Iterative prompting sebagai model pengembangan aplikasi keuangan koperasi berbasis AI

Vincenzhea Chrissa Siwy[1], Stephani Inggrit Swastini[2], Ridwan Sanjaya[3], Andre Kurniawan Pamudji[4]
Universitas Katolik Soegijapranata, Jl. Pawiyatan Luhur Bendan Duwur Sel.IV No.I, Bendan Duwur, Kec. Gajahmungkur, Kota Semarang
Email: 24n10033@student.unika.ac.id

Received 16 August 2025; Revised 27 August 2025; Accepted for publication 27 August 2025; Published 26 September 2025

Abstract *— The development of Artificial Intelligence (AI), particularly Large Language Models, has opened up new opportunities in software development, including in the financial application development sector. However, the use of AI to generate code is often hampered by errors, inconsistencies, and malfunctions due to unclear instructions. This study aims to test the effectiveness of iterative prompting in generating more accurate code, especially for cooperative financial applications. The methods used were literature review, code generation with GPT-4, and functional testing using the Black Box Testing method. A series of structured prompts were designed to build application features in stages, then implemented using Visual Studio Code and integrated with a MySQL database. The test results showed that iterative prompting was able to increase the success of application functions and improve code order compared to using lazy prompting. In conclusion, planned prompt engineering is key to optimally utilizing AI, especially for organizations with limited technological resources.*

Keywords *— Black Box Testing, GPT-4, Large Language Models, MySQL, Visual Studio Code*

Abstrak — Perkembangan *Artificial Intelligence (AI)*, khususnya *Large Language Models* membuka peluang baru dalam pengembangan perangkat lunak, termasuk di sektor pembuatan aplikasi keuangan. Namun, penggunaan AI untuk menghasilkan kode seringkali terkendala error, inkonsistensi, dan malfungsi akibat instruksi yang kurang jelas. Penelitian ini bertujuan menguji efektivitas *iterative prompting* dalam menghasilkan kode yang lebih akurat, terutama untuk aplikasi keuangan koperasi. Metode yang digunakan adalah studi literatur, pembuatan kode dengan GPT-4, dan pengujian fungsional menggunakan metode *Black Box Testing*. Serangkaian *prompt* terstruktur dirancang untuk membangun fitur aplikasi secara bertahap, kemudian diimplementasikan menggunakan *Visual Studio Code* dan diintegrasikan dengan basis data *MySQL*. Hasil pengujian menunjukkan bahwa *iterative prompting* mampu meningkatkan keberhasilan fungsi aplikasi dan memperbaiki keteraturan kode dibanding menggunakan *lazy prompting*. Kesimpulannya, *prompt engineering* yang terencana menjadi kunci pemanfaatan AI secara optimal, khususnya bagi organisasi dengan keterbatasan sumber daya teknologi.

Kata Kunci — *Black Box Testing, GPT-4, Large Language Models, MySQL , Visual Studio Code*

PENDAHULUAN

Perkembangan *Artificial Intelligence (AI)* generatif, khususnya model bahasa besar atau biasa disebut *Large Language Models (LLM)* seperti *GPT -4* telah membawa perubahan signifikan dalam berbagai bidang, terutama pada proses pengembangan perangkat lunak[1]. Baik para ahli maupun pemula menggunakan *LLM* yang mampu menghasilkan kode secara otomatis sesuai arahan yang diminta untuk mempermudah dan mempercepat pekerjaan, serta menurunkan persentase kesalahan[2]. Meski demikian, berbagai studi menunjukkan bahwa hasil kode yang dihasilkan oleh AI masih memiliki kelemahan mendasar seperti bug, celah keamanan, dan ketidaksesuaian dengan konteks sehingga justru seringkali menambah masalah dalam proses kerja[3]. Hal ini dapat menjadi masalah serius ketika *AI* digunakan untuk mengembangkan aplikasi di sektor yang rawan dan sensitif seperti aplikasi keuangan, dimana integritas data dan fungsionalitas sistem sangat krusial[4].

Dalam praktiknya di lapangan, penggunaan *AI* untuk menulis kode dapat dilakukan dengan pendekatan *prompt* yang berbeda. Yang paling umum adalah *prompt* yang terlalu pendek dan tidak terstruktur, dikenal juga dengan sebutan *lazy prompting*, dimana pengguna hanya memberikan arahan singkat yang tidak terlalu jelas sehingga sulit dipahami oleh sistem dan mengakibatkan hasil tidak memuaskan[5]. Sebaliknya, *prompt* yang baik adalah yang dijelaskan secara detil dan terstruktur, namun tidak menggunakan terlalu banyak token[6]. Ini menunjukkan adanya celah yang harus dijembatani agar potensi *AI* dapat benar-benar dimanfaatkan

dengan optimal guna mendapatkan hasil berkualitas. Pendekatan yang lebih terencana adalah *iterative prompting*, yaitu proses pemberian instruksi secara bertahap untuk menyempurnakan kode berdasarkan umpan balik sebelumnya[7]. Dalam konteks ini, multi-step prompting merujuk pada pembagian tugas menjadi beberapa langkah terstruktur (prompt chaining), sehingga model diarahkan secara sistematis menuju hasil akhir, sedangkan iterative prompting merupakan pendekatan yang menekankan siklus evaluasi dan revisi prompt secara berulang agar respons yang dihasilkan semakin akurat dan konsisten[8], [9]. Kombinasi keduanya penting karena multi-step prompting memastikan struktur penyelesaian masalah, sementara iterative memberi ruang perbaikan di tiap tahap yang membuat kualitas kode lebih terjamin terutama pada aplikasi keuangan yang sensitif.

Dari penelitian yang sudah ada, mayoritas studi mengenai *AI* lebih berfokus pada perbandingan model atau pada aspek performa komputasi semata[10]. Masih sedikit kajian yang menelaah secara sistematis bagaimana variasi struktur prompt, terutama yang menggunakan pendekatan bertahap akan mempengaruhi kualitas kode yang dihasilkan[11].

Konteks ini semakin mendesak mengingat koperasi di Indonesia masih menghadapi tantangan besar dalam digitalisasi sistemnya[12]. Banyak koperasi khususnya di daerah, belum memiliki sumber daya pengembang perangkat lunak yang memadai, sehingga *AI* dapat menjadi solusi sebagai jembatan yang mempermudah proses transformasi digital ini[13]. Namun, tanpa pendekatan *prompt engineering* yang tepat, penggunaan *AI* justru berpotensi memperburuk keadaan, seperti kesalahan kode maupun kebocoran data[14].

Penelitian ini bertujuan untuk menguji efektivitas *iterative prompting* dimana kode dirancang satu per satu dan diperiksa lalu diperbaiki hingga sempurna dalam menghasilkan kode untuk aplikasi keuangan koperasi. Pendekatan ini yakni memberikan instruksi secara bertahap dengan terlebih dahulu menetapkan konteks, batasan, dan detail yang jelas. Setelahnya akan dilakukan evaluasi metode *BlackBox Testing*, pengujian yang menilai fungsi program dari sisi *input-output* tanpa mengakses logika internal kode[15], [16], [17].

Kontribusi utama studi ini ada pada dua aspek. Pertama merancang dan menguji berbagai template *prompt* bertahap yang relevan dengan domain koperasi. Kedua, menilai dampak teknik pendekatan terhadap kualitas kode. Dengan fokus ini, diharapkan penelitian dapat memberikan solusi nyata bagi pengembangan sistem keuangan koperasi di Indonesia.

METODE PENELITIAN

Penelitian ini menggunakan pendekatan kualitatif deskriptif yang berfokus pada optimalisasi prompt engineering untuk menghasilkan kode melalui. Metode penelitian menggabungkan studi literatur, eksperimen pembuatan kode menggunakan *AI*, dan pengujian fungsional dalam pengembangannya dengan metode *Black Box Testing*. Metode ini dipilih karena efisien untuk menilai apakah seluruh fitur aplikasi berjalan sebagaimana mestinya tanpa perlu memeriksa struktur internal kode).

Alat dan Bahan Penelitian: Menggunakan 2 jenis perangkat
Perangkat keras:
Satu unit laptop untuk menjalankan seluruh proses

Perangkat lunak:
1. *Chat GPT (GPT -4.0)* sebagai *AI* yang digunakan untuk *code generator*
2. *Visual Code Studio* sebagai *Integrated Development Environment*
3. *My SQL* dan *PHPMyAdmin* sebagai sistem basis data

Bahan penelitian:
1. Serangkaian *prompt* terstruktur untuk membuat aplikasi dari awal hingga laporan akhir

Sampel Data Penelitian: Hasil Kode AI dan Aplikasi Jadi
Berupa kode yang dihasilkan oleh *AI* dan modul aplikasi koperasi akhir yang tercipta berdasarkan *prompt* yang telah diberikan ChatGPT.

Data penelitian meliputi:
1. *Prompt* pertama
2. Contoh *Iterative Prompting*
3. Hasil uji fungsi menggunakan metode

Tahapan Penelitian: Sebanyak Lima Tahap
Penelitian dilakukan secara runtun sebagai berikut:
1. Studi Literatur

Mengumpulkan dan membandingkan referensi terkait *AI* dalam pembuatan kode, berbagai teknik *prompt engineering,* dan konsep *Black Box Testing* dari berbagai sumber kredible seperti jurnal, prosiding, dan publikasi relevan.

2. Perancangan *Prompt* dan Pembuatan Kode
   Membuat serangkaian prompt terstruktur dan beruntun untuk setiap fitur aplikasi hingga benar. *Prompt* terus diuji dan disesuaikan agar menghasilkan kode yang optimal.
3. Integrasi Kode
   Kode yang sudah dihasilkan untuk prompt yang sudah diberikan dieksekusi untuk menghasilkan aplikasi menggunakan *Visual Studio Code* dan dihubungkan ke basis data *MySQL* melalui *PHPMyAdmin.*
4. *Functional Black Box Testing*
   Semua fitur diuji berdasarkan input dan output yang diberikan. Hasil dicatat dalam tabel pengujian dengan status sukses atau gagal.
5. Analisis Hasil
   Data dianalisis secara deskriptif untuk melihat tingkat keberhasilan fitur, jumlah revisi prompt, kualitas dokumentasi kode, dan konsistensi struktur kode.

Teknik Analisis dan Parameter Penelitian: Keberhasilan Kode yang Dihasilkan

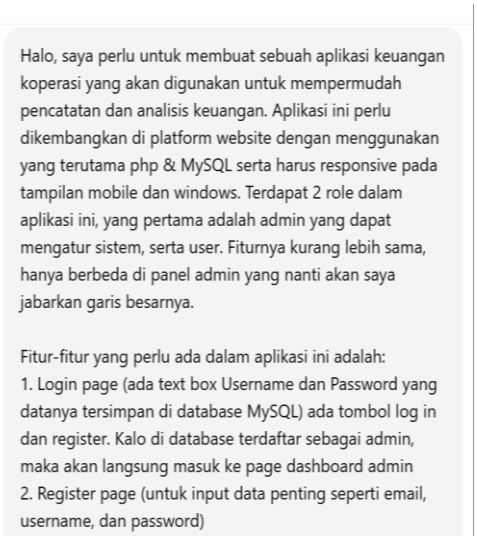
Terdapat empat parameter yang digunakan, pertama meliputi fungsionalitas kode apakah bekerja sesuai fungsi yang diuji dengan *Black Box Testing*. Kedua yaitu konsistensi kode, keteraturan penamaan variabel, struktur, dan kesesuaian dengan konsep yang dtentukan. Ketiga adalah efisiensi prompt, jumlah iterasi *prompt* yang dibutuhkan hingga kode sesuai. Terakhir berada pada kualitas dokumentasi berupa penjelasan atau komentar yang ditambahkan pada hasil kode *AI*.

Data kemudian dianalisis menggunakan analisis deskriptif kualitatif, dengan mencantumkan hasil pengujian serta eksekusi *prompt* sebagai bukti pendukung.

HASIL DAN PEMBAHASAN

Bab ini menyajikan hasil penelitian yang telah dilakukan beserta pembahasannya. Penelitian diawali dengan penyusunan prompt terstruktur yang diberikan kepada *ChatGPT* untuk menghasilkan kode aplikasi keuangan koperasi. Setiap *prompt* dirancang untuk membangun fitur secara bertahap, dari proses *log in*, pencatatan transaksi, pengelolaan database, hingga laporan keuangan. *Prompt* yang pertama kali diberikan berfungsi sebagai arahan dasar bagi *ChatGPT* untuk memahami dan membangun kode yang sesuai dengan kebutuhan aplikasi.

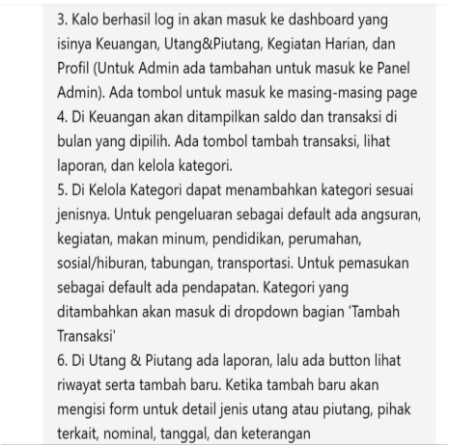
Gambar 1. *Prompt* awal yang berisi deskripsi aplikasi yang diinginkan secara detil(1)


Halo, saya perlu untuk membuat sebuah aplikasi keuangan koperasi yang akan digunakan untuk mempermudah pencatatan dan analisis keuangan. Aplikasi ini perlu dikembangkan di platform website dengan menggunakan yang terutama php & MySQL serta harus responsive pada tampilan mobile dan windows. Terdapat 2 role dalam aplikasi ini, yang pertama adalah admin yang dapat mengatur sistem, serta user. Fiturnya kurang lebih sama, hanya berbeda di panel admin yang nanti akan saya jabarkan garis besarnya.

Fitur-fitur yang perlu ada dalam aplikasi ini adalah:
1. Login page (ada text box Username dan Password yang datanya tersimpan di database MySQL) ada tombol log in dan register. Kalo di database terdaftar sebagai admin, maka akan langsung masuk ke page dashboard admin
2. Register page (untuk input data penting seperti email, username, dan password)

Untuk membuat *prompt* yang baik, perintah dimulai dengan menjelaskan tujuan yang diinginkan. Jelaskan apa yang ingin dibuat dan apa fungsi aplikasi yang diinginkan, serta tentukan apakah berupa aplikasi atau *website*. Sertakan tipe *role* pengguna dan jelaskan diferensiasi *role* seperti yang dicontohkan pada Gambar 1.

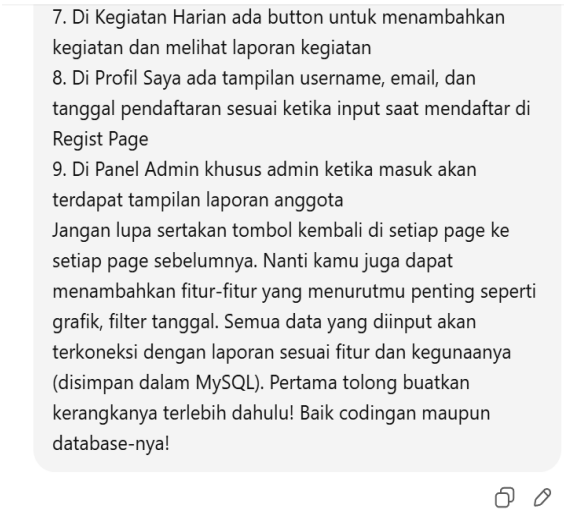
Gambar 2. *Prompt* awal yang berisi deskripsi aplikasi yang diinginkan secara detil(2)


3. Kalo berhasil log in akan masuk ke dashboard yang isinya Keuangan, Utang&Piutang, Kegiatan Harian, dan Profil (Untuk Admin ada tambahan untuk masuk ke Panel Admin). Ada tombol untuk masuk ke masing-masing page
4. Di Keuangan akan ditampilkan saldo dan transaksi di bulan yang dipilih. Ada tombol tambah transaksi, lihat laporan, dan kelola kategori.
5. Di Kelola Kategori dapat menambahkan kategori sesuai jenisnya. Untuk pengeluaran sebagai default ada angsuran, kegiatan, makan minum, pendidikan, perumahan, sosial/hiburan, tabungan, transportasi. Untuk pemasukan sebagai default ada pendapatan. Kategori yang ditambahkan akan masuk di dropdown bagian 'Tambah Transaksi'
6. Di Utang & Piutang ada laporan, lalu ada button lihat riwayat serta tambah baru. Ketika tambah baru akan mengisi form untuk detail jenis utang atau piutang, pihak terkait, nominal, tanggal, dan keterangan

ISSN: 2986-531X

Dilanjutkan dengan penekanan fungsi setiap fitur yang diinginkan, pada tahap ini perlu menjelaskan secara detil fitur-fitur tersebut, bahkan alur dan konektivitas antar fitur yang dapat dilihat pada Gambar 2.

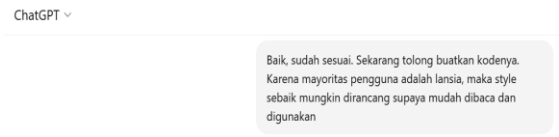
Gambar 3. *Prompt* awal yang berisi deskripsi aplikasi yang diinginkan secara detil(3)



Terakhir, sesuai tertera pada gambar 3, untuk mengakhiri *prompt* awal diberikan perintah supaya *AI* tidak lupa menambahkan fitur-fitur penting yang mungkin terlewat, seperti tombol kembali. Timbal balik yang diinginkan pada awal prompting adalah struktur aplikasi, selebihnya dapat ditambahkan setelahnya. Jika sudah setuju dengan balasan pertama yang diberikan maka mulai dapat meminta susunan kode dari page di struktur utama, sebaliknya jika masih tidak puas dapat meminta *AI* membuatkan ulang hingga dirasa jawaban yang diinginkan sudah sesuai. Aplikasi yang membutuhkan koneksi database harus diperintahkan pula untuk disambungkan ke layanan database seperti *MySQL*.

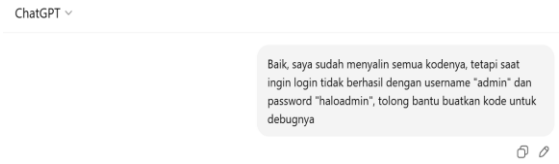
Kode yang dihasilkan sebagai jawaban dari prompt awal diperiksa dari segi fungsionalitas dan keterbacaan. Jika ditemukan celah atau kekurangan seperti ketidakkonsistenan penamaan variabel atau logika yang berjalan tidak semestinya, prompt direvisi dan menekankan pada bagian yang salah lalu dijalankan ulang hingga kode yang dihasilkan sesuai kebutuhan. Proses ini berlangsung secara berulang *(iterative prompting),* sampai hasil kode yang diperoleh dapat digunakan langsung untuk aplikasi koperasi.

Gambar 4. Contoh *Iterative Prompting* untuk Menekankan Konsep



Gambar 4 memperlihatkan bagaimana *iterative prompting* digunakan untuk menguatkan pemahaman konsep. Proses ini berlangsung secara bertahap dimulai dari penjabaran dari instruksi awal secara umum, kemudian diperbaiki dengan masukan tambahan, hingga menghasilkan jawaban yang lebih terarah dan sesuai kebutuhan. Setiap langkah bertujuan membantu model menyaring informasi, menambahkan detil, dan menjaga agar hasil akhir sesuai yang diinginkan.

Gambar 5. Contoh *Iterative Prompting* Ketika Ditemukan Error

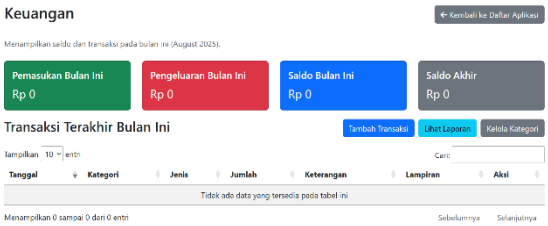


Kode dari *ChatGPT* kemudian diimplementasikan ke dalam aplikasi menggunakan *Visual Studio Code* dan diintegrasikan dengan basis data *MySQL* melalui *PHP MyAdmin*. Setelah semua fitur berhasil ditanamkan, dilakukan pengujian fungsional menggunakan metode *Functional Black Box Testing.* Pengujian ini dilakukan dengan cara memberi kan *input* pada setiap fitur aplikasi dan memeriksa apakah *output* yang dihasilkan sesuai dengan yang dharapkan, tanpa memeriksa isi kode.

Gambar 6. Tampilan Hasil Akhir dari *Iterative Prompting* pada Gambar 5

Setelah meminta *AI* melakukan perbaikan kode untuk memperbaiki *error* yang terjadi sebelumnya, admin dapat masuk ke halaman admin tanpa kendala. Maka ketika memasukkan username yang telah terdaftar sebagai admin di database pada form log in yang tertera dalam Gambar 6, pengguna akan langsung diarahkan ke halaman khusus admin.

Gambar 7. Tampilan Hasil Akhir Keuangan *Page*



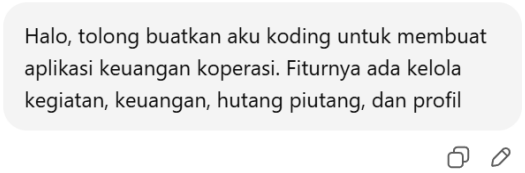
Sesuai pada Gambar 7, tampilan dari aplikasi yang dihasilkan berwarna mencolok dan berukuran besar uuntuk memudahkan para pengguna yang mayoritas merupakan lansia untuk mengoperasikan aplikasi dimana hal ini sesuai dengan isi *prompt* yang diminta ke *AI.*

Hasil pengujian dicatat dalam tabel dengan kategori “Sesuai” jika fitur bekerja sesuai fungsinya, dan “Gagal” jika ditemukan masalah.

Tabel 1. Hasil uji aplikasi menggunakan functional black box testing

Page	***Testing Button & Dropdown Box***	***Status***
Log In	*Log In* *Register*	Sesuai Sesuai
Register	*Register* *Log In*	Sesuai Sesuai
Dashboard	Buka Aplikasi Keuangan Kelola Utang & Piutang Buka Kegiatan Harian Buka Profil *Log Out* Buka Panel Admin	Sesuai Sesuai Sesuai Sesuai Sesuai Sesuai
Keuangan	Tambah transaksi Lihat Laporan Kelola Kategori Kembali ke *Dashboard*	Sesuai Sesuai Sesuai Sesuai
Tambah	Tanggal	Sesuai
Transaksi	Jenis Transaksi Kategori *Choose File* Simpan Batal Kembali	Sesuai Sesuai Sesuai Sesuai Sesuai Sesuai
Lihat Laporan	Batas Tanggal Filter *Export* ke Excel Kembali	Sesuai Sesuai Sesuai Sesuai
Kelola Kategori	Kategori Pengeluaran Kategori Pemasukan Tambah Kembali ke Keuangan	Sesuai Sesuai Sesuai Sesuai
Utang Piutang	Lihat Riwayat Tambah Batal Simpan Kembali ke Daftar *App*	Sesuai Sesuai Sesuai Sesuai Sesuai
Riwayat	Batas Tanggal Jenis Filter *Reset* Kembali ke Utang Piutang	Sesuai Sesuai Sesuai Sesuai Sesuai
Kegiatan Harian	Tambah Lihat laporan Kembali ke Aplikasi	Sesuai Sesuai Sesuai
Laporan Kegiatan	Batas Tanggal Status *Filter* Kembali ke Kegiatan	Sesuai Sesuai Sesuai Sesuai
Profil Saya	Ganti *Password* *Log Out* Kembali ke Aplikasi	Sesuai Sesuai Sesuai
Laporan Anggota	Pilih Bulan Pilih Tahun Pilih Banyak Entri Tampilkan Kembali ke *Dashboard*	Sesuai Sesuai Sesuai Sesuai Sesuai
KARA	Lihat Atur/Organisir Kembali	Sesuai Sesuai Sesuai

Gambar 8. Contoh *Lazy Prompt*



Berbeda dengan jenis prompting sebelumnya yang menekankan arahan jelas dan bertahap, *lazy prompt* justru cenderung membiarkan *AI* berpikir sendiri tanpa kontrol dari pengguna. Pola ini biasanya muncul ketika instruksi diberikan terlalu singkat, kabur, atau bahkan hanya berupa kata kunci seadanya. Hasilnya sering kali kurang memuaskan seperti jawaban yang terasa acak, tidak akurat, bahkan jauh dari kebutuhan sebenarnya.

Masalah lain dari *lazy prompt* adalah ketika output yang dihasilkan ternyata salah atau tidak sesuai, proses perbaikannya justru lebih merepotkan. Alih-alih membantu, kita dipaksa menghabiskan lebih banyak waktu untuk mengoreksi, mengulang instruksi, atau bahkan membuang hasil sebelumnya.

KESIMPULAN

Penelitian ini menunjukkan bahwa penerapan *iterative prompting* dalam rangkaian *multi-step prompting* dapat meningkatkan kualitas kode yang dihasilkan *AI* dalam pengembangan aplikasi keuangan koperasi. Pendekatan yang dilakukan bertahap dengan penjelasan konsep, batasan, dan detail yang jelas mampu mengurangi kesalahan fungsi, memperbaiki konsistensi penamaan variabel, serta meningkatkan keterbacaan kode. Pengujian *Black Box* membuktikan bahwa fitur-fitur yang dijalankan sesuai dengan fungsinya. Dengan demikian, teknik *prompt engineering* dengan penyempurnaan terstruktur hingga menjadi strategi untuk memaksimalkan pemanfaatan *AI,* khususnya bagi koperasi yang memiliki keterbatasan sumber daya pengembang. Sebagai tindak lanjut, penelitian serupa dapat dikembangkan untuk berbagai sektor dan menguji model lain, guna mendapatkan pendekatan yang semakin efektif dan aman.

DAFTAR PUSTAKA

[1] C. Liu *et al.*, "Guiding ChatGPT for Better Code Generation: An Empirical Study," *Proc. - 2024 IEEE Int. Conf. Softw. Anal. Evol. Reengineering, SANER 2024*, pp. 102–113, 2024, doi: 10.1109/SANER60148.2024.00018.

[2] J. Wei *et al.*, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," *Adv. Neural Inf. Process. Syst.*, vol. 35, no. NeurIPS, pp. 1–43, 2022.

[3] Y. Fu *et al.*, "Security Weaknesses of Copilot-Generated Code in GitHub Projects: An Empirical Study," *ACM Trans. Softw. Eng. Methodol.*, vol. 1, no. 1, 2025, doi: 10.1145/3716848.

[4] C. Negri-Ribalta, R. Geraud-Stewart, A. Sergeeva, and G. Lenzini, "A systematic literature review on the impact of AI models on the security of code generation," *Front. Big Data*, vol. 7, 2024, doi: 10.3389/fdata.2024.1386720.

[5] X. Hou *et al.*, "Large Language Models for Software Engineering: A Systematic Literature Review," *ACM Trans. Softw. Eng. Methodol.*, vol. 33, no. 8, pp. 1–79, 2024, doi: 10.1145/3695988.

[6] E. G. Santana *et al.*, "Which Prompting Technique Should I Use? An Empirical Investigation of Prompting Techniques for Software Engineering Tasks," 2025, [Online]. Available: http://arxiv.org/abs/2506.05614

[7] Z. Shao, Y. Gong, Y. Shen, M. Huang, N. Duan, and W. Chen, "Synthetic Prompting: Generating Chain-of-Thought Demonstrations for Large Language Models," *Proc. Mach. Learn. Res.*, vol. 202, pp. 30706–30775, 2023.

[8] T. Wu, M. Terry, and C. J. Cai, *AI Chains: Transparent and Controllable Human-AI Interaction by Chaining Large Language Model Prompts*, vol. 1, no. 1. Association for Computing Machinery, 2022. doi: 10.1145/3491102.3517582.

[9] S. Krishna, C. Agarwal, and H. Lakkaraju, "Understanding the Effects of Iterative Prompting on Truthfulness," *Proc. Mach. Learn. Res.*, vol. 235, pp. 25583–25602, 2024.

[10] B. Reeves *et al.*, "Evaluating the Performance of Code Generation Models for Solving Parsons Problems with Small Prompt Variations," *Annu. Conf. Innov. Technol. Comput. Sci. Educ. ITiCSE*, vol. 1, pp. 299–305, 2023, doi: 10.1145/3587102.3588805.

[11] C. Treude and M.-A. Storey, "Generative AI and Empirical Software Engineering: A Paradigm Shift," 2025.

[12] A. Nurdany and A. C. Prajasari, "Digitalization in Indonesian Cooperatives: Is It Necessary?," *J. Dev. Econ.*, vol. 5, no. 2, p. 125, 2020, doi: 10.20473/jde.v5i2.19447.

[13] D. R. Indriastuti, R. Susanti, R. Wulandari, and J. Intan, "Building public trust in cooperatives through digitalization," *J. Manag. Digit. Bus.*, vol. 5, no. 1, pp. 69–84, 2025, doi: 10.53088/jmdb.v5i1.1410.

[14] S. Shukla and H. Joshi, "Security degradation in iterative AI code generation: A systematic analysis of the paradox," 2025, [Online]. Available: https://arxiv.org/abs/2506.11022

[15] A. Aggarwal, P. Kr. Lohia, N. Seema, K. Dey, and D. Saha, "Black box fairness testing of machine learning models," *Proc. 2019 27th ACM Jt. Meet. Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, 2019, doi: https://doi.org/10.1145/3338906.3338937.

[16] Team, "Black Box Testing Adalah: Teknik Dan Contoh Pengujiannya." [Online]. Available: https://codingstudio.id/blog/black-box-testing-adalah/

[17] K. Meinke, "Automated black-box testing of functional correctness using function approximation," vol. 9, no. 4, 2004, doi: 10.1145/1013886.1007532.

PENULIS

Vincenzhea Chrissa Siwy, prodi Sistem Informasi, Fakultas Ilmu Komputer, Universitas Katolik Soegijapranata.

Stephani Inggrit Swastini, prodi Sistem Informasi, Fakultas Ilmu Komputer, Universitas Katolik Soegijapranata.

Ridwan Sanjaya, prodi Sistem Informasi, Fakultas Ilmu Komputer, Universitas Katolik Soegijapranata.

Andre Kurniawan Pamudji, prodi Sistem Informasi, Fakultas Ilmu Komputer, Universitas Katolik Soegijapranata.