JURNAL ILMIAH SISTEM INFORMASI (JUSI)
VOL 5, No. 3, November 2026, pp. 417-432
p-ISSN : 2803-1507 (media cetak) | e-ISSN : 2803-1531 (media online)
Link Publikasi: https://ejurnal.provisi.ac.id/index.php/JUISI/issue/view/175

Maintainability Evaluation of Geo-Tagged Mobile Attendance Application Using Pragmatic Clean Architecture

Raka Putra Pratama*[1], Mamok Andri Senubekti[2], Novita Anggraini[3]
[1,2,3] Program Studi Informatika, Universitas Teknologi Digital
Jl. Cibogo Indah III, 40292, Bandung, Jawa Barat, Indonesia
e-mail*[1]: raka20125132@digitechuniversity.ac.id

ARTICLE INFO
Article History:

Received: 20-06-2026
Revised: 21-07-2026
Accepted: 30-08-2026
Published: 28-10-2026

ABSTRACT
The fingerprint-based attendance system at Universitas Teknologi Digital still faces operational limitations, including physical queues, manual monthly recapitulation, limited attendance points, and potential inconsistencies in attendance records. Mobile attendance can improve efficiency, but it also introduces location spoofing risks and maintainability issues when developed using a monolithic structure. This study aims to implement a geo-tagged mobile attendance application using Pragmatic Clean Architecture and evaluate its maintainability compared with a monolithic prototype. The research applies Agile Scrum, object-oriented design, native Android implementation, Progressive Web App access, server-side validation, anti-spoofing testing, and static code analysis using Android Studio Narwhal on Java source code. Location validation is supported through Android mock-location detection, heuristic validation on the PWA layer, and server-side validation. In a limited anti-spoofing test using Fake GPS, the system rejected or marked suspicious locations in 5 out of 5 attempts on both native Android and PWA access, although PWA validation required a longer checking process and remains heuristic. Maintainability evaluation shows that the average Lines of Code per method decreased from 36.26 to 16.85, equivalent to a 53.5% reduction. The average Cyclomatic Complexity decreased from 2.32 to 1.78, equivalent to a 23.27% reduction and remaining in a low-risk category. These findings indicate that the proposed system improves attendance validation and provides an initial indication of a more analyzable, modifiable, and testable code structure. However, LOC and Cyclomatic Complexity are treated as initial maintainability indicators, not a complete representation of all ISO/IEC 25010 maintainability sub-characteristics.
Keywords: *Clean Architecture, Geo-tagging, Maintainability, Mobile Attendance, Software Quality.*

Abstrak
Sistem presensi berbasis *fingerprint* di Universitas Teknologi Digital masih menghadapi keterbatasan operasional, seperti antrean fisik, rekapitulasi bulanan manual, keterbatasan titik presensi, dan potensi ketidaksesuaian data kehadiran. Sistem presensi mobile dapat meningkatkan efisiensi, tetapi juga membawa risiko manipulasi lokasi serta masalah maintainability apabila dikembangkan menggunakan struktur monolitik. Penelitian ini bertujuan mengimplementasikan aplikasi presensi mobile berbasis geo-tagging dengan *Pragmatic Clean Architecture* serta mengevaluasi maintainability-nya dibandingkan prototipe monolitik. Metode yang digunakan meliputi Agile Scrum, desain berorientasi objek, *implementasi Android native*, akses *Progressive Web App*, validasi sisi server, pengujian *anti-spoofing*, dan *static code analysis* menggunakan *Android Studio Narwhal* pada kode sumber Java. Validasi lokasi didukung oleh deteksi mock location pada Android, validasi heuristik pada PWA, dan validasi sisi server. Pada pengujian anti-spoofing terbatas menggunakan *Fake GPS*, sistem berhasil menolak atau menandai lokasi mencurigakan pada 5 dari

DOI: https://doi.org/10.51903/ykfts924

p-ISSN : 2803-1507 e-ISSN : 2803-1531

5 percobaan pada jalur *Android native* dan PWA, meskipun validasi PWA membutuhkan waktu pengecekan lebih lama dan tetap bersifat heuristik. Evaluasi *maintainability* menunjukkan rata-rata *Lines of Code* per *method* menurun dari 36,26 menjadi 16,85 atau sebesar 53,5%. Rata-rata *Cyclomatic Complexity* juga menurun dari 2,32 menjadi 1,78 atau sebesar 23,27% dan berada pada kategori *low risk*. Temuan ini menunjukkan bahwa sistem usulan meningkatkan validasi data presensi sekaligus memberikan indikasi awal terhadap struktur kode yang lebih mudah dianalisis, dimodifikasi, dan diuji. Namun, LOC dan *Cyclomatic Complexity* diposisikan sebagai indikator awal *maintainability*, bukan representasi penuh dari seluruh subkarakteristik maintainability dalam ISO/IEC 25010.

Kata Kunci: *Clean Architecture, Geo-tagging, Maintainability, Presensi Mobile, Kualitas Perangkat Lunak.*

1. PENDAHULUAN

Perkembangan teknologi informasi mendorong perguruan tinggi untuk melakukan digitalisasi proses administrasi sumber daya manusia, termasuk pencatatan presensi karyawan. Data presensi bukan hanya digunakan sebagai bukti kehadiran, tetapi juga menjadi dasar evaluasi kedisiplinan, rekapitulasi bulanan, dan pengambilan keputusan manajerial. Penelitian terkait menunjukkan bahwa digitalisasi sistem informasi internal, perencanaan strategis sistem informasi, monitoring prosedur organisasi, dukungan kualitas jaringan, serta perancangan UI/UX menjadi aspek pendukung dalam keberhasilan layanan digital institusi [1], [2], [21], [24]. Sistem presensi berbasis mobile dan *geolocation* semakin banyak dikembangkan karena dapat mengurangi ketergantungan pada perangkat fisik, mempercepat pencatatan, dan meningkatkan aksesibilitas pengguna [14], [17]. Namun, transformasi dari sistem *fingerprint* menuju sistem mobile tidak cukup hanya dilakukan dari sisi fungsi, karena validitas lokasi dan kualitas struktur kode juga perlu diperhatikan.

Universitas Teknologi Digital masih menggunakan sistem presensi berbasis mesin *fingerprint* untuk karyawan. Sistem tersebut telah membantu proses pencatatan kehadiran, tetapi memiliki keterbatasan karena hanya tersedia dua unit mesin pada dua lokasi kampus. Kondisi ini menimbulkan potensi antrean pada jam masuk dan pulang kerja, sementara pengolahan data kehadiran bulanan masih memerlukan rekapitulasi manual yang dapat berlangsung hingga satu minggu. Selain itu, mesin *fingerprint* yang hanya mengandalkan satu jari dan menyediakan alternatif input kode manual memiliki risiko manipulasi kehadiran apabila tidak diawasi secara memadai. Permasalahan operasional tersebut menunjukkan bahwa sistem presensi berbasis perangkat fisik belum sepenuhnya menjawab kebutuhan aksesibilitas dan efisiensi data secara *real-time*.

Sebelum penelitian ini dilakukan, telah tersedia prototipe aplikasi presensi *mobile* berbasis *QR Code*. Namun, pendekatan *QR Code* memiliki kelemahan karena membutuhkan kode fisik yang dicetak dan ditempelkan di lokasi presensi. Kode fisik tersebut rentan rusak, hilang, difoto, atau dibagikan secara tidak sah sehingga fleksibilitas aplikasi *mobile* tidak sepenuhnya tercapai. Selain itu, prototipe awal dibangun dengan struktur monolitik yang mencampurkan logika antarmuka, proses data, dan komunikasi *server* dalam satu alur kode. Kondisi ini menimbulkan *technical debt* dan mempersulit pengembangan fitur baru. Studi terkait menunjukkan bahwa struktur kode, modularitas, dan *code smells* memiliki hubungan erat dengan *maintainability* aplikasi, khususnya pada aplikasi Android [3], [6], [7].

Solusi yang diusulkan dalam penelitian ini adalah pembangunan ulang aplikasi presensi *mobile* menggunakan *geo-tagging evidence* dan *Pragmatic Clean Architecture*. *Geo-tagging* digunakan untuk menyertakan informasi koordinat perangkat sebagai bukti pendukung presensi, sedangkan validasi utama dilakukan melalui *server* untuk menjaga konsistensi aturan presensi. Ancaman *location spoofing* dimitigasi melalui deteksi *mock location* pada aplikasi *native* Android dan pendekatan heuristik pada PWA. Hal ini relevan dengan kajian keamanan lokasi yang menunjukkan bahwa data *Global Navigation Satellite System* (GNSS) rentan terhadap spoofing apabila tidak dilengkapi mekanisme validasi tambahan [8], [10].

Dari sisi rekayasa perangkat lunak, penelitian ini tidak hanya berfokus pada keberhasilan fungsi presensi, tetapi juga pada kualitas internal kode. Standar ISO/IEC 25010 digunakan sebagai kerangka evaluasi kualitas perangkat lunak, khususnya karakteristik *maintainability* yang meliputi analyzability, modifiability, dan testability [11], [13]. Evaluasi dilakukan menggunakan *static code analysis* dengan metrik *Lines of Code* (LOC) dan *Cyclomatic Complexity* (CC). Gap penelitian ini terletak pada integrasi antara validasi presensi berbasis geo-tagging, distribusi akses Android *native* dan PWA, serta evaluasi *maintainability* berbasis metrik kuantitatif pada aplikasi presensi *mobile* kampus.

Tujuan penelitian ini adalah merancang dan mengimplementasikan aplikasi presensi *mobile* karyawan dengan pendekatan *Pragmatic Clean Architecture*, menerapkan *geo-tagging evidence* untuk meningkatkan validitas data presensi, dan mengevaluasi *maintainability* aplikasi dibandingkan pendekatan monolitik.

p-ISSN : 2803-1507 e-ISSN : 2803-1531

Kontribusi penelitian ini adalah menyediakan model implementasi presensi *mobile* yang tidak hanya memperbaiki proses operasional presensi, tetapi juga memberikan bukti empiris bahwa pemisahan tanggung jawab kode dapat menurunkan kompleksitas dan meningkatkan kemudahan pemeliharaan sistem. Dengan demikian, penelitian ini diharapkan dapat menjadi referensi bagi institusi pendidikan yang ingin mengembangkan sistem presensi digital yang lebih valid, efisien, dan berkelanjutan.

2. TINJAUAN PUSTAKA

2.1. Sistem Presensi *Mobile dan Geo-tagging Evidence*

Sistem presensi mobile merupakan sistem informasi yang memungkinkan pengguna melakukan pencatatan kehadiran melalui perangkat bergerak. Keunggulan utama pendekatan ini adalah akses yang lebih fleksibel, pengurangan ketergantungan pada perangkat fisik tertentu, dan percepatan proses rekapitulasi data presensi. Beberapa penelitian terbaru menunjukkan bahwa presensi mobile dapat dikombinasikan dengan *geolocation, geofencing, face recognition, liveness detection,* dan fitur riwayat kehadiran untuk meningkatkan akurasi serta efisiensi pencatatan kehadiran [14], [17]. Dalam konteks penelitian ini, presensi mobile tidak hanya diposisikan sebagai pengganti mesin fingerprint, tetapi juga sebagai sistem validasi data kehadiran yang lebih adaptif. *Geo-tagging evidence* adalah penyematan informasi lokasi geografis, waktu, dan bukti pendukung pada data digital. Pada sistem presensi, pendekatan ini dapat digunakan untuk memastikan bahwa *check-in* dan *check-out* dilakukan pada area yang telah ditentukan. Namun, data lokasi pada perangkat *mobile* tidak selalu dapat dipercaya secara langsung karena dapat dipengaruhi oleh manipulasi koordinat, konfigurasi perangkat, dan aplikasi *Fake GPS*. Oleh karena itu, penelitian sebelumnya mengenai presensi berbasis geolocation dan fake GPS detection menjadi dasar penting dalam merancang validasi lokasi yang lebih kuat [8], [15], [16].

2.2. *Pragmatic Clean Architecture* pada Aplikasi Mobile

Clean Architecture merupakan pendekatan perancangan perangkat lunak yang menekankan pemisahan tanggung jawab antar komponen sistem. Konsep utama arsitektur ini adalah menjaga logika inti agar tidak bergantung langsung pada detail antarmuka atau infrastruktur. Pada aplikasi *mobile*, pendekatan ini umumnya diterapkan melalui pemisahan lapisan *presentation*, *domain*, dan *data*. Penelitian sebelumnya menunjukkan bahwa *Clean Architecture* membantu pengembang mengelola kompleksitas, meningkatkan modularitas, dan mempermudah proses pengujian serta pemeliharaan aplikasi [3], [4]. Penelitian ini menggunakan *Pragmatic Clean Architecture*, yaitu adaptasi *Clean Architecture* yang lebih ringkas untuk konteks *thin client*. Aplikasi *mobile* dalam penelitian ini berperan sebagai pengumpul *data* presensi, sedangkan validasi bisnis utama seperti radius, jam kerja, dan persetujuan *data* dilakukan pada *server*. Oleh karena itu, lapisan *domain* tidak diterapkan secara berlebihan apabila hanya menghasilkan *pass-through logic*. Struktur yang digunakan mengombinasikan *Model-View-ViewModel* (MVVM) dan *Repository Pattern* agar UI tidak berkomunikasi langsung dengan sumber *data*. Pendekatan ini tetap mempertahankan *separation of concerns* dan *dependency rule*, tetapi mengurangi *boilerplate code* yang tidak diperlukan.

2.3. *Maintainability, ISO/IEC 25010, LOC, dan Cyclomatic Complexity*

Maintainability adalah kemampuan perangkat lunak untuk dianalisis, dimodifikasi, diuji, dan dikembangkan kembali ketika terjadi perubahan kebutuhan. Dalam *ISO/IEC 25010*, *maintainability* memiliki sub-karakteristik seperti modularity, reusability, *analyzability*, *modifiability*, dan *testability*. Karakteristik ini penting karena biaya pemeliharaan sering menjadi bagian terbesar dalam siklus hidup perangkat lunak. Beberapa studi menunjukkan bahwa evaluasi kualitas perangkat lunak berbasis *ISO/IEC 25010*, arsitektur, dan metrik internal dapat digunakan untuk menilai kecenderungan kualitas sistem secara lebih objektif [11]-[13], [20], [25]. *Lines of Code* dan *Cyclomatic Complexity* merupakan metrik internal yang sering digunakan untuk mengevaluasi struktur kode. *LOC* menggambarkan ukuran dan kepadatan kode pada method, sedangkan *CC* menggambarkan jumlah jalur logika independen yang harus dipahami dan diuji. Kode dengan *LOC* yang tinggi cenderung lebih sulit dianalisis, sementara kode dengan *CC* tinggi memerlukan lebih banyak skenario pengujian. Penurunan kedua metrik tersebut tidak otomatis mewakili seluruh kualitas perangkat lunak, tetapi dapat menjadi indikator awal terhadap *analyzability*, *modifiability*, dan *testability*, terutama ketika dikombinasikan dengan *static code analysis* [18], [19], [20].

2.4. Keamanan Lokasi dan *Anti-Spoofing*

Keamanan lokasi menjadi tantangan utama pada sistem presensi *mobile*. Pada aplikasi *native Android*, sistem operasi menyediakan mekanisme untuk mengetahui apakah lokasi berasal dari penyedia lokasi tiruan melalui metode location.isMock() pada *Android* S atau lebih baru dan isFromMockProvider() pada versi sebelumnya.

p-ISSN : 2803-1507 e-ISSN : 2803-1531

Penelitian mengenai presensi berbasis geolocation menunjukkan bahwa radius lokasi, foto selfie, dan deteksi *Fake GPS* dapat membantu meningkatkan keandalan data kehadiran [15], [16]. Dengan mekanisme tersebut, aplikasi dapat menolak presensi sejak awal ketika koordinat terindikasi berasal dari *mock location* provider. Pada *PWA*, mekanisme deteksi *native* tidak dapat diakses karena aplikasi berjalan di dalam *browser sandbox*. Oleh karena itu, validasi lokasi pada *PWA* perlu menggunakan pendekatan heuristik, seperti *multi-sampling*, *jitter analysis*, pemeriksaan nilai akurasi, dan sinkronisasi waktu klien-*server*. Pendekatan ini tidak sekuat deteksi *native*, tetapi tetap berguna sebagai lapisan mitigasi tambahan pada perangkat yang tidak dapat menjalankan aplikasi *Android native*. Karena browser hanya memberikan koordinat melalui geolocation API, validasi akhir tetap perlu dilakukan melalui *server-side validation*. Berdasarkan kajian literatur yang telah diuraikan, ringkasan penelitian terdahulu dan gap penelitian disajikan pada Tabel 1.

Tabel 1. Ringkasan Penelitian Terdahulu dan Gap Penelitian

No	Peneliti	Fokus	Gap terhadap penelitian ini
1	Rochimah dkk. [11]	*Framework maintainability* pada OOP	Belum diterapkan pada aplikasi presensi mobile berbasis *geo-tagging*.
2	Bukovcan dkk. [3]	*Clean Architecture* pada *client-side software*	Tidak membahas sistem presensi dan validasi lokasi.
3	Azziqra dan Nuryasin [4]	*Clean Architecture* pada aplikasi mobile Flutter	Tidak mengevaluasi presensi kampus dan tidak fokus pada *geo-tagging*.
4	Meng dkk. [8]	Survei GNSS *spoofing* dan *anti-spoofing*	Belum diimplementasikan dalam sistem presensi mobile.
5	Gupta dkk. [5]	*Code smells* pada aplikasi Android	Tidak merancang ulang arsitektur presensi.
6	Rojas dkk. [12]	Evolusi ISO/IEC 25010	Tidak melakukan implementasi aplikasi atau studi kasus presensi.
7	Suryadi dan Sulistiyani [13]	Evaluasi kualitas aplikasi menggunakan ISO/IEC 25010	Tidak mengukur maintainability berbasis LOC dan CC pada arsitektur presensi.
8	Ahmad dan Pratiwi [14]	Presensi mobile dengan *face recognition, liveness detection,* dan *geolocation*	Belum membahas *maintainability* berbasis *Clean Architecture* dan perbandingan LOC/CC.
9	Mahtum dkk. [15]	Presensi dan penggajian berbasis *mobile-web* dengan *geolocation* dan *fake GPS detection*	Tidak mengevaluasi kualitas struktur kode aplikasi menggunakan ISO/IEC 25010.
10	Pratama dan Siswanto [16]	Presensi berbasis *geolocation* dan radius lokasi	Tidak membahas arsitektur aplikasi mobile dan maintainability kode.
11	Ramdani dkk. [18]	Static code analysis untuk isu keamanan dan kualitas kode	Tidak diterapkan pada aplikasi presensi mobile berbasis geo-tagging.
12	Hussien [20]	*Software metrics*, LOC, dan *Cyclomatic Complexity* terhadap maintainability	Tidak membahas validasi presensi dan arsitektur aplikasi mobile kampus.

Sumber: Hasil kajian literatur penulis, 2026.

3. METODOLOGI PENELITIAN

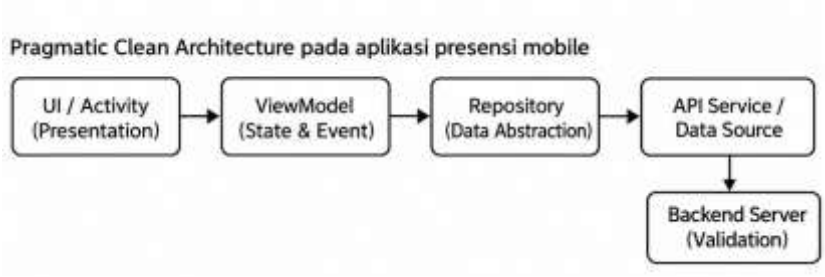
3.1. Jenis, Objek, dan Teknik Pengumpulan Data

Penelitian ini menggunakan pendekatan penelitian dan pengembangan perangkat lunak dengan studi kasus pada Universitas Teknologi Digital. Objek penelitian adalah aplikasi presensi *mobile* karyawan yang dibangun untuk menggantikan ketergantungan terhadap mesin *fingerprint* dan prototipe presensi berbasis *QR Code*. Fokus penelitian berada pada implementasi *geo-tagging evidence*, pemisahan struktur kode menggunakan *Pragmatic Clean Architecture*, dan evaluasi *maintainability* berdasarkan metrik internal perangkat lunak

3.2. Metode Pengembangan Sistem

Metode pengembangan sistem menggunakan *Agile Scrum* karena memungkinkan proses pengembangan dilakukan secara iteratif dan bertahap. *Product backlog* terdiri atas modul autentikasi, modul presensi, modul validasi lokasi, antarmuka pengguna, riwayat presensi, laporan, kalender, dan ekspor Excel. *Sprint* pertama difokuskan pada pembentukan struktur arsitektur dan autentikasi pengguna. *Sprint* kedua difokuskan pada integrasi *geo-tagging*, pengambilan foto, deteksi *mock location*, dan pengiriman data presensi ke *server*. *Sprint* ketiga difokuskan pada integrasi antarmuka, penyempurnaan PWA, laporan, serta pengujian kualitas kode. Implementasi sistem dilakukan pada dua jalur akses. Jalur pertama adalah aplikasi *native* Android yang memanfaatkan API perangkat untuk membaca koordinat, memeriksa indikasi *mock location*, mengambil foto, dan mengirimkan data presensi. Jalur kedua adalah PWA yang disediakan untuk pengguna iOS atau pengguna yang mengakses sistem melalui *browser*. PWA menggunakan fitur *geolocation* pada *browser* dan validasi heuristik untuk mengurangi risiko manipulasi lokasi. Alur Pragmatic Clean Architecture yang digunakan dalam sistem presensi mobile ditampilkan pada Gambar 1.

p-ISSN : 2803-1507 e-ISSN : 2803-1531



Gambar 1. Alur *Pragmatic Clean Architecture* pada sistem presensi mobile. Sumber: Hasil perancangan arsitektur sistem, 2026.

3.3. Alur Validasi Presensi

Alur presensi dimulai ketika pengguna berhasil melakukan *login* dan menekan tombol *check-in* atau *check-out*. Aplikasi kemudian membaca koordinat perangkat dan mengambil foto pengguna sebagai *evidence*. Pada aplikasi *native* Android, sistem memeriksa apakah koordinat berasal dari *mock location provider*. Jika terindikasi palsu, proses presensi dihentikan dan pengguna diminta menonaktifkan aplikasi *Fake GPS*. Jika lokasi tidak terindikasi palsu, data koordinat, waktu, status presensi, dan foto dikirim ke *server*. Di sisi *server*, sistem memvalidasi radius lokasi, waktu presensi, identitas pengguna, dan status presensi harian. Validasi *server* menjadi pusat keputusan agar perubahan aturan presensi dapat dilakukan tanpa memperbarui aplikasi *mobile*. Pada PWA, proses validasi awal dilakukan dengan mengumpulkan beberapa sampel koordinat, kemudian menganalisis kestabilan titik, nilai akurasi, dan metadata lokasi. Hasil validasi PWA tetap dikirim ke *server* untuk memperoleh keputusan akhir. Rincian skenario validasi presensi pada sistem usulan disajikan pada Tabel 2.

Tabel 2. Skenario Validasi Presensi

No	Skenario	Mekanisme	Keluaran
1	Pengguna berada dalam radius kampus	Koordinat, waktu, dan foto dikirim ke server	Presensi diterima jika seluruh aturan valid.
2	Pengguna berada di luar radius	Server menghitung jarak terhadap titik pusat	Presensi ditolak atau diberikan peringatan.
3	Fake GPS aktif pada Android native	Aplikasi memanggil isMock()/isFromMockProvider()	Presensi diblokir sebelum data dikirim.
4	Koordinat PWA terlalu statis	PWA melakukan multi-sampling dan jitter analysis	Presensi ditandai mencurigakan.
5	Jam perangkat tidak sesuai	Server membandingkan waktu klien dan waktu server	Presensi ditolak bila selisih melebihi toleransi.

Sumber: Hasil perancangan dan pengujian sistem, 2026.

3.4. Pengujian *Maintainability*

Pengujian *maintainability* dilakukan menggunakan pendekatan *static code analysis* terhadap dua varian aplikasi, yaitu aplikasi usulan berbasis *Pragmatic Clean Architecture* dan prototipe pembanding berbasis struktur monolitik. Pendekatan *static code analysis* dipilih karena dapat membantu mengidentifikasi karakteristik kualitas kode secara lebih objektif melalui pengukuran metrik internal perangkat lunak [18]. Pengukuran dilakukan menggunakan Android Studio Narwhal terhadap kode sumber aplikasi Android yang dikembangkan menggunakan bahasa pemrograman Java. Metrik yang digunakan adalah *Lines of Code (LOC)* dan *Cyclomatic Complexity (CC)* pada level *method*. LOC digunakan untuk melihat kepadatan kode dalam setiap *method*, sedangkan CC digunakan untuk melihat jumlah jalur logika independen yang perlu dipahami dan diuji. Kode dengan LOC yang tinggi cenderung membutuhkan usaha pemahaman yang lebih besar, sedangkan CC yang tinggi menunjukkan semakin banyaknya kemungkinan jalur eksekusi yang perlu diuji. Oleh karena itu, LOC dan CC dapat digunakan sebagai indikator awal untuk menilai kecenderungan *analyzability*, *modifiability*, dan *testability* pada perangkat lunak [20].

Cakupan kode yang dianalisis dibatasi pada kode sumber aplikasi yang berkaitan dengan fungsi inti presensi. Modul yang dibandingkan meliputi autentikasi pengguna, proses *check-in* dan *check-out*, pengambilan koordinat lokasi, validasi lokasi, pengiriman data presensi ke server, riwayat presensi, dan laporan presensi. Kode yang tidak terkait langsung dengan fungsi inti, seperti konfigurasi *build*, *library* eksternal, dependensi pihak ketiga, *resource layout*, aset gambar, dan komponen antarmuka yang tidak memuat logika presensi, tidak dijadikan dasar utama interpretasi *maintainability*. Perbandingan dilakukan pada cakupan fitur yang setara agar hasil pengukuran tidak dipengaruhi oleh perbedaan kelengkapan fungsi. Kedua varian sama-sama mencakup fungsi autentikasi, presensi, pengambilan lokasi, komunikasi server, riwayat presensi, dan laporan. Dengan demikian, perbedaan nilai LOC dan *Cyclomatic Complexity* diinterpretasikan sebagai indikasi

p-ISSN : 2803-1507 e-ISSN : 2803-1531

perbedaan struktur arsitektur dan distribusi tanggung jawab kode, bukan sebagai akibat dari pengurangan fitur utama pada salah satu varian aplikasi. Rata-rata *Lines of Code* per method dihitung menggunakan Persamaan (1). Variabel LOC menunjukkan jumlah baris kode pada *method* ke-i, sedangkan n menunjukkan jumlah method yang dianalisis. Nilai rata-rata ini digunakan sebagai indikator awal untuk melihat kecenderungan *analyzability* dan *modifiability*.

$$\overline{LOC} = \frac{\sum LOC}{n} \tag{1}$$

Persentase penurunan metrik dihitung menggunakan Persamaan (2). Variabel NMonolithic menunjukkan nilai metrik pada aplikasi monolitik, sedangkan NClean menunjukkan nilai metrik pada aplikasi berbasis *Pragmatic Clean Architecture*. Rumus ini digunakan untuk menghitung persentase penurunan rata-rata LOC dan *Cyclomatic Complexity*.

$$P = \frac{N_{Monolithic} - N_{Clean}}{N_{Monolithic}} \times 100\% \tag{2}$$

Penggunaan LOC dan *Cyclomatic Complexity* dalam penelitian ini dibatasi sebagai indikator awal maintainability. Kedua metrik tersebut tidak dimaksudkan untuk mewakili seluruh subkarakteristik maintainability dalam ISO/IEC 25010, seperti *modularity*, *reusability*, *analyzability*, *modifiability*, dan *testability* secara menyeluruh [11]–[13], [25]. Oleh karena itu, hasil pengujian *maintainability* dalam penelitian ini diinterpretasikan sebagai indikasi kuantitatif terhadap kepadatan kode, keterbacaan method, dan kompleksitas logika, bukan sebagai penilaian penuh terhadap seluruh aspek *maintainability* perangkat lunak.

4. HASIL DAN PEMBAHASAN

4.1. Implementasi Sistem *Presensi Mobile*

Hasil implementasi menunjukkan bahwa aplikasi presensi *mobile* dapat menjalankan fungsi utama yang dibutuhkan oleh karyawan dan admin SDM. Pada sisi karyawan, fitur utama meliputi *login*, beranda presensi, *check-in*, *check-out*, riwayat presensi, peta radius, dan kalender. Pada sisi admin SDM, sistem menyediakan laporan presensi dan fitur unduh rekap dalam format *Excel*. Implementasi ini memberikan alternatif yang lebih fleksibel dibandingkan sistem *fingerprint* karena setiap pengguna dapat melakukan presensi melalui perangkat masing-masing dalam area yang ditentukan. Antarmuka aplikasi *native Android* dirancang sederhana agar pengguna dapat melakukan presensi dengan cepat. Halaman beranda menampilkan informasi waktu, status presensi hari berjalan, dan tombol presensi. Halaman riwayat membantu pengguna memeriksa data kehadiran yang telah tercatat, sedangkan halaman laporan dan kalender membantu admin SDM melakukan pemantauan. Prinsip perancangan *UI/UX* dan prototyping pada penelitian sebelumnya menegaskan bahwa tampilan yang sederhana, konsisten, dan mudah dipahami dapat meningkatkan penerimaan pengguna pada aplikasi layanan digital [22], [23].

4.2. Implementasi *Pragmatic Clean Architecture*

Penerapan *Pragmatic Clean Architecture* menghasilkan pemisahan tanggung jawab antara lapisan presentasi dan lapisan data. Komponen *Activity* dan *Fragment* hanya bertugas menampilkan informasi serta menangani interaksi pengguna. *ViewModel* mengelola *state* dan *event* yang dibutuhkan oleh antarmuka. *Repository* menjadi perantara akses data dan menyembunyikan detail komunikasi dengan *API Service*. Struktur ini mencegah antarmuka berkomunikasi langsung dengan sumber data sehingga perubahan pada API tidak berdampak langsung pada kode tampilan. Dibandingkan struktur monolitik, pendekatan ini membuat alur data lebih jelas dan lebih mudah diuji. Pada prototipe monolitik, logika tampilan, pengambilan data, validasi sederhana, dan komunikasi *server* cenderung bercampur pada satu file atau satu fungsi yang panjang. Pada aplikasi usulan, setiap komponen memiliki tanggung jawab yang lebih spesifik. Hal ini memperkecil peluang munculnya *god method* dan mengurangi *boilerplate* yang tidak dibutuhkan. Struktur yang lebih modular juga mendukung perubahan kebijakan presensi karena sebagian besar aturan bisnis berada pada *server*.

4.3. Implementasi Validasi Keamanan Lokasi

Pada aplikasi *native* Android, validasi lokasi dilakukan dengan memanfaatkan fungsi *isMock()* untuk Android S atau lebih baru dan *isFromMockProvider()* untuk versi sebelumnya. Ketika sistem mendeteksi bahwa koordinat berasal dari *mock location provider*, proses presensi dihentikan sebelum data dikirim ke *server*. Mekanisme ini penting karena dapat mencegah data presensi yang sudah terindikasi manipulasi masuk ke proses validasi *server*. Namun, karena pengecekan tetap berjalan di perangkat klien, *server*-side validation tetap digunakan sebagai pengambil keputusan akhir. Pada PWA, *browser* tidak memberikan akses langsung

p-ISSN : 2803-1507 e-ISSN : 2803-1531

terhadap API *native* Android untuk membaca status mock location. Oleh karena itu, sistem menggunakan validasi heuristik berbasis beberapa sampel koordinat. Lokasi yang terlalu statis, memiliki nilai akurasi terlalu sempurna, tidak memiliki metadata pendukung, atau menunjukkan selisih waktu tidak wajar dapat ditandai sebagai mencurigakan. Hasil pengujian menunjukkan bahwa pendekatan PWA dapat menjadi lapisan mitigasi tambahan, tetapi tidak diposisikan sebagai jaminan mutlak terhadap seluruh bentuk *spoofing*. Kombinasi validasi *native*, heuristik, dan *server* memberikan strategi yang lebih seimbang untuk sistem presensi lintas platform. Perbandingan antara sistem fingerprint dan sistem mobile geo-tagging ditampilkan pada Tabel 3.

Tabel 3. Perbandingan Fitur Sistem Lama dan Sistem Usulan

Parameter	Sistem Fingerprint	Sistem Mobile Geo-tagging	Implikasi
Aksesibilitas	Terbatas pada dua mesin fisik	Menggunakan perangkat masing-masing pengguna	Mengurangi antrean fisik.
Validasi lokasi	Statis pada titik mesin	GPS, radius, server validation, dan evidence foto	Presensi lebih fleksibel tetapi tetap terkendali.
Rekapitulasi	Diolah manual dari data mesin	Laporan dan ekspor Excel tersedia	Mempercepat administrasi SDM.
Fleksibilitas aturan	Konfigurasi perangkat fisik	Aturan terpusat di server	Perubahan dapat dilakukan tanpa pembaruan aplikasi.
Pemeliharaan	Bergantung pada perangkat keras	Bergantung pada kualitas perangkat lunak	Mendorong kebutuhan maintainability kode.

Sumber: Hasil observasi dan implementasi sistem, 2026.

4.4. Hasil Pengujian Kualitas Kode

Pengujian kualitas kode dilakukan dengan membandingkan aplikasi usulan berbasis *Pragmatic Clean Architecture* dan prototipe monolitik berdasarkan metrik *Lines of Code (LOC)* dan *Cyclomatic Complexity (CC).* Hasil pengukuran disajikan pada Tabel 4. Tabel tersebut menunjukkan bahwa aplikasi berbasis *Pragmatic Clean Architecture* memiliki nilai LOC dan CC yang lebih rendah dibandingkan prototipe monolitik pada metrik yang dianalisis.

Tabel 4. Perbandingan Metrik Kualitas Perangkat Lunak

Parameter Pengukuran	*Pragmatic Clean Architecture*	*Monolithic*	Status
Rata-rata LOC per *Method*	16,85	36,26	Reduksi 53,5%
Max LOC *Method* Terpanjang	254	740	*Eliminasi god method*
Rata-rata *Cyclomatic Complexity*	1,78	2,32	*Kategori low risk*
Total *Cyclomatic Complexity*	206	248	Logika lebih sederhana
Total LOC	3.208	4.392	Struktur lebih efisien

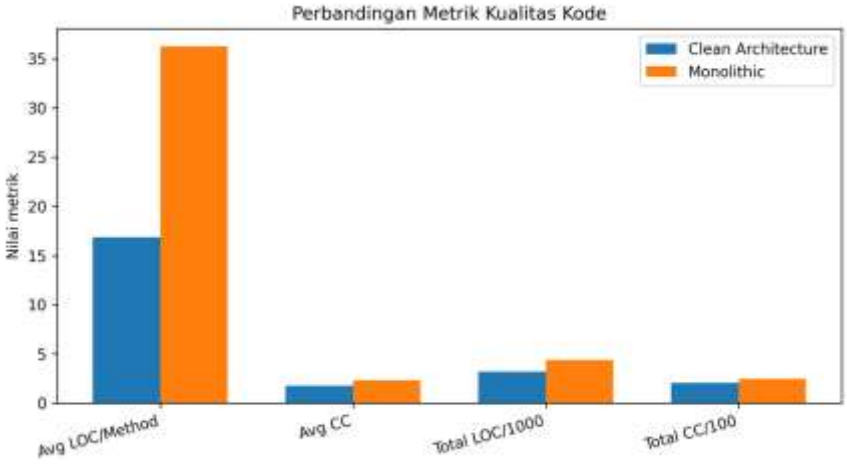
Sumber: Hasil pengolahan data *Static Code Analysis* pada penelitian, 2026.

Berdasarkan Tabel 4, rata-rata LOC per method pada aplikasi berbasis *Pragmatic Clean Architecture* adalah 16,85, sedangkan prototipe monolitik memiliki rata-rata LOC per method sebesar 36,26. Dengan menggunakan Persamaan (2), penurunan rata-rata LOC per method adalah 53,5%. Selain itu, *method* terpanjang pada aplikasi berbasis *Pragmatic Clean Architecture* memiliki 254 LOC, sedangkan pada prototipe monolitik mencapai 740 LOC. Temuan ini menunjukkan bahwa aplikasi usulan memiliki kepadatan kode yang lebih rendah pada level *method*, sehingga kode cenderung lebih mudah dibaca dan dianalisis. Pada metrik *Cyclomatic Complexity*, aplikasi berbasis *Pragmatic Clean Architecture* memiliki rata-rata CC sebesar 1,78, sedangkan prototipe monolitik memiliki rata-rata CC sebesar 2,32. Dengan demikian, terjadi penurunan rata-rata CC sebesar 23,27%. Total *Cyclomatic Complexity* juga menurun dari 248 pada prototipe monolitik menjadi 206 pada aplikasi berbasis *Pragmatic Clean Architecture*, atau sebesar 16,94%. Nilai rata-rata CC pada kedua varian masih berada pada kategori kompleksitas rendah, tetapi aplikasi usulan menunjukkan nilai yang lebih rendah dibandingkan prototipe monolitik.

Total LOC pada aplikasi berbasis *Pragmatic Clean Architecture* adalah 3.208 baris, sedangkan prototipe monolitik memiliki 4.392 baris. Penurunan total LOC sebesar 26,96% menunjukkan bahwa pemisahan tanggung jawab kode melalui *Pragmatic Clean Architecture* dapat mengurangi kepadatan kode secara keseluruhan. Perbandingan ini dilakukan pada cakupan fitur inti yang setara, yaitu autentikasi, presensi, pengambilan lokasi, komunikasi server, riwayat presensi, dan laporan, sehingga hasil pengukuran lebih merepresentasikan perbedaan struktur arsitektur daripada perbedaan kelengkapan fitur. Visualisasi perbandingan metrik kualitas kode internal ditampilkan pada Gambar 2. Gambar tersebut memperlihatkan perbedaan nilai rata-rata LOC per method, rata-rata *Cyclomatic Complexity*, total LOC yang diskalakan, dan total *Cyclomatic Complexity* yang diskalakan antara aplikasi berbasis *Pragmatic Clean Architecture* dan

p-ISSN : 2803-1507 e-ISSN : 2803-1531

prototipe monolitik. Berdasarkan Gambar 2, aplikasi berbasis *Pragmatic Clean Architecture* memiliki nilai metrik yang lebih rendah dibandingkan prototipe monolitik pada seluruh parameter yang divisualisasikan.



Gambar 2. Grafik perbandingan metrik kualitas kode internal. Sumber: Hasil pengolahan data *Static Code Analysis* pada penelitian, 2026.

Secara umum, hasil pengujian menunjukkan bahwa aplikasi berbasis *Pragmatic Clean Architecture* menghasilkan struktur kode yang lebih ringkas dan kompleksitas logika yang lebih rendah dibandingkan prototipe monolitik. Namun, hasil ini tetap diinterpretasikan secara hati-hati karena LOC dan *Cyclomatic Complexity* hanya digunakan sebagai indikator awal *maintainability*. Dengan demikian, hasil pengujian ini tidak dimaksudkan sebagai penilaian penuh terhadap seluruh subkarakteristik *maintainability* dalam ISO/IEC 25010, melainkan sebagai indikasi kuantitatif awal terhadap *analyzability*, *modifiability*, dan *testability*.

4.5. Pembahasan terhadap *Maintainability ISO/IEC 25010*

Hasil pengujian kualitas kode pada Tabel 4 dan visualisasi pada Gambar 2 menunjukkan bahwa aplikasi berbasis *Pragmatic Clean Architecture* memiliki nilai LOC dan *Cyclomatic Complexity* yang lebih rendah dibandingkan prototipe monolitik. Temuan ini dapat dikaitkan dengan karakteristik *maintainability* dalam ISO/IEC 25010, terutama pada aspek *analyzability*, *modifiability*, dan *testability*. Namun, hubungan tersebut perlu dipahami sebagai indikasi awal karena penelitian ini hanya menggunakan metrik internal berupa LOC dan *Cyclomatic Complexity*, bukan seluruh metrik *maintainability* secara komprehensif [11], [13], [25]. Dari aspek *analyzability*, penurunan rata-rata LOC per method dari 36,26 pada prototipe monolitik menjadi 16,85 pada aplikasi berbasis *Pragmatic Clean Architecture* menunjukkan bahwa method pada aplikasi usulan cenderung lebih ringkas. *Method* yang lebih pendek memudahkan pengembang dalam membaca alur kode, memahami tujuan fungsi, menelusuri lokasi kesalahan, dan melakukan analisis terhadap perubahan yang diperlukan. Selain itu, penurunan method terpanjang dari 740 LOC pada prototipe monolitik menjadi 254 LOC pada aplikasi usulan menunjukkan berkurangnya kecenderungan *method* yang terlalu panjang dan sulit dipahami.

Dari aspek *modifiability*, penerapan *Pragmatic Clean Architecture* membantu memisahkan tanggung jawab kode antara *presentation layer*, *ViewModel*, *Repository*, dan *API Service*. Pemisahan ini membuat perubahan pada satu bagian sistem tidak langsung berdampak pada seluruh kode aplikasi. Misalnya, perubahan pada *endpoint server*, format respons API, atau mekanisme pengambilan data dapat lebih banyak ditangani pada lapisan *Repository* atau *API Service* tanpa harus mengubah kode tampilan secara langsung. Dengan demikian, penurunan LOC tidak hanya menunjukkan berkurangnya jumlah baris kode, tetapi juga mengindikasikan bahwa tanggung jawab kode lebih terdistribusi dibandingkan struktur monolitik. Dari aspek testability, penurunan rata-rata *Cyclomatic Complexity* dari 2,32 pada prototipe monolitik menjadi 1,78 pada aplikasi berbasis *Pragmatic Clean Architecture* menunjukkan bahwa jumlah jalur logika independen pada method aplikasi usulan lebih rendah. Total *Cyclomatic Complexity* juga menurun dari 248 menjadi 206. Nilai CC yang lebih rendah dapat membantu proses pengujian karena jumlah percabangan logika yang perlu diverifikasi menjadi lebih sedikit. Hal ini sejalan dengan penggunaan *software metrics* sebagai indikator awal dalam menilai kompleksitas dan potensi usaha pemeliharaan perangkat lunak [20].

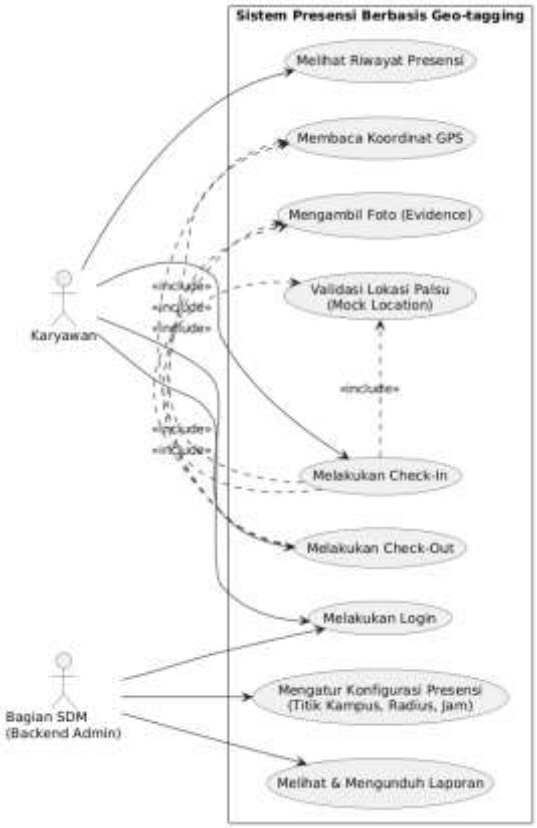
Meskipun demikian, hasil pengujian ini tidak dapat diartikan bahwa aplikasi telah memenuhi seluruh karakteristik maintainability secara penuh. ISO/IEC 25010 mencakup subkarakteristik *maintainability* yang

p-ISSN : 2803-1507 e-ISSN : 2803-1531

lebih luas, seperti *modularity*, *reusability*, *analyzability*, *modifiability*, dan *testability*. Penelitian ini belum mengukur aspek lain seperti *Maintainability Index*, *coupling*, *cohesion, code duplication*, *test coverage*, *defect density*, maupun waktu aktual yang dibutuhkan pengembang untuk melakukan perubahan kode. Oleh karena itu, LOC dan *Cyclomatic Complexity* dalam penelitian ini diposisikan sebagai indikator awal yang mendukung evaluasi *maintainability*, bukan sebagai representasi penuh dari kualitas *maintainability* perangkat lunak. Dengan batasan tersebut, hasil pengujian tetap menunjukkan kecenderungan positif bahwa *Pragmatic Clean Architecture* membantu menghasilkan struktur kode yang lebih ringkas, lebih mudah dianalisis, dan memiliki kompleksitas logika yang lebih rendah dibandingkan prototipe monolitik. Temuan ini memperkuat argumen bahwa pemisahan tanggung jawab kode dapat mendukung pemeliharaan aplikasi presensi mobile, terutama ketika sistem perlu dikembangkan lebih lanjut untuk menyesuaikan perubahan kebijakan presensi, validasi lokasi, laporan, atau integrasi dengan sistem kepegawaian institusi.

4.6. Implikasi Operasional

Secara operasional, aplikasi presensi *mobile* berbasis *geo-tagging* dapat mengurangi ketergantungan pada mesin *fingerprint*. Karyawan tidak perlu melakukan antrean pada titik fisik yang sama karena presensi dapat dilakukan melalui perangkat masing-masing selama berada dalam radius yang valid. Bagi admin SDM, fitur laporan dan ekspor *Excel* dapat mempercepat proses rekapitulasi data yang sebelumnya memerlukan pengolahan manual. Hal ini menunjukkan bahwa sistem usulan memberikan manfaat praktis bagi pengguna dan pengelola data presensi. Dari sisi keberlanjutan, struktur kode yang lebih *maintainable* membuat aplikasi lebih siap menghadapi perubahan kebutuhan. Apabila institusi perlu menambah metode validasi, mengubah tampilan, memperluas fitur laporan, atau menyesuaikan API, perubahan tersebut dapat dilakukan dengan risiko yang lebih rendah karena tanggung jawab komponen sudah dipisahkan. Namun, penelitian ini masih memiliki keterbatasan karena pengujian keamanan belum mencakup seluruh bentuk *spoofing* tingkat lanjut dan belum menerapkan *automated testing* penuh. Keterbatasan ini menjadi dasar pengembangan selanjutnya.



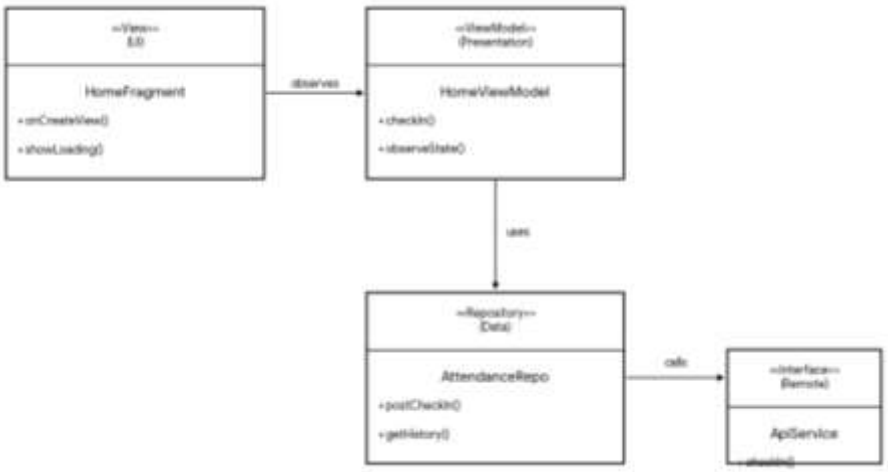
Gambar 3. *Use case diagram* sistem presensi berbasis geo-tagging. Sumber: Dokumentasi hasil implementasi sistem pada laporan penelitian, 2026.

4.7. Pemodelan Interaksi dan Struktur Sistem

Pemodelan sistem dilakukan untuk memperjelas batasan fungsional aplikasi dan hubungan antar komponen sebelum implementasi kode. Pada penelitian ini, pemodelan tidak hanya berfungsi sebagai dokumentasi,

p-ISSN : 2803-1507 e-ISSN : 2803-1531

tetapi juga sebagai alat bantu untuk memastikan bahwa struktur aplikasi sejalan dengan prinsip separation of concerns. Use case diagram digunakan untuk menggambarkan interaksi dua aktor utama, yaitu karyawan sebagai pengguna aplikasi presensi dan admin SDM sebagai pengelola data presensi pada sisi backend. Karyawan melakukan login, melihat riwayat, membaca koordinat GPS, mengambil foto evidence, melakukan *check-in*, dan melakukan *check-out*. Admin SDM mengatur konfigurasi presensi, melihat data kehadiran, dan mengunduh laporan. Berdasarkan model interaksi tersebut, sistem dibangun dengan batas tanggung jawab yang jelas. Aplikasi *mobile* tidak memegang seluruh aturan bisnis presensi karena sebagian besar validasi utama tetap berada di *server*. Keputusan ini dipilih untuk menjaga integritas data dan memudahkan perubahan aturan presensi. Jika radius presensi, jam masuk, toleransi keterlambatan, atau ketentuan cuti berubah, perubahan dapat dilakukan pada sisi *server* tanpa mengubah paket aplikasi yang sudah terpasang pada perangkat karyawan. Dalam konteks aplikasi presensi kampus, pendekatan ini lebih realistis dibandingkan menyimpan seluruh aturan pada klien karena perangkat pengguna memiliki risiko manipulasi lebih tinggi.



Gambar 4. *Class diagram* struktur *Pragmatic Clean Architecture.* Sumber: Dokumentasi hasil implementasi sistem pada laporan penelitian, 2026.

Class diagram menunjukkan bahwa HomeFragment sebagai komponen tampilan tidak mengambil data langsung dari *API Service*. Permintaan data diproses melalui Home*ViewModel* dan Attendance*Repository*, sehingga detail koneksi jaringan tersembunyi dari lapisan antarmuka. Hubungan yang loosely coupled ini membuat perubahan pada sumber data, endpoint, atau format respons tidak langsung berdampak pada tampilan. Pada aplikasi monolitik, perubahan kecil pada komunikasi *server* dapat memaksa pengembang menyentuh file tampilan yang panjang. Pada struktur usulan, dampak perubahan dapat dibatasi pada komponen *Repository* atau *Data Source* yang relevan. *Sequence* diagram memperlihatkan bahwa proses presensi berjalan secara berurutan dari pengguna menuju aplikasi, kemudian ke *repository, data source*, dan *server*. Aplikasi bertugas mengakuisisi data lokasi dan foto, sedangkan *server* bertugas memvalidasi radius, waktu, dan status presensi. *Component diagram* mempertegas hubungan tersebut dengan memisahkan *presentation component* dan *data component*. Dengan pola ini, aplikasi lebih mudah diuji karena pengembang dapat melakukan simulasi respons *repository* tanpa harus menjalankan komunikasi jaringan aktual pada setiap skenario pengujian. Interaksi aktor karyawan dan admin SDM dalam sistem presensi ditampilkan pada Gambar 3, sedangkan struktur kelas yang memperlihatkan hubungan antara HomeFragment, ViewModel, Repository, dan API Service ditampilkan pada Gambar 4.

4.8. Implementasi Antarmuka Native Android dan PWA

Implementasi antarmuka *native* Android direalisasikan berdasarkan rancangan yang telah dibuat pada tahap perancangan. Halaman login menjadi pintu masuk utama aplikasi dan berfungsi memastikan bahwa hanya karyawan terdaftar yang dapat menggunakan sistem. Setelah berhasil login, pengguna diarahkan ke halaman beranda yang menampilkan identitas pengguna, status presensi hari berjalan, waktu saat ini, dan tombol *check-in* atau *check-out*. Penyusunan elemen antarmuka dibuat sederhana agar proses presensi dapat dilakukan cepat, terutama pada waktu masuk dan pulang kerja ketika pengguna membutuhkan interaksi yang singkat. Halaman riwayat presensi disediakan untuk meningkatkan transparansi data antara pengguna dan pengelola sistem. Pengguna dapat memeriksa tanggal presensi, jam masuk, jam pulang, dan status kehadiran tanpa meminta rekap manual kepada bagian SDM. Halaman peta radius menampilkan area presensi yang

p-ISSN : 2803-1507 e-ISSN : 2803-1531

valid sehingga pengguna dapat memahami posisi perangkat terhadap titik koordinat kampus. Halaman laporan dan kalender memberikan dukungan tambahan untuk admin dan pengguna dalam memantau pola kehadiran. Fitur ekspor *Excel* pada laporan membantu admin SDM mengolah data presensi bulanan secara lebih cepat.

PWA disediakan sebagai jalur akses alternatif untuk pengguna iOS dan perangkat yang tidak menggunakan aplikasi *native* Android. Secara fungsi, PWA mempertahankan alur utama yang sama, yaitu login, beranda, riwayat, peta radius, laporan, dan kalender. Perbedaannya terletak pada cara aplikasi mengakses fitur perangkat. Aplikasi *native* Android memiliki akses lebih langsung terhadap API perangkat, sedangkan PWA dibatasi oleh *browser sandbox*. Oleh karena itu, beberapa komponen PWA dibuat lebih sederhana dan difokuskan pada aksesibilitas lintas platform. Keputusan menyediakan dua jalur akses memberikan manfaat operasional karena institusi tidak perlu membatasi presensi hanya pada satu ekosistem perangkat. Namun, perbedaan kemampuan antara *native* Android dan PWA perlu dipahami sebagai konsekuensi teknis. Validasi *native* lebih kuat untuk mendeteksi *mock location*, sementara PWA membutuhkan kombinasi heuristik dan validasi *server*. Dengan dokumentasi keterbatasan ini, sistem dapat digunakan secara lebih bijak sesuai tingkat risiko dan kebutuhan pengguna. Tampilan login aplikasi native Android ditunjukkan pada Gambar 5, sedangkan tampilan beranda PWA sebagai jalur akses lintas platform ditunjukkan pada Gambar 6.



Gambar 5. Tampilan login aplikasi native Android, Sumber: Dokumentasi hasil implementasi sistem pada laporan penelitian, 2026.



Gambar 6. Tampilan beranda PWA sebagai alternatif akses lintas platform. Sumber: Dokumentasi hasil implementasi sistem pada laporan penelitian, 2026.

4.9. Pengujian *Anti-Spoofing* dan Validasi Keamanan Lokasi

Pengujian keamanan lokasi dilakukan untuk memastikan bahwa sistem dapat merespons skenario manipulasi koordinat. Pada aplikasi *native* Android, skenario pengujian dilakukan dengan mengaktifkan aplikasi *Fake GPS* dan mengaktifkan opsi *mock location* pada pengaturan pengembang. Ketika pengguna mencoba melakukan *check-in*, aplikasi memeriksa status lokasi menggunakan API yang tersedia pada sistem operasi. Apabila lokasi terindikasi berasal dari penyedia lokasi tiruan, aplikasi menolak presensi dan menampilkan peringatan kepada pengguna. Hasil ini menunjukkan bahwa validasi di sisi *native* dapat menjadi filter awal yang efektif sebelum data dikirimkan ke *server*. Meskipun mekanisme *native* mampu membaca status *mock location*, penelitian ini tidak menempatkannya sebagai satu-satunya mekanisme keamanan. Aplikasi Android dapat mengalami risiko reverse engineering, modifikasi APK, atau bypass pada klien apabila pelaku memiliki kemampuan teknis. Oleh karena itu, validasi *server* tetap menjadi lapisan utama untuk memastikan koordinat, waktu, identitas pengguna, dan status presensi konsisten dengan aturan yang berlaku. Dengan demikian,

p-ISSN : 2803-1507 e-ISSN : 2803-1531

validasi *native* diposisikan sebagai mekanisme pencegahan awal, sedangkan *server*-side validation berfungsi sebagai pengambil keputusan akhir.



Gambar 7. Tampilan penolakan aplikasi *native Android* terhadap *mock location*. Sumber: Dokumentasi hasil implementasi sistem pada laporan penelitian, 2026.



Gambar 8. Tampilan penolakan PWA terhadap pola koordinat mencurigakan. Sumber: Dokumentasi hasil implementasi sistem pada laporan penelitian, 2026.

Pengujian pada PWA memiliki karakteristik berbeda karena browser tidak dapat memanggil location.isMock() secara langsung. Untuk mengatasi batasan tersebut, sistem mengambil beberapa sampel koordinat dalam rentang waktu tertentu. Jika koordinat terlalu stabil tanpa variasi alami, memiliki nilai akurasi terlalu sempurna, atau tidak menyertakan metadata pendukung, sistem menandai lokasi sebagai mencurigakan. Dalam pengujian, PWA dapat menampilkan peringatan ketika pola koordinat menunjukkan indikasi manipulasi. Namun, karena pendekatan ini bersifat heuristik, akurasinya tidak dapat disamakan dengan deteksi *native* pada Android. Hasil pengujian menunjukkan bahwa sistem usulan lebih kuat dibandingkan presensi berbasis koordinat sederhana yang hanya membaca satu titik lokasi. Multi-layer validation membuat pelaku tidak cukup hanya mengubah koordinat perangkat, karena sistem juga memeriksa pola pergerakan, kewajaran metadata, dan waktu *server*. Walaupun demikian, penelitian ini tetap menyatakan bahwa tidak ada mekanisme klien yang sepenuhnya kebal terhadap manipulasi. Keamanan sistem presensi harus dipandang sebagai kombinasi antara validasi teknis, kebijakan penggunaan, audit data, dan pemantauan oleh pengelola sistem.

Rincian skenario pengujian anti-spoofing ditampilkan pada Tabel 5. Pengujian dilakukan menggunakan Xiaomi Redmi Note 10S dengan sistem operasi Android 13. Aplikasi manipulasi lokasi yang digunakan adalah Fake GPS dari Play Store, sedangkan pengujian PWA dilakukan melalui Google Chrome pada perangkat yang sama. Radius validasi presensi pada sistem bersifat dinamis, tetapi pada pengujian ini radius yang digunakan adalah 200 meter. Setiap skenario diuji sebanyak lima kali untuk melihat konsistensi respons sistem terhadap kondisi lokasi normal dan kondisi ketika aplikasi Fake GPS diaktifkan. Tampilan penolakan aplikasi native Android terhadap mock location ditunjukkan pada Gambar 7, sedangkan tampilan penolakan PWA terhadap pola koordinat mencurigakan ditunjukkan pada Gambar 8.

4.10. Analisis Kelayakan Operasional dan Keberlanjutan Sistem

Analisis kelayakan operasional menunjukkan bahwa sistem *mobile geo-tagging* dapat menjawab beberapa keterbatasan sistem *fingerprint*. Dari sisi aksesibilitas, presensi tidak lagi bergantung pada dua unit mesin fisik. Setiap karyawan dapat melakukan presensi dari perangkat masing-masing selama berada pada radius yang ditentukan. Dampak langsungnya adalah potensi pengurangan antrean dan peningkatan efisiensi waktu pada jam masuk maupun pulang kerja. Dari sisi laporan, fitur rekap dan ekspor *Excel* dapat mengurangi beban pekerjaan manual bagian SDM. Dari sisi fleksibilitas aturan, sistem baru lebih adaptif karena parameter presensi dapat dikelola secara terpusat. Perubahan titik koordinat, radius, jam kerja, atau toleransi waktu dapat diterapkan pada *server*. Hal ini berbeda dengan sistem fisik yang membutuhkan konfigurasi pada

p-ISSN : 2803-1507 e-ISSN : 2803-1531

perangkat mesin presensi. Dalam jangka panjang, kemampuan melakukan perubahan terpusat menjadi penting karena kebijakan presensi institusi dapat berubah mengikuti kebutuhan organisasi, lokasi kampus, atau pola kerja pegawai.

Tabel 5. Skenario Pengujian Anti-Spoofing

No	Jalur Akses	Perangkat dan Lingkungan Uji	Skenario	Jumlah Percobaan	Hasil Pengujian
1	Native Android	Xiaomi Redmi Note 10S, Android 13	Presensi dilakukan dalam radius valid 200 meter tanpa aplikasi Fake GPS	5	Seluruh percobaan berhasil diproses sesuai validasi server
2	Native Android	Xiaomi Redmi Note 10S, Android 13, aplikasi Fake GPS dari Play Store	Presensi dilakukan ketika aplikasi Fake GPS aktif	5	Seluruh percobaan berhasil ditolak oleh aplikasi karena terdeteksi sebagai mock location
3	PWA	Google Chrome pada Android 13	Presensi dilakukan dalam radius valid 200 meter tanpa aplikasi Fake GPS	5	Seluruh percobaan berhasil diproses sesuai validasi server
4	PWA	Google Chrome pada Android 13, aplikasi Fake GPS dari Play Store	Presensi dilakukan ketika aplikasi Fake GPS aktif	5	Seluruh percobaan berhasil ditandai sebagai lokasi mencurigakan dan presensi ditolak, tetapi proses pengecekan membutuhkan waktu lebih lama dibandingkan aplikasi native Android

Sumber: Hasil pengujian sistem, 2026.

Dari sisi keberlanjutan perangkat lunak, hasil *static code analysis* menunjukkan bahwa struktur kode usulan lebih efisien daripada pendekatan monolitik. Penurunan LOC dan CC memberikan indikasi bahwa sistem lebih mudah dianalisis dan diuji. Hal ini penting karena aplikasi presensi bukan sistem yang selesai setelah satu kali dikembangkan. Sistem semacam ini akan terus membutuhkan pemeliharaan, perbaikan bug, penyesuaian laporan, integrasi dengan sistem kepegawaian, dan perubahan antarmuka. Oleh karena itu, kualitas internal kode menjadi faktor strategis, bukan sekadar isu teknis pengembang. Walaupun sistem usulan memiliki manfaat operasional, terdapat beberapa batasan yang harus diperhatikan. Pertama, penelitian belum melakukan pengujian pada jumlah pengguna besar secara bersamaan sehingga aspek *performance* dan *scalability* perlu diuji lebih lanjut. Kedua, validasi PWA masih bergantung pada heuristik sehingga tidak dapat sepenuhnya menggantikan deteksi *native*. Ketiga, pengukuran *maintainability* masih berfokus pada LOC dan CC, belum menghitung *Maintainability Index* secara komprehensif. Batasan ini tidak mengurangi kontribusi penelitian, tetapi menjadi arah penting untuk pengembangan berikutnya.

Selain aspek teknis tersebut, penelitian ini juga memiliki keterbatasan pada evaluasi pengguna. Sistem telah diimplementasikan berdasarkan kebutuhan operasional presensi karyawan dan admin SDM, tetapi penelitian ini belum melakukan pengujian *usability* secara formal kepada pengguna akhir. Oleh karena itu, kemudahan penggunaan, kenyamanan alur *check-in* dan *check-out*, kejelasan tampilan riwayat presensi, serta kesesuaian fitur laporan dengan kebutuhan admin SDM belum diukur menggunakan instrumen *usability* yang terstruktur. Keterbatasan ini perlu diperhatikan karena keberhasilan aplikasi presensi mobile tidak hanya ditentukan oleh validitas lokasi dan kualitas struktur kode, tetapi juga oleh penerimaan dan kemudahan penggunaan oleh karyawan serta pengelola data presensi.

5. KESIMPULAN DAN SARAN

5.1. Kesimpulan

Penelitian ini berhasil merancang dan mengimplementasikan aplikasi presensi mobile karyawan dengan pendekatan *Pragmatic Clean Architecture*. Struktur aplikasi memisahkan tanggung jawab antara *presentation layer*, *ViewModel*, *Repository*, dan *API Service* sehingga antarmuka tidak bergantung langsung pada sumber data. Pemisahan ini membuat alur kode lebih terstruktur dibandingkan prototipe monolitik dan mendukung pengembangan sistem presensi yang lebih mudah dipelihara. Penerapan *geo-tagging evidence* berhasil digunakan sebagai mekanisme pendukung validasi presensi melalui pengambilan koordinat, foto, waktu, dan validasi sisi server. Pada aplikasi *native Android*, deteksi *mock location* dilakukan menggunakan mekanisme yang tersedia pada sistem operasi Android, sedangkan pada PWA digunakan validasi heuristik berbasis pola koordinat, akurasi lokasi, dan validasi sebelum data dikirim ke server. Berdasarkan pengujian terbatas menggunakan Xiaomi Redmi Note 10S dengan Android 13, Google Chrome, radius validasi 200 meter, dan aplikasi Fake GPS dari Play Store, sistem berhasil menolak atau menandai lokasi mencurigakan pada 5 dari 5 percobaan pada jalur native Android dan PWA. Namun, validasi PWA membutuhkan waktu

p-ISSN : 2803-1507 e-ISSN : 2803-1531

pengecekan lebih lama dan tidak diposisikan setara dengan deteksi native Android karena masih berpotensi menghasilkan false positive maupun false negative.

Evaluasi *maintainability* berdasarkan metrik LOC dan *Cyclomatic Complexity* menunjukkan bahwa aplikasi berbasis *Pragmatic Clean Architecture* memiliki struktur kode yang lebih ringkas dan kompleksitas logika yang lebih rendah dibandingkan prototipe monolitik. Rata-rata LOC per method menurun dari 36,26 menjadi 16,85 atau sebesar 53,5%, sedangkan rata-rata *Cyclomatic Complexity* menurun dari 2,32 menjadi 1,78 atau sebesar 23,27%. Total LOC juga menurun dari 4.392 menjadi 3.208, dan total *Cyclomatic Complexity* menurun dari 248 menjadi 206. Hasil ini menunjukkan indikasi awal bahwa aplikasi usulan lebih mudah dianalisis, dimodifikasi, dan diuji. Meskipun demikian, klaim *maintainability* dalam penelitian ini dibatasi pada penggunaan LOC dan *Cyclomatic Complexity* sebagai metrik internal. Kedua metrik tersebut belum merepresentasikan seluruh subkarakteristik *maintainability* dalam ISO/IEC 25010, seperti *modularity*, *reusability*, *analyzability*, *modifiability*, dan *testability* secara menyeluruh. Oleh karena itu, hasil penelitian ini sebaiknya dipahami sebagai indikasi awal kualitas struktur kode, bukan sebagai penilaian komprehensif terhadap *maintainability* perangkat lunak.

5.2. Saran

Pengembangan berikutnya disarankan menambahkan mekanisme *offline caching* agar data presensi dapat disimpan sementara ketika jaringan tidak stabil dan disinkronisasi ketika koneksi kembali tersedia. Fitur ini dapat meningkatkan *reliability* tanpa menghilangkan validasi *server* sebagai pusat keputusan. Keamanan sistem dapat ditingkatkan melalui integrasi autentikasi biometrik, *device binding*, *token refresh* yang lebih ketat, serta hardening terhadap *reverse engineering* menggunakan *ProGuard* atau *R8*. Rekomendasi ini relevan karena penelitian terkait obfuscated *Android* malware dan evaluasi detector terhadap teknik code-obfuscation menunjukkan bahwa aplikasi *Android* memerlukan perlindungan tambahan terhadap manipulasi kode dan pembongkaran aplikasi [9], [10]. Pada sisi *PWA*, validasi heuristik perlu dikombinasikan dengan pembatasan sesi dan analisis pola historis agar deteksi manipulasi lokasi menjadi lebih kuat.

Dari sisi kualitas perangkat lunak, penelitian lanjutan disarankan menghitung *Maintainability Index* secara eksplisit dengan memasukkan *Halstead Volume*, LOC, dan CC. Selain itu, penerapan *unit testing*, *UI testing*, dan *Continuous Integration/Continuous Deployment* (CI/CD) dapat dilakukan untuk memanfaatkan struktur kode yang sudah lebih testable. Penelitian lanjutan disarankan menambahkan evaluasi usability kepada pengguna akhir, khususnya karyawan dan admin SDM. Evaluasi dapat dilakukan melalui uji coba penggunaan aplikasi, kuesioner berbasis skala *Likert*, wawancara singkat, atau metode *usability testing* seperti *System Usability Scale (SUS)*. Aspek yang dapat dinilai meliputi kemudahan login, kejelasan tombol *check-in* dan *check-out*, pemahaman pengguna terhadap validasi lokasi, kemudahan melihat riwayat presensi, kemudahan admin dalam membaca laporan, serta kemudahan ekspor data presensi. Hasil evaluasi tersebut dapat digunakan untuk menyempurnakan antarmuka, alur interaksi, dan kesesuaian sistem dengan kebutuhan operasional institusi.

DAFTAR PUSTAKA

[1] Z. Isnan and R. A. Eryadi, "Pengembangan Lawkeeper: Platform Web Dinamis untuk Persuratan Internal," JATI (Jurnal Mahasiswa Teknik Informatika), vol. 8, no. 6, pp. 11605-11611, 2024, doi: 10.36040/jati.v8i6.11550.

[2] R. Rohman Sukarya, "Perencanaan Strategis Sistem Informasi untuk Meningkatkan Pelayanan pada Masyarakat Menggunakan Metode Ward and Peppard," INFOKOM (Informatika & Komputer), vol. 13, no. 1, pp. 41-57, 2025, doi: 10.56689/infokom.v13i1.2057.

[3] M. Bukovcan, D. Blazevic, K. Nenadic, and M. Stevic, "Clean Architecture of Client-Side Software Development for Smart Furniture Control," in 2022 11th Mediterranean Conference on Embedded Computing (MECO), 2022, doi: 10.1109/MECO55406.2022.9797122.

[4] M. Z. Azziqra and I. Nuryasin, "Implementasi Clean Architecture pada Aplikasi Mobile Al-Quran Berbasis Flutter," JOISIE (Journal of Information Systems and Informatics Engineering), vol. 8, no. 2, p. 369, 2025, doi: 10.35145/joisie.v8i2.4763.

[5] A. Gupta, B. Suri, D. Sharma, S. Misra, and L. Fernandez-Sanz, "Code smells analysis for android applications and a solution for less battery consumption," Scientific Reports, vol. 14, no. 1, 2024, doi: 10.1038/s41598-024-67660-z.

p-ISSN : 2803-1507 e-ISSN : 2803-1531

[6] P. S. Yadav, R. S. Rao, A. Mishra, and M. Gupta, "Machine Learning-Based Methods for Code Smell Detection: A Survey," Applied Sciences, vol. 14, no. 14, 2024, doi: 10.3390/app14146149.

[7] R. H. Diaz Arrieta, B. L. D. Monroy, L. J. C. Heredia, and A. G. V. Cobos, "Global Trends and Empirical Metrics in the Evaluation of Code Smells and Technical Debt: A Bibliometric Study," IEEE Access, vol. 13, pp. 138143-138165, 2025, doi: 10.1109/ACCESS.2025.3594657.

[8] L. Meng, L. Yang, W. Yang, and L. Zhang, "A Survey of GNSS Spoofing and Anti-Spoofing Technology," Remote Sensing, vol. 14, no. 19, 2022, doi: 10.3390/rs14194826.

[9] W. F. Elsersy, A. Feizollah, and N. B. Anuar, "The Rise of Obfuscated Android Malware and Impacts on Detection Methods," PeerJ Computer Science, vol. 8, e907, 2022, doi: 10.7717/peerj-cs.907.

[10] U. Nawaz, M. Aleem, and J. C.-W. Lin, "On the Evaluation of Android Malware Detectors Against Code-Obfuscation Techniques," PeerJ Computer Science, vol. 8, e1002, 2022, doi: 10.7717/peerj-cs.1002.

[11] S. Rochimah, T. R. Hadiningrum, B. D. Mardiana, D. O. Siahaan, R. J. Akbar, and A. M. Shiddiqi, "A Maintainability Framework to Ensure the Software Quality in Object-Oriented Programming," IEEE Access, vol. 13, pp. 195796-195821, 2025, doi: 10.1109/ACCESS.2025.3633265.

[12] H. Rojas, R. Renteria, V. M. Duran, Y. T. Gutierrez, M. J. I. Cabrera, and A. Aminuddin, "Mapping the Evolution and Future Directions of ISO/IEC 25010: A Bibliometric and Thematic Analysis," Engineering, Technology & Applied Science Research, vol. 15, no. 5, pp. 27530-27541, 2025, doi: 10.48084/etasr.11772.

[13] D. A. Suryadi and E. Sulistiyani, "Evaluation of Information Quality Using ISO/IEC 25010:2011 (Case Research: Menu Harianku Application)," International Journal of Innovation in Enterprise System, vol. 6, no. 1, pp. 85-95, 2024, doi: 10.25124/ijies.v6i02.167.

[14] F. A. Ahmad and N. Pratiwi, "Implementation of Face Recognition, Attendance Detection, and Geolocation using TensorFlow Lite and Google ML Kit in a Mobile Attendance Application," Sistemasi: Jurnal Sistem Informasi, vol. 14, no. 1, pp. 172-186, 2025, doi: 10.32520/stmsi.v14i1.4775.

[15] R. Mahtum, Z. Fatah, and A. Homaidi, "Design of Mobile and Web-Based Geolocation Attendance and Payroll Information System for Teachers and Employees at Madrasah Aliyah As'adiyah Meneng Ketapang Banyuwangi," G-Tech: Jurnal Teknologi Terapan, vol. 9, no. 3, pp. 1306-1316, 2025, doi: 10.70609/g-tech.v9i3.7332.

[16] Y. R. Pratama and A. Siswanto, "Application of Geolocation Methods in Student Attendance System Design," Data Science Insights, vol. 2, no. 1, pp. 1-8, 2024, doi: 10.63017/jdsi.v2i1.10.

[17] R. Shinde and A. Nilose, "Design and Development of Geofencing Based Attendance System for Mobile Application," in 2022 10th International Conference on Emerging Trends in Engineering and Technology - Signal and Information Processing (ICETET-SIP), 2022, doi: 10.1109/ICETET-SIP-2254415.2022.9791781.

[18] C. M. S. Ramdani, R. N. Shofa, M. A. K. Anshary, A. I. Gufroni, A. P. Yahya, and W. M. A. F. B. W. Hamzah, "Implementation and Evaluation of Static Code Analysis to Identify Security and Code Quality Issues in Academic Information Systems," Jurnal Teknik Informatika (JUTIF), vol. 6, no. 6, pp. 5791-5804, 2025, doi: 10.52436/1.jutif.2025.6.6.5371.

[19] N. Hake and L. Heena Dip, "Adopting SOLID Principles in Android Application Development: A Case Study and Best Practices," International Journal Software Engineering and Computer Science (IJSECS), vol. 5, no. 1, pp. 406-416, 2025, doi: 10.35870/ijsecs.v5i1.3889.

[20] N. Hussien, "Practical Implementation and Analysis of Software Metrics Impact on Maintainability in Open-Source Systems," AlKadhim Journal for Computer Science, vol. 3, no. 4, pp. 87-99, 2025, doi: 10.61710/kjcs.v3i4.135.

[21] N. N. Qoyimah, N. Anggraini, and M. A. Senubekti, "Pengembangan Tahap Pertama Aplikasi SOPWatch untuk Monitoring Standar Operasional Prosedur Berbasis Website," JATI (Jurnal Mahasiswa Teknik Informatika), vol. 8, no. 5, pp. 10485-10492, 2024, doi: 10.36040/jati.v8i5.11094.

p-ISSN : 2803-1507 e-ISSN : 2803-1531

[22] M. A. Senubekti, G. L. Dajoreyta, and N. Anggraini, "Pembuatan Desain UI/UX dengan Metode Prototyping pada Aplikasi Layanan Pengadilan Negeri Bale Bandung menggunakan Figma," Jurnal Informatika Terpadu, vol. 10, no. 1, pp. 1-10, 2024, doi: 10.54914/jit.v10i1.1001.

[23] R. Widjaya, H. P. Fitrian, and N. Anggraini, "Perancangan Antarmuka Pengguna (UI/UX) untuk Aplikasi Perpustakaan Digital SMK Pasundan Rancaekek," JATI (Jurnal Mahasiswa Teknik Informatika), vol. 8, no. 6, 2024.

[24] H. P. Fitrian, L. S. Lutfiah, R. F. Ash-Shidiq, and M. M. Oktavian, "Analisis Kinerja Jaringan Komputer pada Infrastruktur Cloud Computing Menggunakan Metode Quality of Service (QoS)," JATI (Jurnal Mahasiswa Teknik Informatika), vol. 9, no. 1, pp. 1552-1558, 2025, doi: 10.36040/jati.v9i1.12708.

[25] ISO/IEC, "ISO/IEC 25010:2023 Systems and Software Engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - Product quality model," International Organization for Standardization, 2023.