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

Aldiansyah[1], Dana Sulistyo [2]

[1,2]Fakultas Informatika, Universitas Telkom, Bandung
[1]aaldishantuy@student.telkomuniversity.ac.id,
[2]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.

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

Abstract

This research compares the web application development process with manual and AI-generated approaches, focusing on speed, ease of maintenance, and code duplication. The main objective of the study was to evaluate the time efficiency, maintainability, and code duplication rate of both approaches. The methodology used included manual and AI-assisted web application creation, followed by time testing and code analysis using SonarQube. The application creation process consisted of registration, login, dashboard, and data selection. The results showed that the AI-generated approach was significantly faster in code generation compared to the manual approach. However, although AI-generated speeds up the process, the generated code tends to have a higher duplication rate and more issues to fix. In contrast, the manual approach, while taking longer, generates code with lower duplication and is easier to maintain. The conclusions of this study emphasize the importance of considering the balance between time efficiency and code quality when choosing a software development method. This research provides insight into the impact of using AI in software development and how the transition between AI-generated and manual approaches can affect the development process.

Keywords: Web Application Development, AI-generated Code, Manual vs AI-generated Method, Time Efficiency, Maintainability, Code Duplication, SonarQube, Static Code Analysis, Software Development Comparison.

1. Pendahuluan

Latar Belakang

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 topi*k yang penting karena aplikasi *website memainkan p*eran 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, *mas*ih 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 topi*k yang penting karena aplikasi *website* memainkan *p*eran 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 pa*da pembuatan fitur User Register, User Login, Dashboard, Switch CB Page, dan Switch LBS 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 back-end, 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.

Perumusan Masalah

Berdasarkan dengan latar belakang di atas, Perumusan Masalah pada penelitian adalah sebagai berikut:

1. Bagaimana perbandingan antara pembuatan *website secara manu*al dan penggunaan AI-Generate dalam hal efisiensi waktu, baik dari segi waktu?
2. Sejauh mana *maintainability* (kemudahan pemeliharaan) dari aplikasi *website yang dikemb*angkan secara manual dibandingkan dengan yang dihasilkan oleh *ai-generated*?
3. Bagaimana tingkat *duplications* (duplikasi kode) yang terjadi pada kode aplikasi *website yang dikemb*angkan secara manual dibandingkan dengan yang dihasilkan menggunakan *ai-generated*?

Tujuan

Tujuan penelitian yang menjadi fokus dalam pengembangan ini adalah sebagai berikut.

1. Menilai kecepatan dan efisiensi waktu pembuatan *website menggunakan ai-generated* dibandingkan dengan pendekatan manual dan Mengidentifikasi potensi penghematan waktu.
2. Membandingkan kemudahan pemeliharaan antara aplikasi *website yang dikemb*angkan manual dan yang dihasilkan oleh *ai-generated*, dengan fokus pada kemudahan perbaikan dan pembaruan.
3. Mengukur dan membandingkan tingkat duplikasi kode dalam aplikasi *website yang dikemb*angkan manual versus yang dihasilkan *ai-generated*, dan dampaknya terhadap kualitas kode.

Organisasi Tulisan

Penelitian ini disusun sebagai berikut. Di Bagian 2, membahas secara singkat ilmu atau studi terkait penelitian ini. Pada Bagian 3, adalah penjelasan mengenai perancangan sistem dan alur proses pengembangan yang dilakukan. Pada Bagian 4, bagian yang membahas pengembangan dan evaluasi penelitian. Terakhir, kesimpulan dan saran untuk pengembangan selanjutnya dijelaskan di Bagian 5.

2. Studi Terkait

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 ad*alah aplikasi yang dapat dijalankan dengan menggunakan *website browser, sa*at ini hampir semua gawai dapat menjalankan *website browser yan*g menyebabkan *website dapat dibuk*a di hampir semua gawai yang ada[1]. *Website adalah wada*h digital yang terdiri dari halaman-halaman saling terhubung yang dapat diakses melalui internet. Setiap *website memiliki do*main 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 s*ebagai 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.[2]
Menurut Goldman Sachs, *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, *ai-generated* 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[6]. Alat ini menggunakan aturan analisis yang dapat disesuaikan untuk mengevaluasi berbagai aspek dari kode, termasuk struktur, kompleksitas, dan kepatuhan terhadap standar pengkodean.

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.

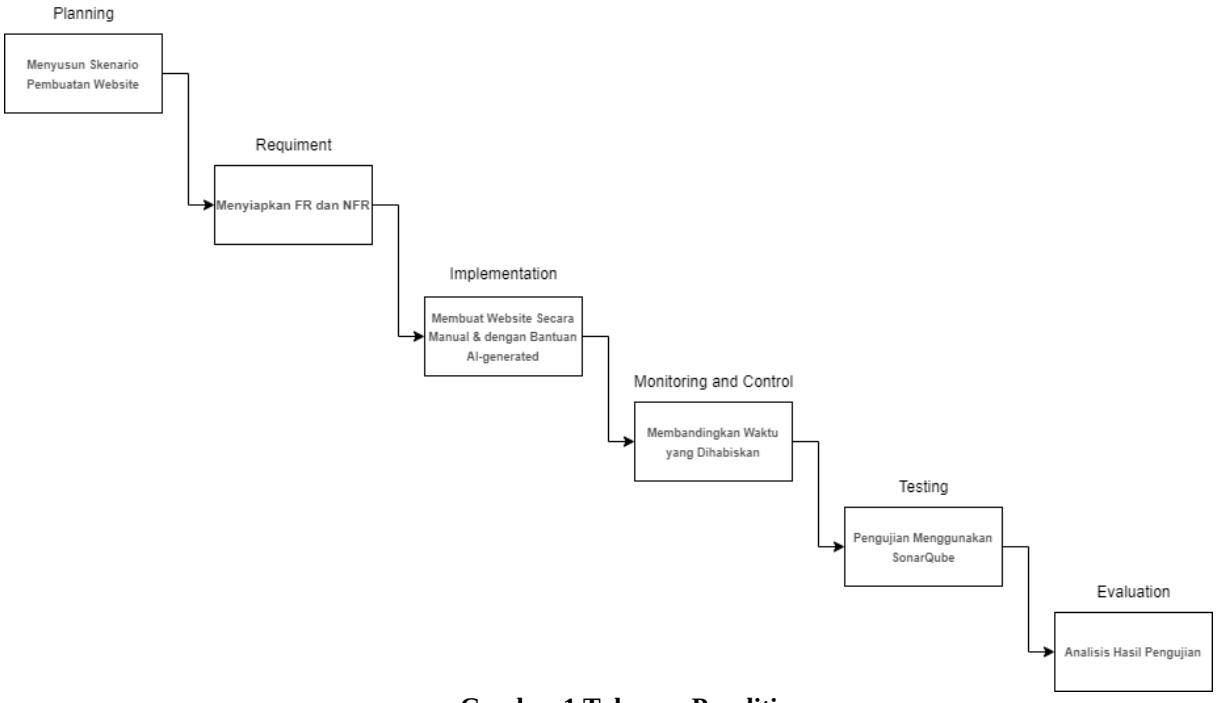
Maitainability

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[7]. pemeliharaan adalah properti sistem perangkat lunak sementara pemeliharaan didefinisikan sebagai proses yang sesuai setelah pengiriman [8]. 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[9]. 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.

3. Metodologi Penelitian

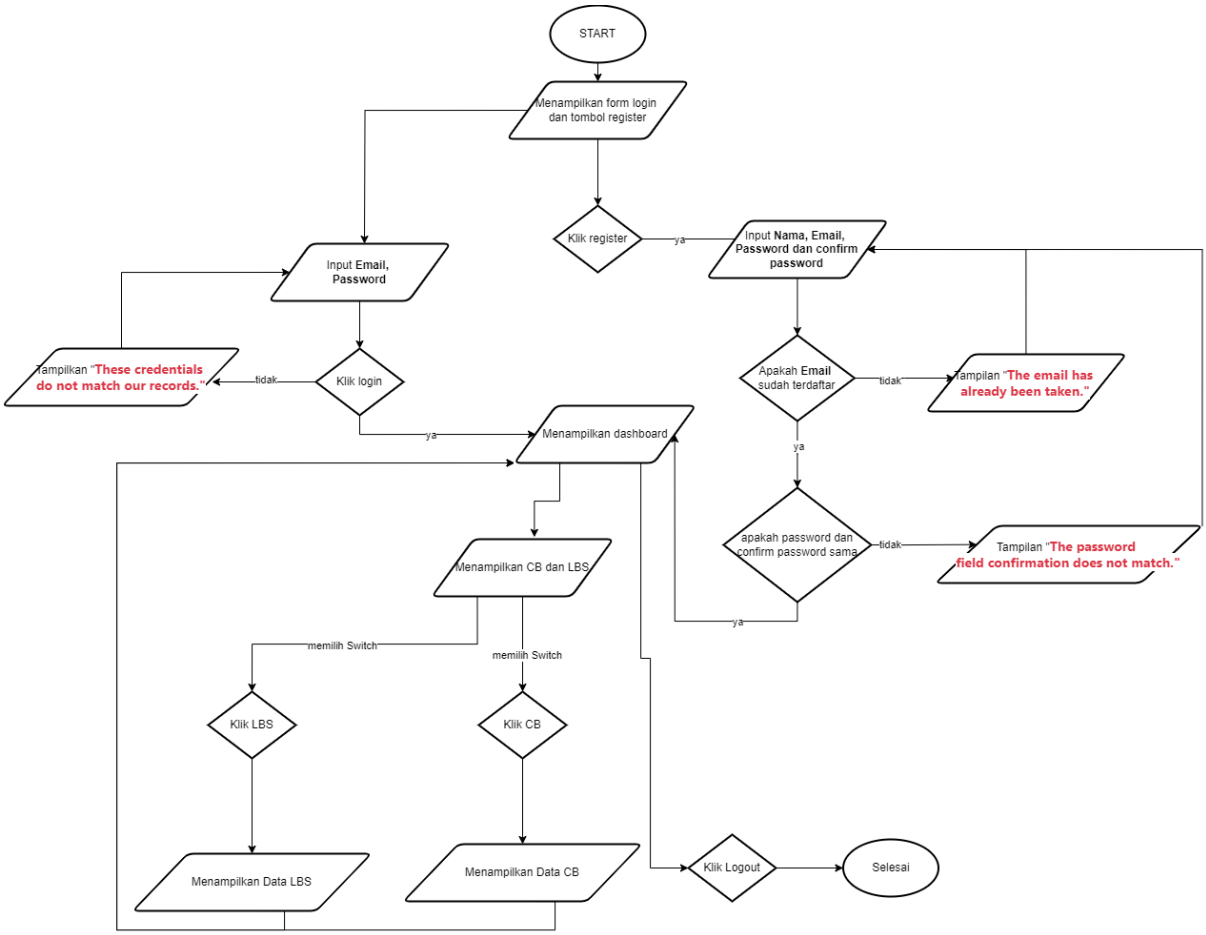


Gambar 1 Tahapan Penelitian

Gambar 1 menggambarkan tahapan penelitian yang di mulai pembuatan *website lalu pengu*jian code statik. Terdapat 6 proses pada penelitian ini, di mulai dari menyiapkan scenario, pembuatan *website secara manu*al dan di bantu dengan ai-generate, membandingkan waktu yang di habiskan, pengujian menggunakan sonarqube dan menganalisis hasil dari pengujian sonarqube.

3.1 Skenario Pembuatan *Website*

Gambar 2 Skenario Pembuatan Website



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

3.2 Program *Website*

*Program vis*ualisasi 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.

3.3 Data Requirement

3.3.1 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 berikut:

Tabel 1 Identifikasi Use Case

Use Case	Nama Use Case	Deskripsi
UC-1	User Register	Pengguna melakukan login dengan menginput email dan password.
UC-2	User Login	Pengguna baru dapat mendaftar dengan memasukkan nama, email, 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.

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.

Tabel 2 Tambahan Use Case

Usecase	Nama Use Case	Deskripsi
UC-7	Data Tabel	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.

Pendekatan manual memberikan kontrol 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.

3.3.2 non-Functional Requirement

Di bawah ini, kami menyajikan tabel yang menjelaskan non-functional requirement sebagai berikut:

Tabel 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.

3.4 Evaluasi

Evaluasi dilakukan menggunakan Stopwatch dan pengujian sonarqube. Stopwatch di gunakan untuk menghitung pengerjaan setiap tugas pembuatan *website dan pengujia*n 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.

4. Evaluasi

4.1 Perbandingan Waktu

Tabel di bawah menunjukkan waktu pemrosesan untuk setiap tugas yang dilakukan menggunakan pendekatan manual dan pendekatan yang dihasilkan *ai-generated*. Awalnya menggunakan tenaga kerja manual, namun kemudian dilengkapi dengan bantuan *ai-generated*. Hal ini memungkinkan Anda mengevaluasi efisiensi waktu penggunaan *ai-generated* dalam mempercepat proses pengembangan.

Tabel 4 Waktu pengerjaan setiap tugas

Usecase	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 jam 16 menit	50 Menit
5	Switch LBS Page	16 Menit	10 menit

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

Tabel di bawah ini membandingkan waktu pemrosesan untuk berbagai tugas yang dilakukan pertama kali menggunakan metode yang dihasilkan *ai-generated* 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.

Tabel 5 waktu pengerjaan *ai-generated* lebih dahulu

Usecase	Task	Ai-Generate	Manual
1	Data Tabel	30 Menit	50 Jam

2	Grafik	35 Menit	56 Menit
3	Simple User Greeting	7 Menit	15 Menit

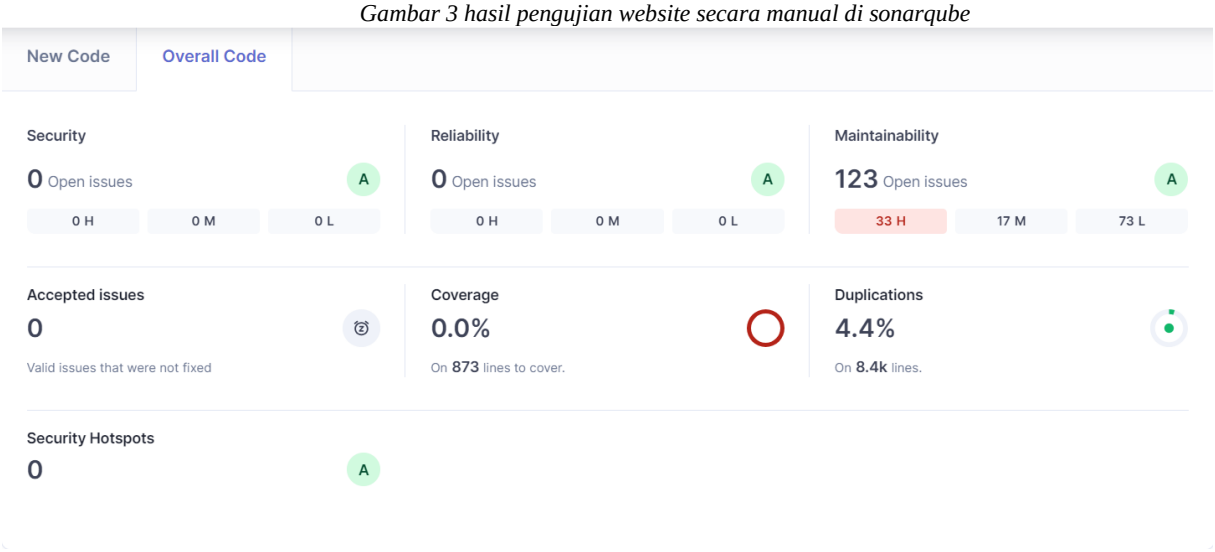
Pertama, penggunaan sesuatu yang dihasilkan oleh AI meningkatkan ketergantungan kita pada teknologi. Beralih ke pemrosesan manual, seperti grafik yang sederhana, memerlukan waktu 35 menit dengan pembuatan *ai-generated* dan 56 menit dengan pemrosesan manual, namun Anda mungkin akan kesulitan untuk membiasakan diri dengan proses manual lagi, terutama jika Anda terbiasa dengan hasil otomatis.

4.2 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 masing-masing aspek, serta langkah-langkah yang diambil untuk mengatasi masalah yang teridentifikasi.

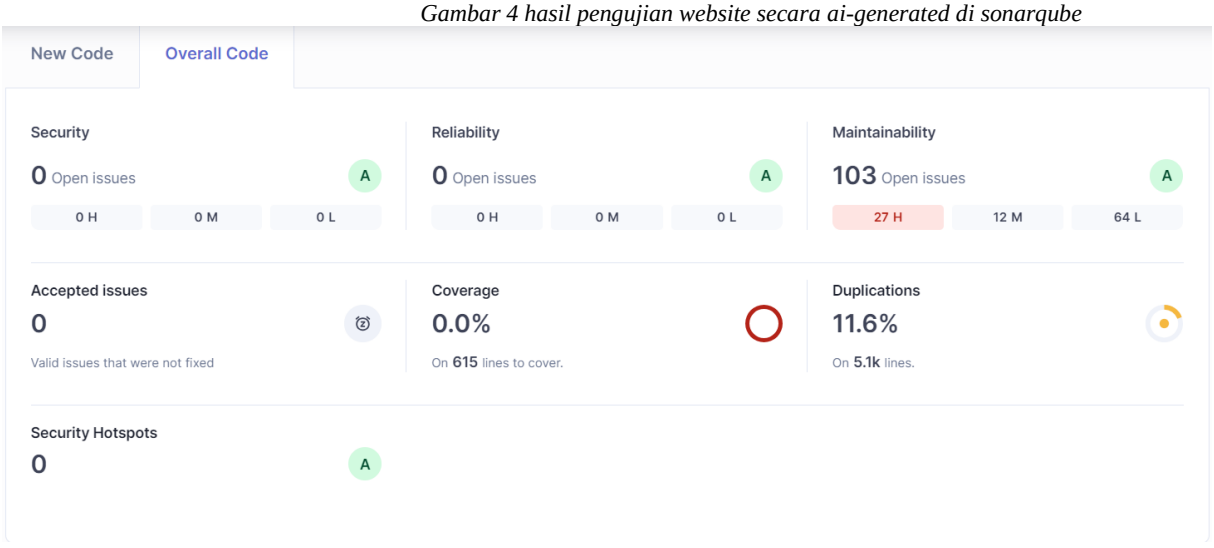
4.2.1 Manual

Gambar 3 hasil pengujian website secara manual di sonarqube



4.2.2 AI-generated

Gambar 4 hasil pengujian website secara ai-generated di sonarqube



4.3 Analisis Hasil Pengujian

A. Maintainability

Tabel 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: 12 Low 64	Pengembangan *ai-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.

Perbandingan: Perbandingan antara pengembangan manual dan *ai-generated* 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, *ai-generated* mungkin memungkinkan pengelolaan dan pemeliharaan yang lebih efisien waktu.

B. Code Duplications

Tabel 7 hasil analisis Code duplications

	TEMUAN	ANALISIS
MANUAL	4.4% duplikasi kode dalam 8.4K baris kode.	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 5.1K baris kode	Pengembangan *ai-generated* menunjukkan tingkat duplikasi kode yang lebih tinggi, yaitu 11.6% dari total 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 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.

4.4 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.

5 Kesimpulan

Berdasarkan hasil analisis, dapat disimpulkan sebagai berikut:

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*.

Dampak penggunaan *ai-generated* terlebih dahulu : Tabel 4 menunjukkan bagaimana penggunaan *ai-generated* 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*.

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.

Maintainability: pengembangan manual menghasilkan 123 isu terbuka dengan proporsi isu kritis dan menengah yang hampir sama dengan AI-generated. 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 AI-generated). 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.

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 AI-generated 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. AI-generated, di sisi lain, menawarkan kecepatan pengembangan yang lebih tinggi tetapi dengan tantangan tambahan dalam mengelola duplikasi kode dan isu menengah.

Efisiensi Waktu dan *Maintainability*: Meskipun AI-generated 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.

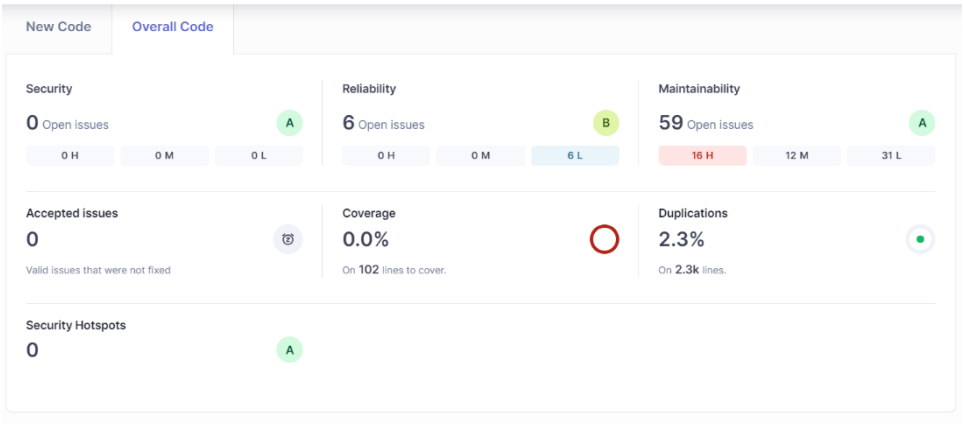
Daftar Pustaka

[1]Sintaro, S., Pandiangan, D., Nainggolan, N., Johanes, A. B., Ramadhanty, A., Gobel, V., Putri, V., & Nainggolan, G. (2023). Pembuatan *Website Sebagai Medi*a Informasi Digital pada Biovina Herbal. Journal of Social Sciences and Technology for Community Service, 4(2). https://doi.org/10.33365/jsstcs.v4i2.3354
[2]Revo. (2023). Retrieved from Revo: https://revou.co/kosakata/generative-ai.
[3]AWS. (2023). Retrieved from aws: https://aws.amazon.com/id/what-is/generative-ai/. Https://Aws.Amazon.Com/Id/What-Is/Generative-Ai/.
[5] D. Marcilio, R. Bonifacio, E. Monteiro, E. Canedo, W. Luz, and G. Pinto, "Are static analysis violations really fixed? a closer look at realistic usage of sonarqube," in IEEE International Conference on Program Comprehension, May 2019, vol. 2019-May, pp. 209–219. doi: 10.1109/ICPC.2019.00040.
[6] Rahmawati, A. F., & Susetyo, Y. A. (2023). ANALISIS QUALITY CODE MENGGUNAKAN SONARQUBE DALAM SUATU APLIKASI BERBASIS LARAVEL. *IT-Explore: Jurnal Penerapan Teknologi Informasi Dan Komunikasi*, *2*(2), 99–103. https://doi.org/10.24246/itexplore.v2i2.2023.pp99-103
[7] Leitner, David. "A Model for Measuring Maintainability Based on Source Code Characteristics." *University of Applied Sciences Technikum Wien, Vienna* (2017).
[8] Jane Radatz, Anne Geraci, and Freny Katki. "IEEE Standard Glossary of Software Engineering Terminology". In: IEEE Std 610.12-1990 (Dec. 1990), pp. 1–84.
[9] Wettel, Richard, and Radu Marinescu. "Archeology of code duplication: Recovering duplication chains from small duplication fragments." *Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC'05)*. IEEE, 2005.

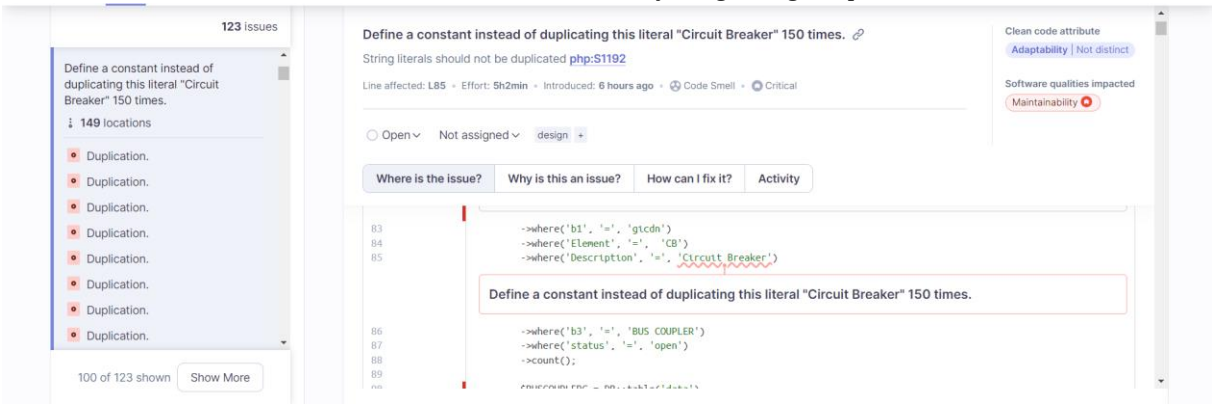
Lampiran

Gambar 5 tampilan *website*

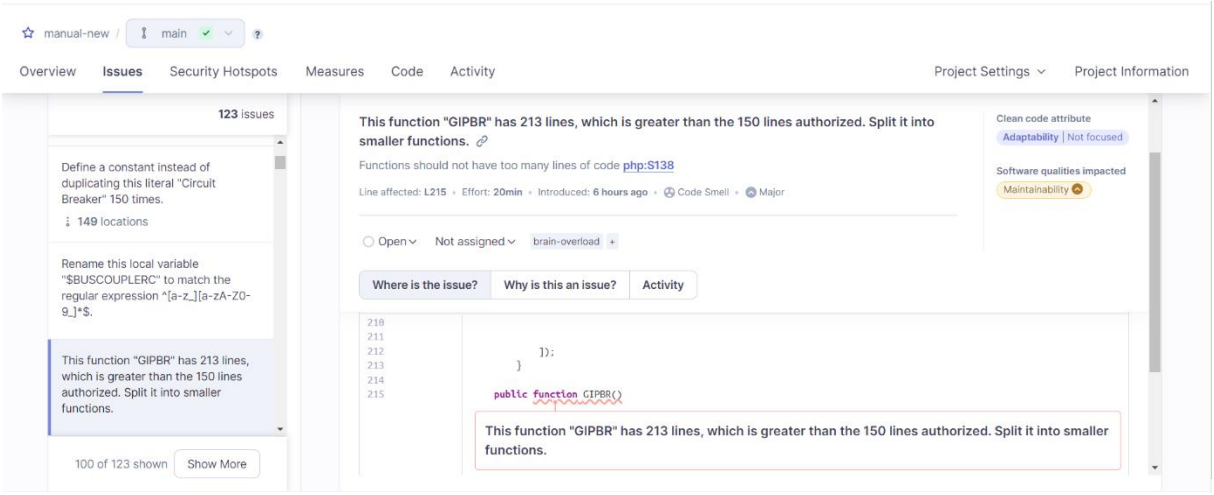
Gambar 6 Pengujian Default Laravel



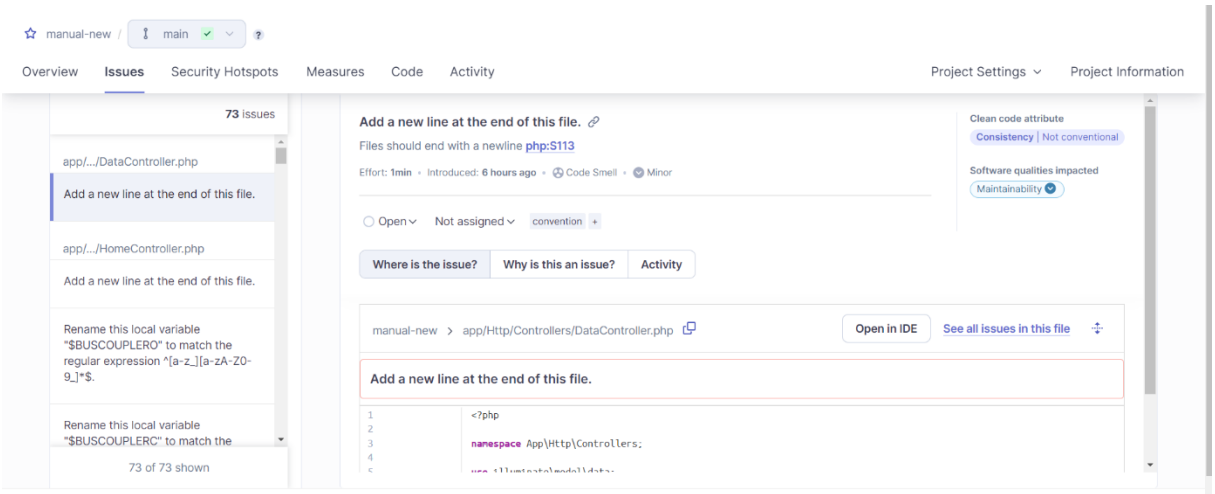
Gambar 7 Salah satu maintaibility dengan High impact di manual



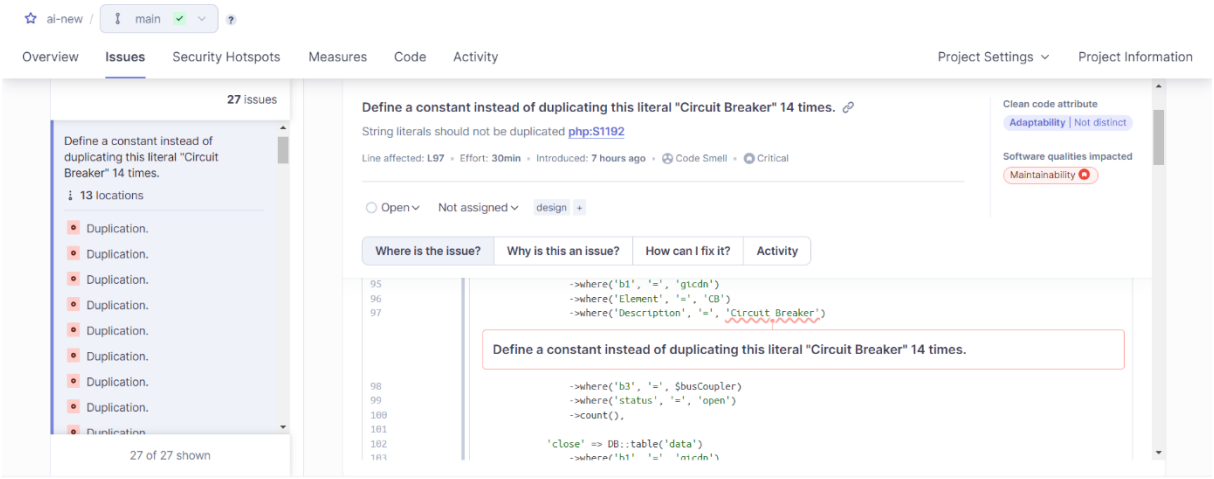
Gambar 8 Salah satu maintaibility dengan medium impact manual



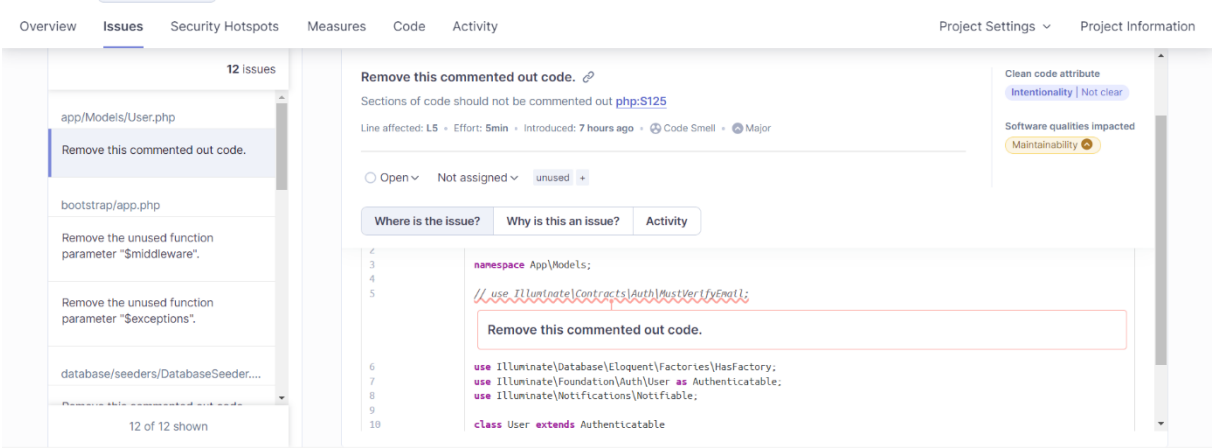
Gambar 9 Salah Satu maintaibility dengan low impact manual



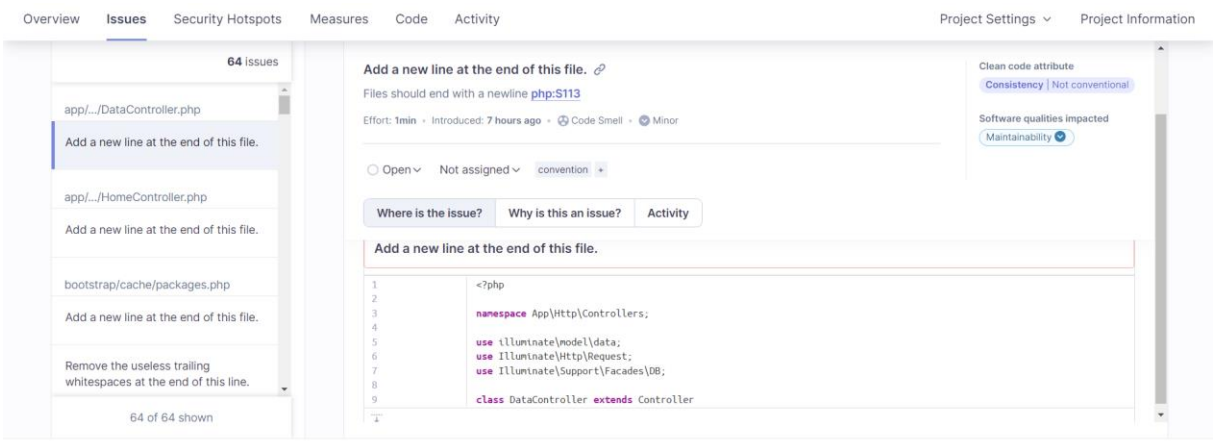
Gambar 10 Salah satu maintainability high impact ai-generate



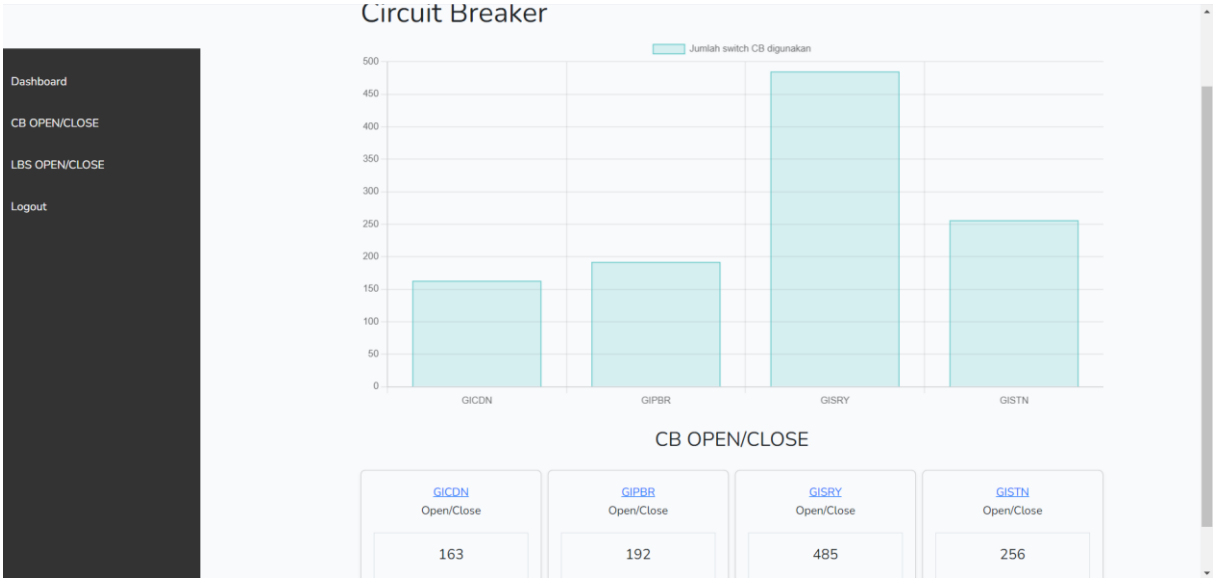
Gambar 11 Salah satu maintainability medium impact ai-generate



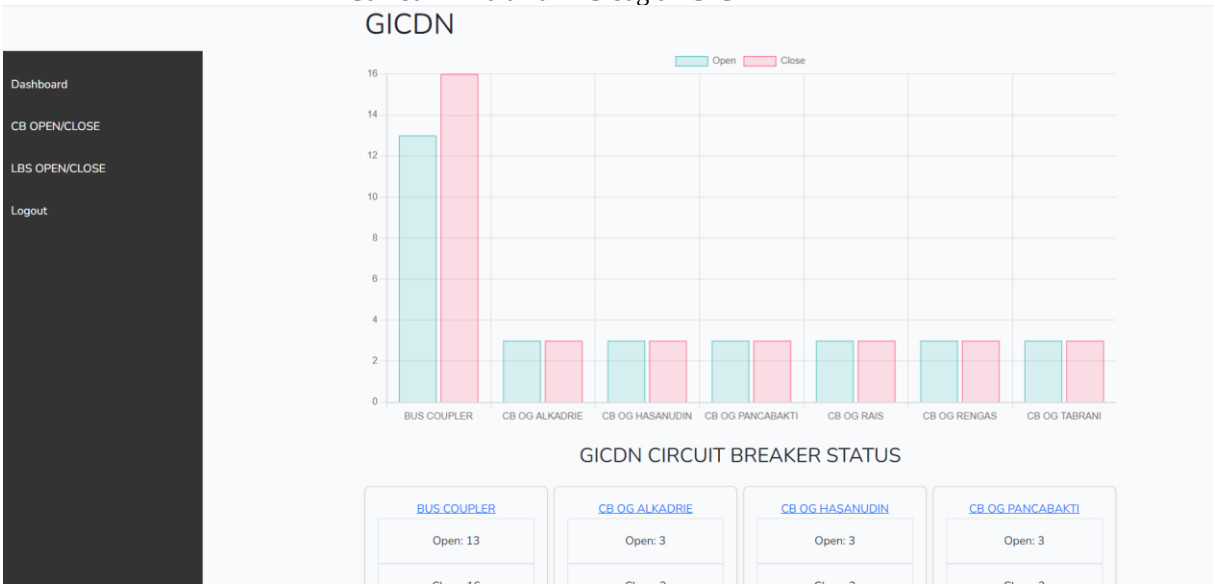
Gambar 12 Salah satu maintainability low impact ai-generate



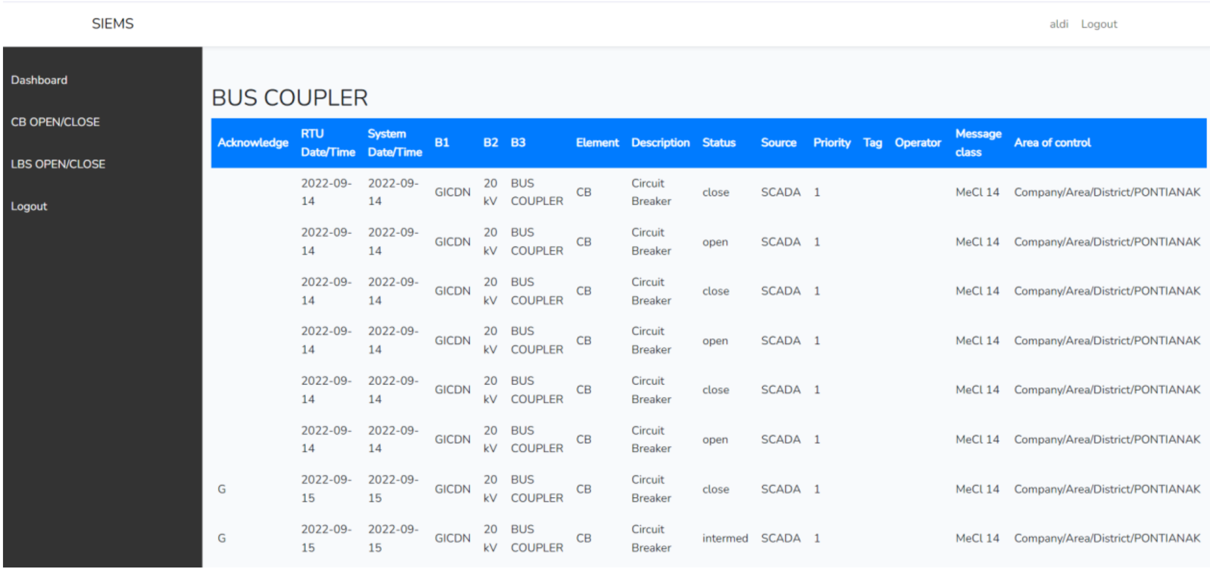
Gambar 13 Halaman CB



Gambar 14 halaman PG bagian GICDN



Gambar 15 Tabel data CB OG ALKADRIE



Gambar 16 Halaman LBS