komponen UI secara otomatis dalam format HTML/Tailwind CSS maupun React JSX dengan struktur yang konsisten dan modular sesuai standar Atomic Design. Seluruh skenario pengujian *black-box* mencapai status Berhasil. Mekanisme *prompt engineering* berlapis terbukti efektif mengarahkan model AI untuk menghasilkan kode antarmuka yang selaras dengan prinsip modularitas dan konsistensi komponen.

Kata kunci: Antarmuka Pengguna, *Atomic Design*, *Generative AI*, *Large Language Model*, *Prompt Engineering*

ABSTRACT

The rapid growth of the modern software industry demands high-quality, consistent, and maintainable user interfaces (UI). Although Generative Artificial Intelligence (AI) technology has opened opportunities for automating UI code generation, the resulting code is often unstructured, difficult to maintain, and inconsistent with standardized component development patterns. This research aims to design and develop an Atomic Design-based UI generator system utilizing a Large Language Model (LLM) as its code generation engine. The system was developed using the Next.js framework, integrating frontend and backend layers within a unified framework, and utilizing the Google Gemini API as the LLM model. The Atomic Design approach was implemented through a layered prompt engineering technique that explicitly embeds the Atom, Molecule, Organism, Template, and Page hierarchy instructions into every prompt. The system supports three input modes Standard Prompt, Template Builder, and Structured Form to accommodate various user needs ranging from creative exploration to controlled academic testing. Evaluation was conducted through output testing and black-box testing across 12 functional scenarios. The results demonstrate that the system successfully generates UI components automatically in both HTML/Tailwind CSS and React JSX formats, with consistent and modular structures aligned with Atomic Design standards. All black-box testing scenarios achieved a Passed status. The layered prompt engineering mechanism proved effective in guiding the AI model to produce interface code that is consistent, modular, and compliant with Atomic Design principles.

Keywords: *User Interface, Atomic Design, Generative AI, Large Language Model, Prompt Engineering*

MOTTO

"I have not failed. I've just found 10,000 ways that won't work."

~Thomas Alva Edision~

"Sekecil apapun usaha yang sudah kamu lakukan pasti akan membuahkan hasil."

~B.J. Habibie~

HALAMAN PERSEMBAHAN

Puji syukur kehadirat Allah SWT atas dukungan dan doa, saya ucapkan rasa syukur dan terima kasih kepada :

1. Allah SWT, karena atas izin dan keberkahan Tugas Akhir ini dapat dibuat dan selesai pada waktunya.
2. Keluarga saya Bapak/Ibu/Saudara yang telah memberikan dukungan dan doa yang tiada henti.
3. Rekan-rekan saya yang selalu mendukung dan memberikan semangat untuk menghadapi Tugas Akhir ini.

KATA PENGANTAR

Alhamdulillahirabbil'alamin, dengan memanjatkan puja dan puji syukur kehadirat Allah SWT yang melimpahkan rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan Tugas Akhir ini dengan judul "Rancang Bangun Sistem Generator Antarmuka Pengguna Berbasis *Atomic Design*", dengan baik. Shalawat serta salam, semoga tercurahkan kepada baginda kita Rasulullah Muhammad SAW, beserta keluarga dan sahabatnya.

Penulisan Tugas Akhir ini sebagai salah satu syarat gelar untuk menyelesaikan Program Studi (S1) Program Studi Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Sunan Kalijaga Yogyakarta. Penulis menyadari bahwa Tugas Akhir ini dengan adanya dukungan, bantuan, bimbingan, dan nasehat dari berbagai pihak selama penyusunan Tugas Akhir ini. Oleh sebab itu, penulis mengucapkan terima kasih kepada :

1. Bapak Prof. Noorhaidi, S.Ag., M.A., M.Phil., Ph.D., selaku Rektor Universitas Islam Negeri Sunan Kalijaga Yogyakarta.
2. Ibu Prof. Dr. Dra. Hj. Khurul Wardati, M.Si. selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Sunan Kalijaga Yogyakarta.
3. Bapak Agus Mulyanto, S.Si., M.Kom. selaku Dosen Pembimbing Akademik Informatika yang selalu membimbing selama perkuliahan.
4. Bapak Dr. Muhammad Mustakim, S.T., M.T. selaku Ketua Program Studi Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Sunan Kalijaga Yogyakarta.
5. Bapak Dr. Muhammad Mustakim, S.T., M.T. selaku Dosen Pembimbing Tugas Akhir yang sabar dan selalu memberikan masukan dan saran selama menjalani perkuliahan.
6. Bapak Dr. Agung Fatwanto, S.Si., M.Kom., ASEAN Eng., dan Bapak Dr. Ir. Aulia Faqih Rifa'i, M.Kom. selaku Dosen Peminatan Perekayasa Perangkat Lunak yang telah menghadiri, mengajar, dan membimbing selama perkuliahan.
7. Keluarga Inti saya termasuk Bapak/Ibu/Saudara atas segala doa, kasih sayang, dan memberikan bantuannya.
8. Seluruh rekan-rekan angkatan Informatika 2022 Universitas Islam Negeri Sunan Kalijaga Yogyakarta dan teman-teman SMA saya yang selalu memberikan support, memberikan semangat, dan berteman bersama selama perkuliahan.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam beberapa tahun terakhir, perkembangan industri perangkat lunak telah meningkatkan kebutuhan akan antarmuka pengguna (*User Interface*/UI) yang berkualitas, konsisten, dan mudah dikembangkan. Sebagai elemen yang berkomunikasi langsung dengan pengguna, antarmuka pengguna yang konsisten menjadi komponen krusial dalam sebuah produk digital [1][2]. Kualitas antarmuka pengguna berdampak signifikan pada pengalaman pengguna (*User Experience*/UX), kemudahan penggunaan sistem, dan kualitas produk perangkat lunak secara keseluruhan [3]. Dengan demikian, proses pengembangan antarmuka pengguna yang efektif menjadi semakin penting seiring meningkatnya kompleksitas aplikasi modern.

Laporan *State of Frontend 2024* menunjukkan bahwa 75,8 persen pengembang frontend telah menggunakan AI dalam pengembangan perangkat lunak. Selain itu, penelitian tersebut menunjukkan bahwa penggunaan desain sistem dan pengembangan berbasis komponen telah menjadi kebiasaan di industri modern [4]. Meskipun demikian, penggunaan AI tersebut umumnya masih bersifat membantu dan belum menyentuh proses penerjemahan desain antarmuka menjadi struktur komponen secara otomatis dan terstandarisasi. Pengembang masih harus menerjemahkan susunan antarmuka menjadi kode program secara manual, membangun ulang komponen yang sebenarnya serupa, serta memastikan keselarasan antara tampilan dan struktur kode. Pada proyek skala menengah hingga besar, kondisi semacam ini berpotensi menurunkan produktivitas pengembang, memperpanjang waktu pengembangan, dan menyulitkan pemeliharaan kode dalam jangka panjang. Selain itu, inkonsistensi antarkomponen yang timbul akibat proses implementasi manual ini berisiko bermanifestasi sebagai *usability defect*, yaitu tampilan atau tata letak yang berubah-ubah tanpa pola yang jelas sehingga membingungkan pengguna, menurunkan kepuasan pengguna, dan memperpanjang waktu penyelesaian tugas [5].

Pengembangan *Large Language Model* (LLM) dan teknologi *Generative Artificial Intelligence* (AI) telah membuka peluang baru untuk membantu proses pengembangan perangkat lunak. McKinsey mengatakan bahwa penggunaan *Generative AI* dapat membuat pengembang lebih produktif dalam berbagai tugas pemrograman, seperti menulis kode dan menyelesaikan tugas pengembangan perangkat lunak lainnya [6]. Selain itu, penelitian yang

dilakukan oleh Dohmke menunjukkan bahwa penggunaan teknologi berbasis AI dapat meningkatkan efisiensi siklus pengembangan perangkat lunak secara signifikan [7]. Temuan tersebut menunjukkan bahwa *Generative AI* berpotensi membantu otomatisasi berbagai tugas pengembangan perangkat lunak, termasuk proses implementasi antarmuka pengguna.

Namun, ada beberapa masalah dengan menggunakan *Generative AI* untuk membuat kode antarmuka pengguna. Kode yang dibuat seringkali tidak memiliki struktur yang konsisten atau halusinasi, sulit dipelihara, dan tidak sesuai dengan pola pengembangan komponen yang terstandarisasi [8]. Akibatnya, meskipun proses pembuatan kode dapat dilakukan lebih cepat, kualitas dan konsistensi hasil belum selalu memenuhi kebutuhan pengembangan *frontend* modern. Kondisi ini menunjukkan bahwa diperlukan suatu pendekatan yang mampu mengarahkan proses pembangkitan kode agar menghasilkan komponen yang lebih terstruktur dan mudah digunakan kembali.

Atomic Design adalah teknik yang banyak digunakan untuk menjaga konsistensi pengembangan antarmuka pengguna [9]. Ini memungkinkan pembuatan antarmuka pengguna secara modular dengan membagi komponen ke dalam beberapa tingkatan yang terstruktur, yang memudahkan pengembangan dan penggunaan ulang komponen. Dengan karakteristik ini, *Atomic Design* dapat digunakan sebagai kerangka untuk proses otomatisasi pemodelan antarmuka pengguna.

Dengan mempertimbangkan kondisi ini, suatu sistem yang mampu memanfaatkan teknologi *Generative AI* untuk membantu proses pembuatan komponen antarmuka pengguna secara otomatis sambil mempertahankan konsistensi struktur komponen diperlukan. Oleh karena itu, penelitian ini menyarankan untuk membangun sistem generator antarmuka pengguna berbasis *Atomic Design* dengan memanfaatkan LLM sebagai mesin pembangkit kode. Diharapkan bahwa sistem ini akan mempercepat proses pengembangan frontend, mengurangi jumlah pekerjaan yang berulang, dan menghasilkan komponen antarmuka pengguna yang lebih terorganisir dan konsisten, yang akan memenuhi kebutuhan pengembangan perangkat lunak modern.

1.2 Rumusan Masalah

1. Bagaimana merancang dan membangun sistem *generator* kode UI secara otomatis?
2. Bagaimana menerapkan *Atomic Design* dalam sistem *generator* untuk menghasilkan struktur komponen yang konsisten dan modular?

1.3 Batasan Masalah

1. Sistem menghasilkan komponen UI *frontend* dengan kerangka kerja Tailwind CSS dan React.

2. Model LLM menggunakan Gemini API sebagai alat bantu pemroses generasi kode dan tidak melakukan pelatihan atau perbandingan model.
3. Sistem hanya menghasilkan komponen UI fungsional secara individual, bukan sekadar aplikasi lengkap *end-to-end.*
4. Komponen yang dihasilkan mengacu pada kelima tingkat *Atomic Design,* ialah *Atom, Molecule, Organism, Template,* dan *Page.*
5. Tidak membahas terkait kinerja komputasi model AI ataupun aspek keamanan sistem yang mendalam.

1.4 Tujuan

1. Merancang dan membangun sistem *generator* kode UI secara otomatis.
2. Menerapkan *Atomic Design* untuk menghasilkan struktur komponen UI secara konsisten dan modular.

1.5 Manfaat

1. Menjadi alat bantu bagi desainer dan pengembang untuk mempercepat proses pembuatan komponen UI yang konsisten, modular, dan siap digunakan.
2. Memberi kontribusi referensi bagi penelitian lanjut dalam pengembangan sistem otomasi UI berbasis *Atomic Design* dan pemanfaatan teknologi AI sebagai alat bantu pengembangan perangkat lunak.

BAB V
KESIMPULAN & SARAN

5.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian yang telah dilakukan secara empiris, penelitian ini berhasil menyelesaikan kedua rumusan masalah yang telah dirumuskan secara sistematis. Berikut ini merupakan kesimpulan utama yang dapat disimpulkan dari penelitian ini.

1. Sistem generator UI berbasis *Atomic Design* telah berhasil dirancang dan dikembangkan dengan memanfaatkan arsitektur Next.js yang mengintegrasikan lapisan *frontend* dan *backend* dalam satu framework terpadu. Sistem tersebut mampu menghasilkan kode komponen UI secara otomatis dalam format HTML/Tailwind CSS maupun React JSX melalui kolaborasi dengan model LLM Google Gemini sebagai penggerak utama proses generasi.
2. Penerapan *Atomic Design* dalam sistem telah berhasil melalui dua pendekatan utama. Pertama, melalui teknik *prompt engineering* berlapis yang menanamkan instruksi hierarki *Atomic Design* secara eksplisit pada setiap *prompt* yang dikirimkan ke model AI, sehingga output yang dihasilkan condong mengikuti struktur modular yang sesuai.
3. Hasil pengujian *Black-box testing* terhadap 12 skenario fungsional menunjukkan pencapaian status Berhasil (Sesuai) secara keseluruhan.
4. Mekanisme *prompt engineering* berlapis yang diterapkan dalam sistem terbukti efektif mengarahkan model AI untuk menghasilkan kode antarmuka yang konsisten, modular, dan selaras dengan standar *Atomic Design*. Perbedaan ketiga *mode input* menyajikan gradasi kontrol dari probabilistik (*Standard Prompt*) hingga deterministik (*Structured Form*), sehingga sistem mampu menampung beragam kebutuhan pengguna, mulai dari eksplorasi kreatif hingga pengujian akademis yang terukur.

5.2 Saran

Walaupun sistem yang dikembangkan telah mencapai seluruh tujuan penelitian secara empiris, terdapat sejumlah keterbatasan yang dapat dijadikan dasar pengembangan pada studi lanjutan. Berikut merupakan saran yang layak dipertimbangkan untuk penyempurnaan dan ekspansi sistem pada masa depan.

1. Kapabilitas sistem dalam menginterpretasikan *mockup* antarmuka abstrak masih menunjukkan keterbatasan akurasi dalam mentranslasikan jarak dan tata letak elemen ke kelas Tailwind/React yang tepat. Studi lanjutan dapat mengeksplorasi model *multimodal* lebih modern atau penambahan pra-pemrosesan gambar guna meningkatkan ketepatan konversi desain ke kode.
2. Pengujian penelitian ini terpusat pada validasi fungsional dan teknis sistem. Studi lanjutan disarankan melibatkan desainer dan pengembang *frontend* sebagai responden untuk data kuantitatif mengenai efisiensi, kemudahan, dan kepuasan pengguna, seperti *System Usability Scale* dan *User Acceptance Testing*.
3. Modul *getAtomicType()* saat ini menggunakan berbasis heuristik aturan, rentan terhadap struktur kode non-konvesional. Studi lanjutan dapat menerapkan model klasifikasi *machine learning* untuk akurasi dan ketahanan deteksi secara otomatis.

DAFTAR PUSTAKA

[1] S. R. Yuniawati, "Perbedaan UI/UX dan Pentingnya dalam Desain Digital," *Dicoding*, Oct. 21, 2024. [Online]. Available: https://www.dicoding.com/blog/perbedaan-ui-ux-dan-pentingnya-dalam-desain-digital/

[2] S. Kurnia and N. Nawaningtyas, "Analisis Interaksi Pengguna dalam Desain User Interface dan User Experience yang Lebih Baik Menggunakan Metode Heuristik," *J. Tek. Mesin, Ind., Elektro Inform.*, vol. 3, no. 4, pp. 113–119, 2024, doi: 10.55606/jtmei.v3i4.4433.

[3] E. F. Yehdeya, C. H. Primasari, T. A. P. Sidhi, Y. P. Wibisono, D. B. Setyohadi, and M. Cininta, "Analisis User Interface (UI) dan User Experience (UX) Sudut Elevasi Pemukul Gamelan Metaverse Virtual Reality Menggunakan User Centered Design (UCD)," *JIKO (J. Inform. Komput.)*, vol. 7, no. 1, pp. 137–146, 2023, doi: 10.26798/jiko.v7i1.757.

[4] The Software House, "State of *Frontend* 2024," TSH.io, 2024. [Online]. Available: https://tsh.io/state-of-*Frontend*

[5] BINUS University, "Usability Defects," BINUS Bekasi, Jul. 2025. [Online]. Available: https://binus.ac.id/bekasi/2025/07/usability-defects/

[6] McKinsey & Company, "Unleashing Developer Productivity with *Generative AI*," McKinsey Digital, 2023. [Online]. Available: https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/unleashing-developer-productivity-with-generative-ai

[7] T. Dohmke, M. Iansiti, and G. Richards, "Sea Change in Software Development: Economic and Productivity Analysis of the AI-Powered Developer Lifecycle," arXiv preprint, arXiv:2306.15033, 2023. [Online]. Available: https://arxiv.org/abs/2306.15033

[8] F. Liu, Y. Liu, L. Shi, Z. Yang, L. Zhang, X. Lian, Z. Li, and Y. Ma, "Beyond Functional Correctness: Exploring Hallucinations in LLM-Generated Code," *arXiv preprint*, arXiv:2404.00971, 2026. [Online]. Available: https://doi.org/10.48550/arXiv.2404.00971

[9] P. Rohles, "Atomic Design Systems: Why the Labels Don't Matter," *Qt*, Nov. 26, 2025. [Online]. Available: https://www.qt.io/software-insights/atomic-design-systems-why-the-labels-dont-matter

[10] Y. Chen, S. Ding, Y. Zhang, W. Chen, J. Du, L. Sun, and L. Chen, "DesignCoder: Hierarchy-Aware and Self-Correcting UI Code Generation with Large Language Models," 2025. [Online]. Available: http://arxiv.org/abs/2506.13663

[11] Y. Gui et al., "UICopilot: Automating UI Synthesis via Hierarchical Code Generation from Webpage Designs," in Proc. ACM Web Conf. (WWW 2025), pp. 1846–1855, doi: 10.1145/3696410.3714891.

[12] Y. Jiang, Y. Zheng, Y. Wan, J. Han, Q. Wang, M. R. Lyu, and X. Yue, "ScreenCoder: Advancing Visual-to-Code Generation for Front-End Automation via Modular Multimodal Agents," 2025. [Online]. Available: http://arxiv.org/abs/2507.22827

[13] F. Wu, C. Gao, S. Li, X. Wen, and Q. Liao, "MLLM-Based UI2Code Automation Guided by UI Layout Information," *Proc. ACM Softw. Eng.*, vol. 2, no. ISSTA, 2025, doi: 10.1145/3728925.

[14] C. Si, Y. Zhang, R. Li, Z. Yang, R. Liu, and D. Yang, "Design2Code: Benchmarking Multimodal Code Generation for Automated Front-End Engineering," *arXiv preprint*, arXiv:2403.03163, 2025. [Online]. Available: https://doi.org/10.48550/arXiv.2403.03163

[15] V. Saravanan, S. Kavitha, S. Ravi, A. Seetha, Ch. Rambabu, and T. V. R. Kanth, "*Generative AI* in Software Engineering: Revolutionizing Code Generation and Debugging," Int. J. Comput. Exp. Sci. Eng., vol. 11, no. 2, pp. 2908–2917, 2025, doi: 10.22399/ijcesen.1718.

[16] M. M. Yoshananda, B. I. Nugroho, and Z. Arif, "Analisis Efektivitas Penggunaan AI dalam Desain UI/UX dengan Figma," RIGGS: J. Artif. Intell. Digit. Bus., vol. 4, no. 3, pp. 6256–6268, 2025, https://doi.org/10.31004/riggs.v4i3.2906

[17] M. Y. Matra, T. F. Kusumasari, dan F. M. Al-Anshary, "Redesign User Interface Website Portal Menggunakan Metode Atomic Design (Studi Kasus: Badan Pengawas Obat dan Makanan)," JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika), vol. 8, no. 2, pp. 500–513, 2023, https://doi.org/10.29100/jipi.v8i2.350

[18] N. P. D. Anggreni, I. N. A. Putra, and I. K. R. Janardana, "Evaluasi dan Perancangan Ulang Antarmuka Pengguna pada Fitur Live Shopee Menggunakan Metode User Experience Questionnaire (UEQ)," *J. Inform. Tek. Elektro Terapan*, vol. 13, no. 2, 2025, doi: 10.23960/jitet.v13i2.6427.

[19] DKV Binus, "UI/UX Designer in 2026: Designing Not Just Screens, But Sound and Gesture Experiences," *Binus University*, Apr. 15, 2026. [Online]. Available: https://binus.ac.id/bandung/dkv/2026/04/15/ui-ux-designer-in-2026-designing-not-just-screens-but-sound-and-gesture-experiences/

[20] K. Abertho, E. Dwinata, A. Gustho, R. Prasetyo, and W. Jayanti, "Analisis Pengaruh User Interface Terhadap Penggunaan Aplikasi Pembelajaran," *J. Teknol. Sist. Inf.*, vol. 6, no. 1, pp. 169–177, Apr. 2025, doi: 10.35957/jtsi.v6i1.10542.

[21] R. Y. Kasenda, J. O. Tenda, E. W. R. Iman, J. A. M. Manantung, Z. J. S. Moekari, and M. C. Pantas, "The Role and Evolution of Frontend Developers in the Software Development Industry," *J. Syntax Admiration*, vol. 5, no. 11, Dec. 2024, doi: 10.46799/jsa.v5i11.1852.

[22] A. Nawangwulan, "Kenali Fungsi dan Daftar Bahasa Pemrograman Frontend Development," *Kelas.work*, Jul. 14, 2023. [Online]. Available: https://kelas.work/blogs/kenali-fungsi-dan-daftar-bahasa-pemrograman-frontend-development

[23] Buildwith Angga, "Menggunakan Tailwind CSS pada ReactJS," Mar. 30, 2023. [Online]. Available: https://buildwithangga.com/tips/menggunakan-tailwindcss-pada-reactjs

[24] A. A. O'rinboev, "Analyzing the Efficiency and Performance Optimization Techniques of React.js in Modern Web Development," *ZDIT*, vol. 2, no. 24, pp. 54–57, 2023, doi: 10.5281/zenodo.8339916.

[25] "Virtual DOM dan Internal," *React.js*. [Online]. Available: https://id.legacy.reactjs.org/docs/faq-internals.html

[26] E. Ramadhan, K. Prihandani, and A. Voutama, "Penerapan Metode *Agile* pada Development Aplikasi Pengelolaan Data Magang Berbasis Web Menggunakan Framework Laravel," J. Ilm. Wahana Pendidik., vol. 9, no. 7, pp. 144–154, 2023, doi: 10.5281/zenodo.7812416.

[27] B. Frost, *Atomic Design*. Pittsburgh, PA, USA: Brad Frost, 2016. [Online]. Available: https://atomicdesign.bradfrost.com/

[28] R. A. Kurniawan, "Pembuatan Design System Menggunakan Pendekatan *Atomic Design* dan A/B Testing," Jurnal Teknologi Sistem Informasi Bisnis, vol. 6, no. 3, pp. 543–549, 2024, doi: 10.47233/jteksis.v6i3.1346.

[29] "*Atomic Design*," Primakara. [Online]. Available: https://primakara.ac.id/blog/info-teknologi/atomic-design

[30] "*Atomic Design*," BINUS Digital, Dec. 2, 2019. [Online]. Available: https://binus.ac.id/binus-digital/2019/12/02/atomic-design/

[31] "*Atomic Design*," BINUS SIS, Mar. 26, 2021. [Online]. Available: https://sis.binus.ac.id/2021/03/26/atomic-design/

[32] Y. Lamine and J. Cheng, "Understanding and Supporting the Design Systems Practice," Empirical Softw. Eng., vol. 27, no. 6, 2022, doi: 10.1007/s10664-022-10181-y.

[33] R. Setiawan, M. Asfi, A. Sevtiana, S. Pranata, and W. E. Septian, "Design System pada Perancangan Antarmuka Perangkat," vol. 9, no. 1, pp. 56–64, 2023. [Online]. Available: https://journal.nurulfikri.ac.id/index.php/JTT/article/view/619

[34] National Institute of Standards and Technology, "Artificial Intelligence Risk Management Framework (AI RMF 1.0)," NIST AI 100-1, 2023, doi: 10.6028/NIST.AI.100-1.

[35] M. Salvagno *et al.*, "The State of Artificial Intelligence in Medical Research: A Survey of Corresponding Authors from Top Medical Journals," *PLoS ONE*, vol. 19, no. 8, p. e0309208, 2024, doi: 10.1371/journal.pone.0309208.

[36] S. Feuerriegel, J. Hartmann, C. Janiesch *et al.*, "*Generative AI*," *Bus. Inf. Syst. Eng.*, vol. 66, pp. 111–126, 2024, doi: 10.1007/s12599-023-00834-7.

[37] C. Ebert and P. Louridas, "*Generative AI* for Software Practitioners," *IEEE Softw.*, vol. 40, no. 4, pp. 30–38, Jul. 2023, doi: 10.1109/MS.2023.3265877.

[38] S. Minaee, T. Mikolov, N. Nikzad, M. A. Chenaghlu, R. Socher, X. Amatriain, and J. Gao, "Large Language Models: A Survey," *arXiv preprint*, arXiv:2402.06196, 2024. [Online]. Available: https://arxiv.org/abs/2402.06196

[39] W. X. Zhao *et al.*, "A Survey of Large Language Models," *arXiv preprint*, arXiv:2303.18223, 2023. [Online]. Available: https://arxiv.org/abs/2303.18223

[40] E. Nijkamp et al., "Codegen: An Open Large Language Model for Code With Multi-Turn Program Synthesis," in Proc. 11th Int. Conf. Learn. Represent. (ICLR 2023), pp. 1–25, 2023.

[41] Gemini Team, Google, "Gemini: A Family of Highly Capable Multimodal Models," *arXiv preprint*, arXiv:2312.11805, 2023. [Online]. Available: https://arxiv.org/abs/2312.11805

[42] Gemini Team, Google, "Gemini 1.5: Unlocking Multimodal Understanding Across Millions of Tokens of Context," *arXiv preprint*, arXiv:2403.05530, 2024. [Online]. Available: https://arxiv.org/abs/2403.05530

[43] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, *Prompt*, and Predict: A Systematic Survey of *Prompt*ing Methods in Natural Language Processing," ACM Comput. Surv., vol. 55, no. 9, 2023, doi: 10.1145/3560815.

[44] J. White et al., "A *Prompt* Pattern Catalog to Enhance *Prompt engineering* with ChatGPT," 2023. [Online]. Available: http://arxiv.org/abs/2302.11382

[45] P. Sahoo, A. K. Singh, S. Saha, V. Jain, S. Mondal, and A. Chadha, "A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications," *arXiv preprint*, arXiv:2402.07927, 2024. [Online]. Available: https://arxiv.org/abs/2402.07927

[46] A. P. Setiany, D. Noviyanto, M. Irfansyahfalah, S. Aisah, A. Saifudin, and I. Kusyadi, "Penggunaan Metode System Development Life Cycle (SDLC) dalam Analisis dan Perancangan Sistem Informasi Penerimaan Kas Sekolah," *J. Teknol. Sist. Inf. Apl.*, vol. 4, no. 3, pp. 179–186, 2021. [Online]. Available: https://openjournal.unpam.ac.id/index.php/JTSI/article/view/11992

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

[48] H. Flora, "A Systematic Study on *Agile* Software Development Methodologies and Practices," *Int. J. Comput. Sci. Inf. Technol.*, vol. 5, pp. 3626–3637, 2014.

[49] D. Mishra and Y. J. Alzoubi, "Structured Software Development versus *Agile* Software Development: A Comparative Analysis," *Int. J. Syst. Assurance Eng. Manag.*, vol. 14, pp. 1504–1526, 2023, doi: 10.1007/s13198-022-01783-y.

[50] S. W. Ramdany, S. A. Kaidar, B. Aguchino, C. Amelia, and A. Putri, "Penerapan UML *Class Diagram* dalam Perancangan Sistem Informasi Perpustakaan Berbasis Web," vol. 5, no. 1, 2024.

[51] I. Fahrozi, A. F. H. Pratama, Y. Nuraeni, and R. P. Juniar, "Pengujian Black Box Testing pada Aplikasi Action & Strategy Berbasis Android dengan Teknologi Phonegap," *OKTAL*, vol. 2, no. 5, pp. 1347–1354, Jun. 2023.

[52] U. Hanifah, R. Alit, and S. Sugiarto, "Penggunaan Metode Black Box pada Pengujian Sistem Informasi Surat Keluar Masuk," *SCAN – J. Teknol. Inf. Komun.*, vol. 11, no. 2, pp. 33–40, 2016. [Online]. Available: http://ejournal.upnjatim.ac.id/index.php/scan/article/view/643

[53] "WCAG Checker," Friendly Captcha. [Online]. Available: https://friendlycaptcha.com/wcag-checker/

[54] Ernita, "Zero-Shot Prompting: Masa Depan NLP Tanpa Pelatihan Khusus," Nevacloud, Sep. 21, 2024. [Online]. Available: https://nevacloud.com/blog/zero-shot-prompting-masa-depan-nlp-tanpa-pelatihan-khusus/

[55] W. Zhang and J. J. Xia, "Structured-Prompt-Driven Development (SPDD)," Martin Fowler, Apr. 28, 2026. [Online]. Available: https://martinfowler.com/articles/structured-prompt-driven/

[56] "Strict Mode," Ajv JSON Schema Validator. [Online]. Available: https://ajv.js.org/strict-mode.html