LOGIC: Jurnal Penelitian Informatika
Vol. 4, Issue 1, pp. 22–28 (2026)
doi: http://doi.org/10.25124/logic.v4i1.6405

RESEARCH ARTICLE

Analisis Perbandingan Proses Pembangunan Aplikasi Berbasis *Website*: Pendekatan Manual Vs *AI-generated*

Aldiansyah and Dana Sulistyo*

Fakultas Informatika, Universitas Telkom, Bandung, 40257, Jawa Barat, Indonesia
*Corresponding author: danakusumo@telkomuniversity.ac.id

Abstrak

Penelitian ini membandingkan proses pengembangan aplikasi web dengan pendekatan manual dan yang dihasilkan oleh *AI-generated*, fokus pada kecepatan, kemudahan pemeliharaan, dan duplikasi kode. Tujuan utama penelitian adalah untuk mengevaluasi efisiensi waktu, kemudahan pemeliharaan, dan tingkat duplikasi kode dari kedua pendekatan. Metodologi yang digunakan meliputi pembuatan aplikasi web secara manual dan dengan bantuan *AI-generated*, diikuti dengan pengujian waktu dan analisis kode menggunakan SonarQube. Proses pembuatan aplikasi terdiri dari pendaftaran, login, *dashboard*, dan pemilihan data. Hasil penelitian menunjukkan bahwa pendekatan *AI-generated* secara signifikan lebih cepat dalam pembuatan kode dibandingkan dengan pendekatan manual. Namun, meskipun *AI-generated* mempercepat proses, kode yang dihasilkan cenderung memiliki tingkat duplikasi yang lebih tinggi dan lebih banyak isu yang harus diperbaiki. Sebaliknya, pendekatan manual meskipun memerlukan waktu lebih lama, menghasilkan kode dengan duplikasi yang lebih rendah dan lebih mudah dipelihara. Kesimpulan penelitian ini menegaskan pentingnya mempertimbangkan keseimbangan antara efisiensi waktu dan kualitas kode saat memilih metode pengembangan perangkat lunak. Penelitian ini memberikan wawasan tentang dampak penggunaan *AI-generated* dalam pengembangan perangkat lunak dan bagaimana transisi antara pendekatan *AI-generated* dan manual dapat mempengaruhi proses pengembangan.

Key words: Pengembangan Aplikasi Web, *AI-generated Code*, Metode Manual vs *AI-generated*, Efisiensi Waktu, *Maintainability*, Duplikasi Kode, SonarQube, Analisis Kode Statik, Perbandingan Pengembangan Perangkat Lunak.

Pendahuluan

Pengembangan aplikasi *website* adalah bidang yang terus berkembang, di mana efisiensi waktu dan kualitas kode merupakan faktor kunci dalam menentukan keberhasilan proyek. Metode pengembangan manual, meskipun memberikan kontrol penuh atas setiap aspek pembuatan aplikasi, sering kali mengalami tantangan besar terkait waktu dan potensi kesalahan. Proses manual ini memerlukan keahlian teknis yang tinggi dan perhatian detail yang mendalam, yang dapat mengakibatkan masalah seperti duplikasi kode dan kesulitan dalam pemeliharaan.

Dengan kemajuan teknologi, khususnya dalam kecerdasan buatan (AI), pendekatan pengembangan aplikasi *website* mengalami transformasi yang signifikan. *AI-generated*, melalui algoritma dan model komputasional, memungkinkan otomatisasi berbagai aspek pembuatan aplikasi, termasuk desain dan penulisan kode. Meskipun *ai-generated* menawarkan kecepatan dan efisiensi waktu yang lebih tinggi, kualitas kode yang dihasilkan sering menjadi perhatian, terutama terkait dengan penerapan prinsip *clean code*. penelitian ini bertujuan untuk membandingkan efektivitas antara pengembangan aplikasi *website* secara manual dan menggunakan *ai-generated* dalam hal efisiensi waktu, kemudahan pemeliharaan, dan tingkat duplikasi kode. Penelitian ini akan menggunakan alat analisis kode seperti SonarQube untuk menilai kualitas kode dari kedua pendekatan.

Penelitian melibatkan pembuatan aplikasi web dengan dua pendekatan: manual dan *ai-generated*. Setelah pembuatan, analisis kode akan dilakukan menggunakan SonarQube untuk menilai *clean code*, duplikasi, dan kompleksitas. Penelitian ini bertujuan untuk memberikan pemahaman yang lebih baik tentang kelebihan dan kekurangan dari kedua pendekatan serta dampaknya terhadap efisiensi waktu dan kualitas kode. Isu *maintainability* (kemudahan pemeliharaan) akan dianalisis untuk melihat sejauh mana kode dapat diperbaiki atau dimodifikasi tanpa menyebabkan masalah baru. Sementara itu, *code duplication* (duplikasi kode) akan diteliti untuk mengukur sejauh mana adanya pengulangan kode yang dapat mengurangi efisiensi dan kualitas aplikasi.

Penulisan jurnal ini dimulai dengan studi pustaka yang mencakup penelitian terkait dengan pengembangan manual dan *ai-generated*. Bagian sistem yang dibangun akan menjelaskan implementasi dari kedua pendekatan, diikuti dengan evaluasi hasil dan perbandingan

temuan. Kesimpulan akan menyimpulkan hasil penelitian dan memberikan rekomendasi untuk pengembangan lebih lanjut.

Pengembangan aplikasi *website* adalah topik yang penting karena aplikasi *website* memainkan peran kunci di berbagai sektor. Efisiensi waktu dan kualitas kode sangat menentukan keberhasilan proyek pengembangan. Dalam era digital ini, perbandingan antara metode pengembangan manual dan berbasis *ai-generated* menjadi semakin relevan untuk dieksplorasi.

Penelitian ini menarik karena teknologi *ai-generated* semakin banyak digunakan untuk meningkatkan produktivitas dan kualitas dalam pengembangan aplikasi *website*. Namun, masih ada celah dalam pemahaman tentang dampak *ai-generated* terhadap kualitas kode. Pengembangan manual sering kali lebih lambat dan rentan terhadap duplikasi kode, sedangkan *ai-generated* berpotensi mengatasi masalah ini. Oleh karena itu, penelitian ini bertujuan untuk mengevaluasi efektivitas dan kualitas kode dari kedua pendekatan tersebut.

Pengembangan aplikasi *website* adalah topik yang penting karena aplikasi *website* memainkan peran kunci di berbagai sektor. Efisiensi waktu dan kualitas kode sangat menentukan keberhasilan proyek pengembangan. Dalam era digital ini, perbandingan antara metode pengembangan manual dan berbasis kecerdasan buatan (AI) menjadi semakin relevan untuk dieksplorasi.

Topik dan Batasannya

Berikut adalah contoh batasan masalah untuk skripsi yang fokus pada pengembangan aplikasi berbasis *website* menggunakan pendekatan manual dan *ai-generated*:

1. Pengembangan aplikasi *website* dibatasi pada pembuatan fitur *User Register, User Login, Dashboard*, *Switch CB PageSwitch LBS, dan Page*, Hanya fitur-fitur tersebut yang akan dianalisis dan dibandingkan antara pendekatan manual dan pendekatan *ai-generated*.
2. Evaluasi kualitas kode dibatasi pada aspek *maintainability* dan duplikasi kode, menggunakan SonarQube sebagai alat utama untuk analisis statis. Pengujian dan evaluasi hanya mencakup kode *backend*, tanpa memperhitungkan aspek UI/UX atau desain *front-end*.
3. *Platform* yang digunakan untuk pengembangan adalah Laravel untuk manual coding dan alat *ai-generated* yang dipilih untuk pendekatan *ai-generated*. Penggunaan alat tambahan seperti *Bootstrap* 5 untuk *front- end* adalah untuk konsistensi tampilan, tetapi tidak menjadi fokus utama dalam evaluasi.
4. Fokus utama analisis adalah pada kualitas kode, termasuk *maintainability* dan duplikasi kode. Pengukuran aspek lain seperti kecepatan eksekusi atau performa *runtime* dari aplikasi tidak akan menjadi bagian dari evaluasi ini.

Tinjauan Pustaka

Penelitian ini disusun sebagai berikut. Dibagian 2, membahas secara singkat ilmu atau studi terkait penelitian ini. Pada bagian 3, adalah penjelasan mengenai tahapan penelitian, skenarop pembuatan *website* lalu setiap tugas yang di kerjakan manual maupun *ai-generated*. Pada bagian 4, membahas Waktu pengerjaan, pengujian dan hasil analisis *code static*. Terakhir, kesimpulan dan saran untuk pengembangan selanjutanya di bagian 5.

Website

Website adalah aplikasi yang dapat dijalankan dengan menggunakan *website* browser, saat ini hampir semua gawai dapat menjalankan i browser yang menyebabkan *website* dapat dibuka di hampir semua gawai yang ada [1]. *Website* adalah wadah digital yang terdiri dari halaman-halaman saling terhubung yang dapat diakses melalui internet. Setiap *website* memiliki domain unik dan umumnya mencakup halaman beranda sebagai titik awal navigasi. Komponen utama termasuk konten berupa teks, gambar, dan multimedia, *hyperlink* untuk navigasi, desain responsif untuk keterbacaan pada berbagai perangkat, serta fungsi seperti formulir, *database*, dan keamanan. *Website* berfungsi sebagai sarana penyampaian informasi, interaksi *online*, dan penawaran layanan atau produk, dengan variasi jenis seperti blog, situs berita, *e-commerce*, dan forum diskusi. Penggunaan teknologi seperti analitika, SEO, dan integrasi media sosial menambah dimensi fungsional dan interaktif bagi pengguna, sementara keberlanjutan pengalaman *online* juga dipertahankan melalui pembaruan perangkat lunak dan keamanan.

AI-generated

AI-generated adalah salah satu cabang *artificial intelligence* yang memungkinkan algoritma untuk menghasilkan konten secara otomatis, mulai dari teks, gambar, audio, hingga video. Menurut Goldman Sachs [2], *ai-generated* dapat mendorong peningkatan sebesar 7 persen (atau hampir 7 miliar USD) dalam produk domestik bruto (PDB) global. Mereka juga mengantisipasi bahwa *ai-generated* dapat meningkatkan pertumbuhan produktivitas dengan poin persentase sebesar 1,5 selama 10 tahun [3].

Algoritma *ai-generated* membuka kemungkinan baru dalam penelitian dengan cara menggali dan menganalisis data yang kompleks. Ini memungkinkan penemuan tren dan pola yang sebelumnya tidak terdeteksi, serta dapat merangkum konten, menghasilkan ide, dan membuat dokumentasi rinci. Penggunaan *ai-generated* generatif secara signifikan mempercepat inovasi, seperti dalam industri farmasi di mana digunakan untuk mengoptimalkan urutan protein dan mempercepat pengembangan obat. Selain itu, *ai-generated* juga memberikan dampak positif pada pengalaman pelanggan dengan kemampuannya merespons percakapan manusia secara alami. Penggunaan chatbot dan asisten virtual dapat meningkatkan keterlibatan pelanggan melalui komunikasi yang dipersonalisasi.

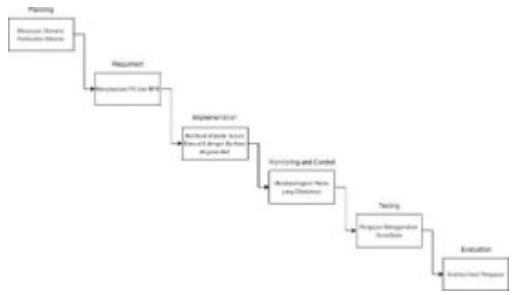
Dalam konteks bisnis, *ai-generated* membantu mengoptimalkan proses di berbagai bidang, termasuk pemasaran, keuangan, dan layanan pelanggan. Ini melibatkan ekstraksi data, evaluasi skenario untuk pengurangan biaya, serta penghasilan data sintetis untuk pembelajaran mesin. Penerapan *ai-generated* juga meningkatkan produktivitas karyawan dengan berbagai cara, seperti mendukung tugas kreatif, menghasilkan saran kode perangkat lunak, dan memberikan dukungan kepada manajemen dalam pembuatan laporan dan proyeksi.

Dalam berbagai industri, *ai-generated* memiliki potensi besar. Sebagai contoh, dalam jasa keuangan, chatbot dapat memberikan rekomendasi produk, sementara dalam layanan kesehatan, *ai-generated* dapat mempercepat penelitian obat dengan merancang urutan protein baru. Di sektor otomotif, perusahaan dapat mengoptimalkan desain suku cadang dan layanan pelanggan, sedangkan dalam media dan hiburan, aigenerated dapat menghasilkan konten baru secara lebih efisien.

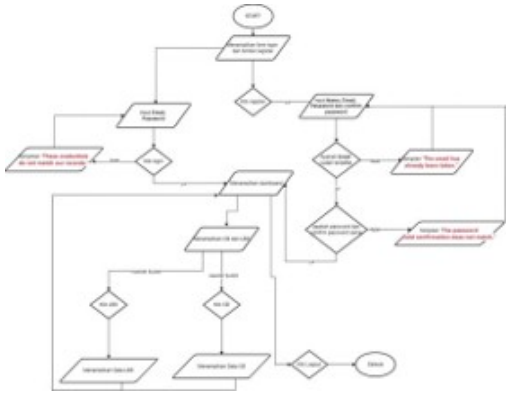
Telekomunikasi dapat meningkatkan layanan pelanggan dan performa jaringan dengan menggunakan *ai-generated*, sementara dalam sektor energi, teknologi ini dapat membantu analisis data kompleks, meningkatkan keamanan lokasi operasional, dan mengoptimalkan produksi energi [3].

SonarQube

SonarQube adalah *platform open-source* yang dikembangkan oleh SonarSource, digunakan untuk inspeksi berkelanjutan pada kualitas kode aplikasi [4]. Alat ini menggunakan aturan analisis yang dapat disesuaikan untuk mengevaluasi berbagai aspek dari kode, termasuk struktur, kompleksitas, dan kepatuhan terhadap standar pengkodean.



Gambar 1. Tahap Penelitian



Gambar 2. Skenario Pembuatan *Website*

SonarQube menyediakan wawasan mendalam mengenai *maintainability* dan kualitas kode secara keseluruhan melalui *dashboard* dan laporan terperinci. Dengan analisis ini, pengembang dapat mengidentifikasi dan memperbaiki masalah sejak awal dalam siklus pengembangan, memastikan bahwa kode tetap bersih, aman, dan mudah dipelihara. Berbeda dengan alat pengujian yang fokus pada validasi fungsionalitas perangkat lunak, SonarQube fokus pada analisis kualitas kode untuk meningkatkan keseluruhan *maintainability* dan keamanan aplikasi.

Maintainability

Pemeliharaan sistem perangkat lunak adalah topik yang banyak diteliti. Sejumlah besar publikasi melaporkan pentingnya masalah ini dan menyatakan pentingnya pemeliharaan yang baik dalam pengembangan perangkat lunak [5]. pemeliharaan adalah properti sistem perangkat lunak sementara pemeliharaan didefinisikan sebagai proses yang sesuai setelah pengiriman [6]. Ini mencakup faktor-faktor seperti kompleksitas kode, struktur kode, dan keterbacaan, yang semuanya mempengaruhi seberapa mudah kode dapat dipahami dan dimodifikasi.

Code Duplications

Duplikasi kode adalah masalah umum, dan merupakan tanda desain yang buruk [7]. Ini terjadi ketika blok kode yang identik atau hampir identik digunakan di berbagai tempat dalam aplikasi, seringkali karena kurangnya modul atau fungsi yang dapat digunakan kembali. Duplikasi meningkatkan risiko inkonsistensi dan kompleksitas, membuat kode sulit dipahami dan dipelihara. SonarQube mendeteksi dan melaporkan duplikasi kode, memungkinkan pengembang untuk melakukan *refactoring* dan menggabungkan logika yang terulang, sehingga meningkatkan *maintainability* dan efisiensi perangkat lunak.

Metodologi Penelitian

Gambar 1 menggambarkan tahapan di mulai pembuatan *website* lalu pergujian *code* statik. Terdapat 6 proses pada penelitian ini, di mulai dari menyiapkan scenario, pembuatan *website* secara manual dan di bantu dengan ai-enerate, membandingkan waktu yang di habiskan pengujian menggunakan sonarqube dan menganalis pengujian sonarqube.

Skenario Pembuatan *Website*

Gambar 2 menggambarkan skenario pembuatan *website* yang di mulai dari register hingga melihat *report data*. Terdapat 6 tahap pada *website*, dimulai dari register, login, halaman *dashboard*, memilih *switch*, memilih Gardu Induk atau Gardu Hubung, memilih penyulang, dan tabel data.

Table 1. Indentifikasi Usecase

Use Case	Nama Use Case	Deskripsi
UC-1	*User Register*	Pengguna melakukan login dengan menginput email
UC-2	*User Login*	Pengguna baru dapat mendaftar dengan memasukkan password, dan konfirmasi password.
UC-3	*Dashboard*	Pengguna dapat melihat *dashboard* setelah login berhasil
UC-4	*Switch CB Page*	Pengguna memilih data CB untuk ditampilkan di *dashboard*
UC-5	*Switch LBS Page*	Pengguna memilih data LBS untuk ditampilkan di *dashboard*
UC-6	*Logout*	Pengguna melakukan *logout* dari sistem.

Program *Website*

Program visualisasi data untuk *Switch* Gardu Induk dan *Load Break Switch* dirancang untuk mengumpulkan, memproses, dan menyajikan data operasional dari perangkat tersebut dalam format yang mudah dipahami. Program ini mengolah data seperti status *on/off*, arus, dan tegangan untuk membuat grafik diagram batang yang memungkinkan pengguna menganalisis tren, membandingkan data, dan mendeteksi anomali. Dengan fitur interaktif seperti filter data dan notifikasi, program ini membantu pengguna memantau kinerja perangkat secara *real-time*, meningkatkan pengambilan keputusan, dan mempermudah pemeliharaan sistem kelistrikan secara keseluruhan.

Data Requirement

1. *Functional Requirement* Fitur-fitur dalam aplikasi ini dibuat dengan dua cara: cara pertama akan dilakukan secara manual terlebih dahulu lalu setelah pembuatan selesai akan melakukan cara kedua yaitu pembuatan yang dibantu *ai-generated*, seperti pada tabel 1. Fitur tambahan yang dikembangkan menggunakan pendekatan yang dihasilkan AI dieksekusi sebelum fitur yang dikembangkan secara manual.

   Tujuan dari langkah ini adalah untuk mengidentifikasi dan menunjukkan potensi risiko dan bias yang mungkin timbul dari penggunaan metode AI dalam pengembangan fitur, sehingga mengurangi kemungkinan dampak yang tidak diinginkan pada hasil akhir proyek. Pendekatan manual pada tabel 2 memberikan kontrol

Table 2. Tambahan Use Case

Use case	Nama Use Case	Deskripsi
UC-7	DataTabel	Pengguna melihat data dalam format tabel di *switch* tempat yang dipilih.
UC-8	Grafik	Pengguna melihat data dalam bentuk grafik di setiap page.
UC-9	*Simple User Greeting*	Pengguna menerima pesan sambutan sederhana setelah login berhasil.

Table 3. Non-Functional Requiment

Usecase	Nama Use Case	Deskripsi
UC-10	*Maintainability*	Kode harus mengikuti prinsip *clean code* dan meminimalisir *code duplication*, didukung dengan analisis SonarQube.

Table 4. Waktu Pengerjaan Setiap Tugas

Use case	Task	Manual	*AI-generated*
1	*User Register*	23 Menit	06 Menit
2	*User Login*	23 Menit	06 Menit
3	*Dashboard*	30 Menit	12 Menit
4	*Switch CB Page*	2 jam16 menit	50 Menit
5	*Switch LBS Page*	16 Menit	10 Menit

penuh atas setiap elemen pengembangan, memungkinkan penyesuaian yang lebih mudah dan pemahaman yang lebih baik tentang kebutuhan sistem. Ini penting untuk memastikan bahwa solusi *ai-generated* diimplementasikan dengan benar dan sesuai dengan kebutuhan spesifik proyek.

2. *Non-Functional Requirement* Di bawah ini, kami menyajikan tabel yang menjelaskan *non-functional requirement* sebagai berikut dalam tabel 3

Evaluasi

Evaluasi dilakukan menggunakan *Stopwatch* dan pengujian sonarqube. *Stopwatch* di gunakan untuk menghitung pengerjaan setiap tugas pembuatan *website* dan pengujian SonarQube untuk menganalisis *code* statik. Waktu akan di bandingkan Ketika mengerjakan secara manual dan di bantu *ai-generated* dan Analisa *code* statik untuk mencari 3 aspek utama yaitu, *maintainability*, dan *code duplications*.

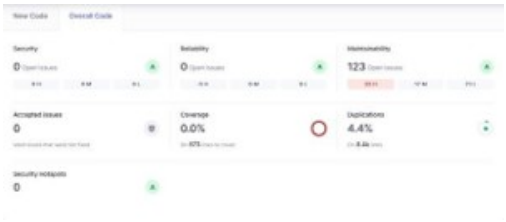
Hasil dan Pembahasan

Perbandingan Waktu

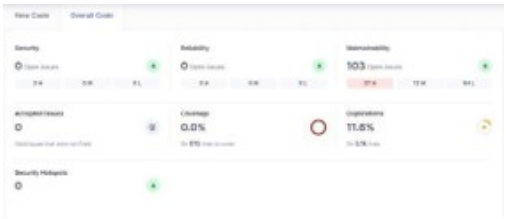
Penelitian ini dilakukan dengan dua skenario utama: *supervised learning* dan *zero-shot*. Setiap skenario diuji pada dua kondisi dataset yang berbeda: dataset yang telah melalui tahap prapemrosesan pembersihan (*cleaned*) dan dataset asli tanpa pembersihan (*uncleaned*). Tabel 5 menunjukkan bahwa waktu pemrosesan untuk setiap usecase seringkali lebih lama untuk pendekatan manual dibandingkan pendekatan yang dihasilkan *ai-generated*. Misalnya saja proses registrasi pengguna dan login pengguna yang biasanya memakan waktu 23 menit secara manual, namun dengan bantuan *ai-generated*, waktu prosesnya dikurangi menjadi 6 menit. Demikian pula, peralihan pekerjaan di sisi

Table 5. Waktu Pengerjaan *Ai-generated* Lebih Dahulu

Use case	Task	Manual	*AI-generated*
1	User Register	30 Menit	50 Jam
2	User Login	35 Menit	56 Menit
3	*Dashboard*	7 Menit	15 Menit



Gambar 3. Hasil Pengujian *Website* Secara Manual di Sonarqube



Gambar 4. Hasil Pengujian *Website* Secara *AI-generated* di Sonarqube

CB biasanya memakan waktu 2 jam 16 menit secara manual, namun penggunaan *ai-generated* dapat mempersingkatnya menjadi 50 menit.

Tabel 5 ini membandingkan waktu pemrosesan untuk berbagai tugas yang dilakukan pertama kali menggunakan metode yang dihasilkan aigenerated dan kemudian menggunakan tenaga kerja manual. Pendekatan ini mengeksplorasi bagaimana efisiensi waktu yang dihasilkan oleh *ai-generated* dapat berdampak pada proses kerja dan bagaimana transisi dari efisiensi yang dihasilkan oleh *ai-generated* ke efisiensi waktu manual dapat menyebabkan gangguan. Pertama, penggunaan sesuatu yang dihasilkan oleh AI meningkatkan ketergantungan kita pada teknologi. Beralih ke pemrosesan manual, seperti grafik yang sederhana, memerlukan waktu 35 menit dengan pembuatan *ai-generated* dan 56 menit dengan pemrosesan manual, namun Anda mungkin akan kesulitan untuk membiasakan diri dengan proses manual lagi, terutama jika Anda terbiasa dengan hasil otomatis.

Pengujian menggunakan SonarQube

Analisis dilakukan menggunakan SonarQube untuk mengevaluasi kualitas kode dari proyek pengembangan *website* yang diuji. Analisis ini difokuskan pada tiga aspek utama: *maintainability*, dan *code duplications*. Hasil analisis akan dibahas secara rinci untuk masingmasing aspek, serta langkah-langkah yang diambil untuk mengatasi masalah yang teridentifikasi.

1. Manual
   Hasil pengujian *website* secara manual di sonarqube dapat dilihat pada gambar 3.
2. *AI-generated* Hasil pengujian *website* secara *AI-generated* di sonarqube dapat dilihat pada gambar 4.

Analisis Hasil Pengujian

1. *Maintainability*

Table 6. Hasil Analisis *Maintainability*

	Temuan	Analisis
Manual	123 isu terbuka, dengan detail klasifikasi: High: 33 Medium: 17 Low: 73	Pengembangan manual mencatat total 123 isu terbuka, dengan 33 isu "High," 17 "Medium," dan 73 "Low." Persentase isu kritis (26.8%) hampir sama dengan *ai- generated*, tetapi proporsi isu menengah sedikit lebih tinggi (13.8%). Persentase masalah rendah juga lebih tinggi (59.3%), menunjukkan bahwa meskipun banyak isu rendah, pengembangan manual mungkin memerlukan perhatian lebih pada masalah menengah.
AI-generated	103 isu terbuka, dengan detail klasifikasi: High: 27 Medium: 12Low 64	Pengembanganai-generated mencatat total. 103 isu terbuka, dengan 27 isu "High," 12 "Medium," dan 64 "Low. "Persentase isu kritis (26.2%) serupa dengan manual, tetapi proporsi isu menengah lebih rendah (11.7%). Persentase masalah rendah sedikit lebih tinggi (62.1%), menunjukkan bahwa *ai-generated* mungkin lebih baik dalam mengelola isu menengah dan menghasilkan kode dengan kualitas lebih baik.

Table 7. Hasil Analisis *Code Duplications*

TEMUAN		ANALISIS
Manual	4.4% duplikasi	Pengembangan manual menunjukkan tingkat duplikasi kode sebesar 4.4% dari total 8.4 ribu baris kode. Ini berarti bahwa duplikasi kode dalam pengembangan manual relatif rendah, mencerminkan desain kode yang lebih bersih dan terstruktur. Dengan persentase duplikasi yang rendah, kode manual mungkin lebih mudah dikelola dan diperbaiki, mengurangi potensi masalah dalam *maintainability* dan meningkatkan efisiensi pengembangan.
Ai-Generated	11.6% duplikasi kode dalam 5.1 K baris kode	Pengembangan*ai-generated* menunjukkan tingkat duplikasi kode yanglebih tinggi, yaitu 11.6% daritotal 5.1 ribu baris kode. Persentase duplikasi ini hampir tiga kali lipat lebih tinggi dibandingkan dengan pengembangan manual. Meskipun jumlah baris kode yang dihasilkan lebih sedikit, tingkat duplikasi yang tinggi menunjukkan bahwa kode *ai-generated* mungkin lebih repetitif dan memerlukan perhatian tambahan untuk mengurangi redundansi dan meningkatkan kualitas kode.

Perbandingan: Perbandingan antara pengembangan manual dan *ai-generated* pada tabel 6 menunjukkan bahwa pengembangan manual menghasilkan lebih banyak isu terbuka (123) dibandingkan dengan *ai-generated* (103). Meskipun persentase isu kritis hampir sama (26.8% untuk manual dan 26.2% untuk ai-generated), pengembangan manual memiliki proporsi isu menengah yang sedikit lebih tinggi (13.8% dibandingkan dengan 11.7% pada ai- generated). Di sisi lain, *ai-generated* memiliki persentase isu rendah yang sedikit lebih tinggi (62.1% dibandingkan dengan 59.3% pada manual). Hal ini menunjukkan bahwa *ai-generated* mungkin lebih efektif dalam mengelola isu menengah dan menghasilkan kode dengan kualitas lebih baik secara keseluruhan, meskipun jumlah isu total lebih rendah pada pendekatan AI.

Dampak: Isu yang lebih banyak dan beragam dalam pengembangan manual dapat mengakibatkan peningkatan waktu dan usaha dalam pemeliharaan serta perbaikan kode. Proposi isu menengah yang lebih tinggi menunjukkan adanya area yang memerlukan perbaikan yang lebih mendalam, sementara proporsi masalah rendah yang lebih tinggi dapat menunjukkan bahwa meskipun ada banyak masalah kecil, tetap diperlukan perhatian untuk menghindari akumulasi masalah seiring waktu. Di sisi lain, dengan jumlah isu total yang lebih rendah dan proporsi isu menengah yang lebih rendah, aigenerated mungkin memungkinkan pengelolaan dan pemeliharaan yang lebih efisien waktu.

2. *Code Duplications*

Perbandingan: Perbandingan ini menunjukkan bahwa pengembangan manual memiliki tingkat duplikasi kode yang lebih rendah dibandingkan dengan pengembangan *ai-generated*. Hal ini mengindikasikan bahwa meskipun *ai-generated* mungkin lebih cepat dalam menghasilkan kode, ia cenderung menghasilkan lebih banyak duplikasi yang dapat mempengaruhi *maintainability* dan efisiensi kode. Di sisi lain, pengembangan manual, meskipun menghasilkan lebih banyak baris kode, menunjukkan desain kode yang lebih bersih dengan duplikasi yang lebih rendah.

Dampak: Tingkat duplikasi kode yang lebih tinggi dalam *ai-generated* dapat menyebabkan berbagai masalah, termasuk kesulitan dalam pemeliharaan dan peningkatan potensi bug. Duplikasi kode yang tinggi biasanya memerlukan perubahan dan pembaruan yang lebih banyak dan dapat meningkatkan waktu dan biaya pemeliharaan. Hal ini juga dapat membuat kode menjadi lebih sulit untuk dipahami dan dikelola. Sebaliknya, tingkat duplikasi kode yang lebih rendah dalam pengembangan manual menunjukkan kode yang lebih efisien dan mudah dikelola, yang dapat mengurangi waktu dan biaya pemeliharaan serta meningkatkan kualitas dan konsistensi kode secara keseluruhan.

Keterhubungan

Keterhubungan antara hasil evaluasi waktu pengerjaan dan hasil pengukuran SonarQube menunjukkan: Kecepatan Pengembangan *AI-generated*: Kecepatan pengembangan yang dihasilkan *ai-generated* lebih cepat, namun harus diimbangi dengan kualitas kode, terutama duplikasi dan kemampuan pemeliharaan kode. Tingkat duplikasi yang tinggi dalam kode yang dihasilkan *ai-generated* dapat memengaruhi kualitas kode secara keseluruhan, meskipun ada efisiensi dalam waktu pemrosesan. Kualitas Kode Manual: Pengembangan manual memakan waktu, tetapi menghasilkan kode dengan tingkat duplikasi yang lebih rendah, yang menunjukkan desain yang lebih bersih dan terstruktur. Hal ini meningkatkan kemudahan pemeliharaan dan memungkinkan pemeliharaan yang lebih efisien dalam jangka panjang. Secara keseluruhan, generasi *ai-generated* memungkinkan pengembangan yang lebih cepat, namun penting untuk mempertimbangkan dampaknya terhadap kualitas kode, terutama dalam hal duplikasi kode. Pengembangan manual memakan waktu, tetapi memberikan kode yang lebih bersih dan mudah dipelihara. Oleh karena itu, untuk mencapai hasil optimal dalam proyek pengembangan perangkat lunak, Anda harus mempertimbangkan keseimbangan antara efisiensi waktu dan kualitas kode saat memilih metode pengembangan.

Kesimpulan

Berdasarkan hasil analisis, dapat disimpulkan sebagai berikut:

1. Efisiensi waktu yang dihasilkan ai-generated
   Berdasarkan Tabel 3 terlihat jelas bahwa pendekatan yang dihasilkan *ai-generated* lebih efisien dalam hal waktu pemrosesan dibandingkan dengan cara manual. Misalnya, registrasi pengguna manual dan login pengguna masing-masing membutuhkan waktu 23 menit untuk diproses, namun pembuatan *ai-generated* secara signifikan mengurangi waktu tersebut menjadi 6 menit. Demikian pula, tugas kompleks seperti pemrosesan data CB yang memakan waktu 2 jam 16 menit secara manual dapat dikurangi menjadi 50 menit dengan bantuan yang dihasilkan *ai-generated*.
2. Dampak penggunaan *ai-generated* terlebih dahulu
   Tabel 5 menunjukkan bagaimana penggunaan aigenerated generasi pertama sebelum pekerjaan manual berdampak pada pengalaman dan efisiensi kerja. Misalnya, pembuatan data tabular yang dihasilkan *ai-generated* memerlukan waktu 30 menit, sedangkan tugas yang sama memerlukan waktu 50 jam untuk dibuat secara manual. Meskipun proses yang dihasilkan ai- generated menjadi lebih cepat, transisi ke proses manual seringkali bergantung pada teknologi, yang dapat menimbulkan kesulitan dan kebingungan ketika kembali ke proses manual. Pekerjaan di mana *ai-generated* secara manual menghasilkan grafik sederhana dalam 56 menit yang memakan waktu 35 menit menggambarkan kegagalan mengadaptasi teknik manual setelah terbiasa dengan efisiensi *ai-generated*.
3. Kebingungan dan Ketergantungan
   Tingginya penggunaan yang dihasilkan oleh *ai-generated* dapat menyebabkan ketergantungan pada teknologi, sehingga beralih ke cara manual nantinya dapat menyebabkan kebingungan dan kesulitan. Perbedaan signifikan dalam waktu dan efisiensi proses dapat memengaruhi cara penyelesaian tugas manual, sehingga menyebabkan ketidaknyamanan dan mengurangi produktivitas selama kalibrasi ulang.
4. *Maintainability*
   Pengembangan manual menghasilkan 123 isu terbuka dengan proporsi isu kritis dan menengah yang hampir sama dengan AIgenerated. Namun, proporsi isu menengah pada manual sedikit lebih tinggi (13.8% dibandingkan dengan 11.7% pada AI-generated), sementara isu dengan tingkat keparahan rendah sedikit lebih rendah (59.3% pada manual versus 62.1% pada AIgenerated). Ini menunjukkan bahwa *AI-generated* mungkin lebih efektif dalam menangani isu menengah dan menghasilkan kode yang lebih bersih secara keseluruhan, meskipun jumlah total isu lebih rendah pada metode AI.
5. *Code Duplications*
   pengembangan manual menunjukkan tingkat duplikasi yang lebih rendah (4.4% dari 8.4 ribu baris kode), mencerminkan desain kode yang lebih bersih dan terstruktur. Sebaliknya, pengembangan *AI-generated* memiliki tingkat duplikasi kode yang lebih tinggi (11.6% dari 5.1 ribu baris kode), hampir tiga kali lipat lebih tinggi dari manual. Ini menunjukkan bahwa meskipun AIgenerated lebih cepat dalam menghasilkan kode, ia menghasilkan lebih banyak kode yang duplikat, yang dapat mempengaruhi *maintainability* dan efisiensi kode. Secara keseluruhan, meskipun pengembangan manual menghasilkan lebih banyak isu terbuka dan baris kode, ia memiliki tingkat duplikasi kode yang lebih rendah dan mungkin lebih mudah dikelola. AIgenerated, di sisi lain, menawarkan kecepatan pengembangan yang lebih tinggi tetapi dengan tantangan tambahan dalam mengelola duplikasi kode dan isu menengah.
6. Efisiensi Waktu dan *Maintainability*
   Meskipun AIgenerated mempercepat proses pengembangan, hasil analisis SonarQube menunjukkan bahwa pengembangan manual menghasilkan kode dengan duplikasi yang lebih rendah dan masalah *maintainability* yang lebih terkelola. Ini menegaskan bahwa kecepatan pengembangan yang lebih tinggi tidak selalu diimbangi dengan kualitas kode yang sama baiknya. *AI-generated* mungkin lebih cepat, tetapi kode yang dihasilkan memerlukan perhatian lebih dalam pemeliharaan.

Daftar Pustaka

1. Sintaro S, Pandiangan D, Nainggolan N, Johanes AB, Ramadhanty A, Gobel V, et al. Pembuatan Website Sebagai Media Informasi Digital pada Biovina Herbal. Journal of Social Sciences and Technology for Community Service. 2023;4(2).
2. RevoU. Generative AI; 2023. Accessed: 2026-06-10. Available from: `https://revou.co/kosakata/generative-ai`.
3. Amazon Web Services. What is Generative AI?; 2023. Accessed: 2026-06-10. Available from: `https://aws.amazon.com/id/what-is/generative-ai/`.
4. Rahmawati AF, Susetyo YA. Analisis Quality Code Menggunakan SonarQube dalam Suatu Aplikasi Berbasis Laravel. IT-Explore: Jurnal Penerapan Teknologi Informasi dan Komunikasi. 2023;2(2):99-103.
5. Leitner D. A Model for Measuring Maintainability Based on Source Code Characteristics. Vienna: University of Applied Sciences Technikum Wien; 2017.
6. Radatz J, Geraci A, Katki F. IEEE Standard Glossary of Software Engineering Terminology; 1990.
7. Wettel R, Marinescu R. Archeology of Code Duplication: Recovering Duplication Chains from Small Duplication Fragments. In: Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2005). IEEE; 2005. .
8. Marcilio D, Bonifacio R, Monteiro E, Canedo E, Luz W, Pinto G. Are Static Analysis Violations Really Fixed? A Closer Look at Realistic Usage of SonarQube. In: IEEE International Conference on Program Comprehension (ICPC); 2019. p. 209-19.