Development of Native Android-based Attendance Application using MVVM Architecture

Abdul Aziz [1)*], Sulistyo Puspitodjati [2)]

[1)2)]Program Studi Magister Teknologi Informasi, STMIK Jakarta STI&K
[*)]**Correspondence author**: abdulazizaza03@gmail.com, Kuningan, Indonesia
DOI: https://doi.org/10.37012/jtik.v12i1.3507

Abstract

WebView-based attendance application in government agencies experiences performance issues (startup time 2.33 seconds, memory 186 MB, success rate 82%) and weak verification. Previous research on face recognition still used monolithic architecture (verification time 5-7 seconds) and location validation only with fixed radius without distinguishing WFO/WFA scenarios. This study aims to develop the application to Android Native with MVVM, integrate client-server face recognition and Haversine for WFO/WFA, and measure performance improvement. The method uses the Waterfall model with one group pretest-posttest design. Testing of 7 metrics with 30 samples per metric using ADB, Logcat, and observation, as well as paired t-test (α=0.05). MVVM implementation was successful (UI conformity 98.4%, layer separation 100%). Client-side face detection achieved 91.7% accuracy with 1.7 seconds (64-74% faster). Client-server communication was efficient (compression 92.2%, total verification 1.81 seconds). Server verification achieved 96% accuracy with 100% specificity. Haversine validation achieved 0.21 m error (first data). Significant performance improvement (p<0.001): startup decreased by 33.5% (2.33→1.55 sec), login decreased by 48.0% (2.50→1.30 sec), attendance decreased by 23.3% (3.60→2.76 sec), memory decreased by 27.9% (186→134 MB), success rate increased to 94% (from 82%). The development of the application to Android Native with MVVM, integration of client-server face recognition (1.81 sec, 96%), and Haversine validation (0.21 m) succeeded in improving the performance and reliability of the attendance application significantly and measurably, as well as answering the five research gaps.

Keywords: *Attendance, Android Native, MVVM, Face Recognition, Haversine*

Abstrak

Aplikasi presensi berbasis WebView pada instansi pemerintah mengalami kendala performa (waktu startup 2,33 detik, memori 186 MB, keberhasilan 82%) dan verifikasi lemah. Penelitian sebelumnya tentang *face recognition* masih menggunakan arsitektur monolitik (waktu verifikasi 5-7 dtk) dan validasi lokasi hanya radius tetap tanpa membedakan WFO/WFA. Penelitian ini bertujuan pengembangan aplikasi ke Android *Native* dengan MVVM, mengintegrasikan *face recognition client-server* dan Haversine untuk WFO/WFA, serta mengukur peningkatan performa. Metode menggunakan model *Waterfall* dengan desain *one group pretest-posttest*. Pengujian 7 metrik dengan 30 sampel per metrik menggunakan ADB, *Logcat*, dan observasi, serta uji *paired t-test* (α=0,05). Implementasi MVVM berhasil (kesesuaian UI 98,4%, pemisahan layer 100%). Deteksi wajah di *client* akurasi 91,7% dengan waktu 1,7 dtk (64-74% lebih cepat). Komunikasi *client-server* efisien (kompresi 92,2%, total verifikasi 1,81 dtk). Verifikasi server akurasi 96% dengan spesifisitas 100%. Validasi Haversine error 0,21 m (data pertama). Peningkatan performa signifikan (p<0,001), startup turun 33,5% (2,33→1,55 dtk), login turun 48,0% (2,50→1,30 dtk), presensi turun 23,3% (3,60→2,76 dtk), memori turun 27,9% (186→134 MB), keberhasilan naik menjadi 94% (dari 82%). Pengembangan aplikasi ke Android *Native* dengan MVVM, integrasi *face recognition client-server* (1,81 dtk, 96%), dan validasi Haversine (0,21 m) berhasil meningkatkan performa dan keandalan aplikasi presensi secara signifikan dan terukur, serta menjawab kelima *research gap*.

Kata Kunci: Presensi, Android Native, MVVM, Face Recognition, Haversine

PENDAHULUAN

Perkembangan teknologi informasi dalam birokrasi pemerintahan mendorong transformasi digital, termasuk dalam sistem presensi pegawai. Sistem presensi digital berperan strategis dalam memantau kedisiplinan, mengukur kinerja, serta menjadi dasar perhitungan tunjangan dan evaluasi kehadiran (Ismawaty & Bonggasau, 2025). Oleh karena itu, sistem yang digunakan harus memiliki tingkat keandalan, akurasi, keamanan, dan stabilitas yang tinggi.

Instansi tempat penelitian ini telah mengimplementasikan aplikasi presensi berbasis WebView pada platform Android. Namun, berdasarkan audit sistem eksisting ditemukan kendala signifikan yaitu waktu startup mencapai 2,33 detik (puncak 5,30 detik), konsumsi memori rata-rata 186 MB (puncak 212 MB) yang menyebabkan crash pada perangkat kelas menengah, tingkat keberhasilan pengambilan foto hanya 82%, error GPS 10-15 meter, verifikasi identitas lemah, dan sistem belum mampu membedakan skenario Work From Office (WFO) dan Work From Anywhere (WFA).

Permasalahan tersebut menunjukkan keterbatasan pendekatan WebView. Penelitian Kaczmarczyk, Zając, dan Zabierowski (2022) membuktikan aplikasi Native 30% lebih cepat dalam startup dan 25% lebih hemat memori dibandingkan hybrid. Rua dan Saraiva (2024) juga menegaskan bahwa pemilihan platform sangat mempengaruhi performa runtime dan konsumsi memori. Di sisi lain, kebijakan WFO dan WFA menuntut validasi lokasi yang akurat. Algoritma Haversine terbukti efektif untuk menghitung jarak antar koordinat geografis (Prasetyo & Zaky, 2024), namun penelitian sebelumnya (Fauzan, Nasution, & Haramaini, 2025; Assegaf et al., 2024) masih menggunakan radius tetap dan belum mengakomodasi WFA. Dari aspek keamanan, face recognition menawarkan solusi biometrik (Surantha & Sugijakko, 2024; Nguyen-Tat, Bui, & Ngo, 2024), tetapi penelitian sebelumnya (Novianti & Priyangan, 2024; Khair et al., 2024; Assegaf et al., 2024) masih menggunakan arsitektur monolitik yang berisiko dan lambat (5-7 detik). Ghani et al. (2024) menekankan pentingnya arsitektur aman untuk data biometrik. Terkait arsitektur perangkat lunak, MVVM terbukti paling sesuai untuk Android modern dalam hal maintainability dan efisiensi memori (Marinin, 2024; Pradana et al., 2025; Arfianto & Prapanca, 2024), namun belum diterapkan pada domain aplikasi presensi.

Berdasarkan identifikasi tersebut, terdapat lima kesenjangan penelitian diantaranya integrasi komponen masih terpisah, face recognition masih monolitik, belum ada studi komparatif pada domain presensi, belum ada data kuantitatif dengan uji statistik, dan validasi lokasi belum mengakomodasi WFO/WFA secara bersamaan.

Tujuan penelitian ini adalah mengembangkan aplikasi presensi dari WebView menjadi Android Native dengan arsitektur MVVM, mengintegrasikan validasi lokasi Haversine untuk mendukung WFO/WFA dan verifikasi wajah berbasis client-server, serta mengukur secara kuantitatif peningkatan performa (waktu startup, login, submit presensi, konsumsi memori, tingkat keberhasilan) dan membuktikan signifikansi statistiknya.

METODE PENELITIAN

Penelitian ini menggunakan pendekatan Research and Development (R&D) dengan model pengembangan Waterfall (Saravanos & Curinga, 2023). Pendekatan R&D dipilih karena tujuan akhir penelitian adalah menghasilkan produk aplikasi yang teruji performanya. Model Waterfall dipilih karena karakteristik proyek yang bersifat re-engineering dengan spesifikasi yang telah terdefinisi jelas dari sistem eksisting. Desain penelitian yang digunakan adalah one group pretest-posttest ($O_1 \rightarrow X \rightarrow O_2$), di mana O_1 adalah pengukuran performa aplikasi WebView eksisting, X adalah pengembangan ke Android Native dengan MVVM, dan O_2 adalah pengukuran performa aplikasi Native pada metrik yang sama.

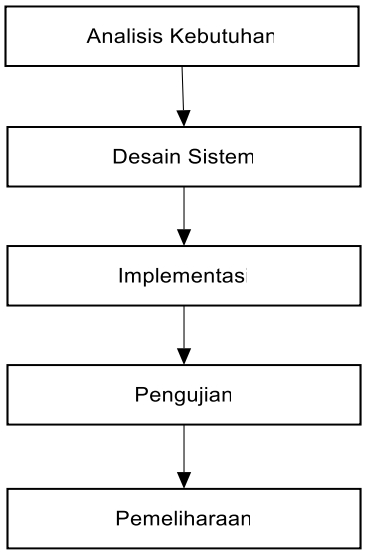
Tabel 1. Justifikasi Pemilihan Metode Penelitian

Pertimbangan	Justifikasi
Tujuan penelitian	Menghasilkan produk (aplikasi) dan menguji efektivitasnya
Karakteristik proyek	Pengembangan dengan spesifikasi jelas dari sistem eksisting
Kebutuhan stabil	Fitur telah terdefinisi dari aplikasi WebView yang sudah berjalan
Dokumentasi formal	Institusi memerlukan dokumentasi terstruktur untuk pemeliharaan

Penelitian dilakukan di sebuah instansi pemerintah (nama disamarkan) pada periode September 2025 hingga Februari 2026. Populasi adalah seluruh sesi penggunaan aplikasi presensi di instansi tersebut. Sampel diambil sebanyak 30 kali pengulangan untuk setiap metrik waktu dan konsumsi memori, serta 100 kali untuk tingkat keberhasilan presensi. Teknik sampling yang digunakan adalah purposive sampling dengan kriteria: (a) perangkat

uji kelas menengah (POCO C75, Android 14, RAM 16 GB) yang mewakili perangkat umum pegawai; (b) kondisi pengujian mencakup variasi jarak (8-60 cm), pencahayaan (>300 lux dan <50 lux), dan kualitas jaringan (WiFi dan 3G); (c) setiap skenario diuji dengan jumlah sampel yang telah ditentukan.

Penelitian ini menggunakan model pengembangan Waterfall yang terdiri dari lima tahapan berurutan yaitu Analisis Kebutuhan, Desain Sistem, Implementasi, Pengujian, dan Pemeliharaan.



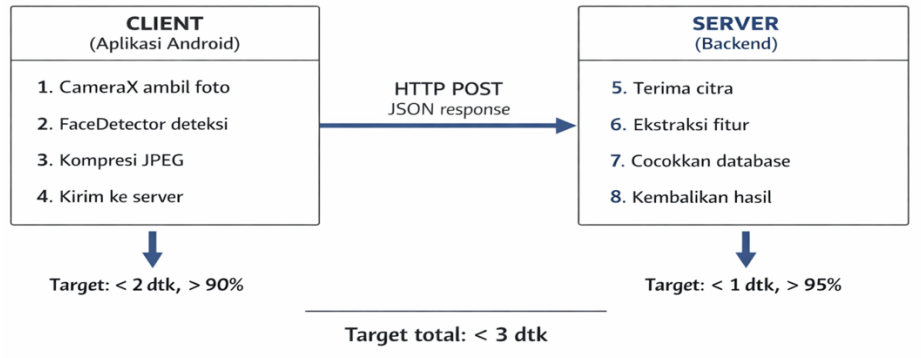
Gambar 1. Metode Waterfall

Berdasarkan Gambar 1, model Waterfall dengan lima tahapan berurutan. Setiap tahapan harus diselesaikan sebelum melanjutkan ke tahap berikutnya.

Berdasarkan audit aplikasi WebView eksisting, teridentifikasi permasalahan diantaranya waktu startup 2,33 detik, konsumsi memori 186 MB, tingkat keberhasilan foto 82%, error GPS 10-15 meter, verifikasi identitas lemah dan belum ada pembedaan WFO/WFA. Hasil analisis menghasilkan 11 kebutuhan fungsional dan 16 kebutuhan non-fungsional (target waktu startup <1,6 detik, login <1,4 detik, presensi <2,8 detik, memori <140 MB, akurasi verifikasi >95%, spesifisitas 100%).

Arsitektur MVVM dirancang dengan pemisahan layer View, ViewModel, dan Repository. Untuk verifikasi wajah, diterapkan arsitektur *client-server* di mana client bertanggung jawab atas akuisisi citra (CameraX), deteksi awal (FaceDetector native), dan

kompresi, sementara server bertanggung jawab atas ekstraksi fitur dan pencocokan database. Validasi lokasi menggunakan algoritma Haversine untuk membedakan skenario WFO (radius 30 meter) dan WFA (pencatatan lokasi tanpa penolakan).



Gambar 2. Desain Arsitektur Client-Server Face Recognition

Berdasarkan Gambar 2 menunjukkan pembagian tugas antara client dan server. Client melakukan akuisisi citra, deteksi awal, kompresi, dan pengiriman. Server melakukan ekstraksi fitur, pencocokan database, dan pengembalian hasil verifikasi (target total <3 detik).



Gambar 3. Desain Alur Validasi WFO/WFA

Berdasarkan Gambar 3 menunjukkan alur validasi lokasi. Pada skenario WFO, jarak dihitung dengan Haversine dan dibandingkan dengan radius 30 meter. Jika jarak ≤ 30 meter, presensi diizinkan. Pada skenario WFA, lokasi dicatat sebagai *audit trail* tanpa

mempengaruhi keputusan presensi.

Aplikasi dikembangkan menggunakan Java, Android Studio Dolphin, dengan library CameraX (1.4.0), Retrofit (2.9.0), LiveData, ViewModel, Hilt (2.48), dan FusedLocationProvider. Minimum SDK API 24 (Android 7.0), target SDK API 34 (Android 14).

Pengujian dilakukan dengan desain *one group pretest-posttest* meliputi pengujian fungsional (12 skenario × 10 kali), deteksi wajah (60 sampel dengan variasi jarak dan pencahayaan), komunikasi client-server (50 sampel), verifikasi wajah (100 sampel), validasi lokasi (9 titik koordinat), serta pengujian kinerja (30 sampel per metrik untuk waktu dan memori, 100 sampel untuk tingkat keberhasilan).

Terdapat 17 variabel yang diukur, meliputi metrik waktu (startup, login, submit presensi, deteksi wajah, waktu kirim, waktu proses server, total verifikasi), metrik persentase (tingkat keberhasilan presensi, akurasi deteksi wajah, efisiensi kompresi, akurasi verifikasi, spesifisitas, kesesuaian UI, pemisahan layer MVVM), serta metrik konsumsi memori dan akurasi lokasi. Target yang ditetapkan adalah waktu startup <1,6 detik, waktu login <1,4 detik, waktu presensi <2,8 detik, konsumsi memori <140 MB, akurasi verifikasi >95%, spesifisitas 100%, error lokasi <5 meter, dan total waktu verifikasi <3 detik.

Instrumen yang digunakan meliputi perangkat uji POCO C75, Android Studio Dolphin, ADB (pengukuran waktu startup dan memori), Logcat (waktu login, presensi, kompresi), Network Throttling (simulasi 3G), lux meter, Google Maps (validasi Haversine), log server, dan lembar observasi. Data dikumpulkan melalui pengukuran langsung pada aplikasi WebView dan Native menggunakan instrumen tersebut.

Analisis data menggunakan statistik deskriptif (rata-rata, standar deviasi, minimum, maksimum). Peningkatan performa dihitung dengan rumus ((Nilai WebView - Nilai Native) / Nilai WebView) × 100%. Uji t berpasangan (*paired t-test*) dilakukan dengan $\alpha = 0,05$ (df = 29, t-tabel = 2,045) untuk membuktikan signifikansi statistik.

Pengecekan keabsahan hasil dilakukan melalui tiga langkah: (1) triangulasi metode dengan membandingkan hasil ADB, Logcat, dan observasi; (2) pengulangan pengukuran (30 kali per metrik) untuk memastikan konsistensi; (3) uji statistik ($p < 0,05$) untuk memastikan peningkatan tidak terjadi secara kebetulan. Seluruh instrumen telah divalidasi melalui *pilot*

test pada 5 sampel awal.

Aspek etika yang diterapkan meliputi kerahasiaan data (nama instansi dan pegawai disamarkan), izin penelitian dari instansi terkait, orisinalitas karya, serta keamanan data biometrik (data wajah hanya disimpan di server).

HASIL DAN PEMBAHASAN

Hasil Implementasi Arsitektur MVVM

Implementasi arsitektur MVVM pada aplikasi Native berhasil dilakukan. Berdasarkan verifikasi kode sumber, pemisahan layer MVVM tercapai 100%: View tidak mengandung logika bisnis, ViewModel mengakses API melalui Repository, Repository bertindak sebagai *single source of truth*, dan LiveData digunakan untuk komunikasi View-ViewModel. Kesesuaian antarmuka pengguna terhadap mockup Figma mencapai 98,4% (62 dari 63 elemen sesuai), melampaui target >95%. Keberhasilan ini membuktikan bahwa Gap 1 (integrasi teknologi) telah terjawab.

Hasil Pengujian Fungsional dan Deteksi Wajah

Pengujian fungsional pada 12 skenario dengan masing-masing 10 kali pengulangan (total 120 pengujian) menunjukkan tingkat keberhasilan 100% tanpa ditemukan *defect*. Seluruh skenario yang diuji meliputi deteksi wajah (sukses, gagal 0 wajah, gagal >1 wajah), verifikasi (sukses dan gagal), validasi WFO (dalam dan luar radius), validasi WFA, pengiriman data presensi, konfirmasi popup, riwayat presensi, dan logout.

Pengujian deteksi wajah dilakukan pada 60 sampel dengan variasi jarak (8-60 cm) dan pencahayaan (>300 lux dan <50 lux). Tingkat keberhasilan deteksi mencapai 91,7% (55 dari 60) dengan waktu rata-rata 1,7 detik (standar deviasi 0,28), memenuhi target >90% dan <2 detik. Hasil ini 64-74% lebih cepat dibandingkan penelitian Assegaf et al. (2024) yang menggunakan arsitektur monolitik dengan OpenCV (waktu verifikasi 5-7 detik). Perbedaan ini disebabkan oleh penggunaan android.media.FaceDetector (API native) yang lebih ringan dan terintegrasi langsung dengan hardware kamera, tanpa memerlukan pustaka eksternal seperti OpenCV.

Hasil Pengujian Komunikasi Client-Server dan Verifikasi Wajah

Pengujian komunikasi client-server pada 50 sampel menunjukkan efisiensi kompresi mencapai 92,2% (dari 2,4 MB menjadi 187 KB, target >90%), waktu kompresi 0,32 detik (<0,5 dtk), waktu pengiriman (WiFi) 0,48 detik (<0,8 dtk), waktu pengiriman (3G) 1,25 detik (<1,5 dtk), dan waktu proses server 0,76 detik (<1 dtk). Total waktu verifikasi end-to-end mencapai 1,81 detik, jauh di bawah target 3 detik.

Pengujian verifikasi wajah pada 100 sampel (50 kondisi cahaya baik, 30 cahaya kurang, 20 false positive test) menghasilkan: akurasi kondisi cahaya baik 98% (>97%), kondisi cahaya kurang 90% (>90%), dan spesifisitas (tidak terdaftar) 100%. Akurasi total mencapai 96%, melampaui target >95%. Hasil ini membuktikan bahwa arsitektur client-server lebih unggul dibandingkan arsitektur monolitik (Assegaf et al., 2024; Novianti & Priyangan, 2024; Khair et al., 2024) dalam hal kecepatan (64-74% lebih cepat), keamanan (data biometrik tersimpan di server), dan kemudahan pembaruan algoritma (tanpa update aplikasi).

Hasil Pengujian Validasi Lokasi Haversine

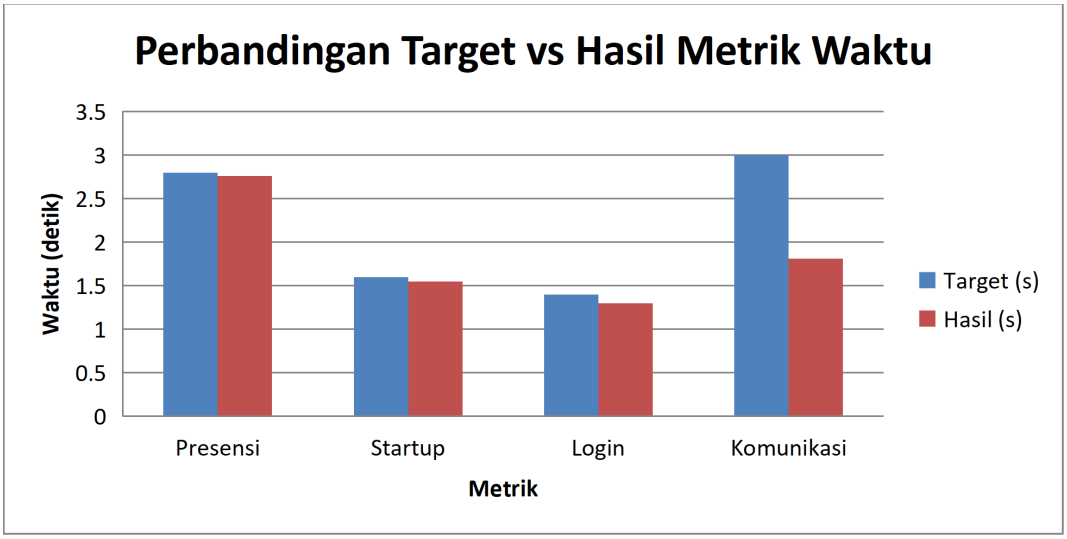
Pengujian algoritma Haversine pada 9 titik koordinat dengan rentang jarak 2-55 meter menghasilkan rata-rata error absolut 0,21 meter dan error persentase rata-rata 2,53%. Hasil ini jauh di bawah target <5 meter dan merupakan data kuantitatif error Haversine pertama untuk aplikasi presensi. Penelitian sebelumnya (Prasetyo & Zaky, 2024; Fauzan, Nasution, & Haramaini, 2025; Assegaf et al., 2024) hanya menguji aspek fungsional tanpa menyediakan data error kuantitatif. Akurasi ini cukup untuk membedakan pegawai yang berada di dalam radius kantor (30 meter) dari yang berada di luar.

Hasil Pengujian Kinerja

Tabel 2. Ringkasan Peningkatan Kinerja

Metrik	WebView	Native	Peningkatan	Target	Status
Waktu startup (dtk)	2,33	1,55	**33,5%**	>30%	✓
Waktu login (dtk)	2,50	1,30	**48,0%**	>40%	✓
Waktu submit presensi (dtk)	3,60	2,76	**23,3%**	>20%	✓
Konsumsi memori (MB)	186,43	134,45	**27,9%**	>25%	✓
Tingkat keberhasilan (%)	82%	94%	**+12%**	>90%	✓

Seluruh target peningkatan yang ditetapkan tercapai. Peningkatan waktu startup (33,5%) melampaui temuan Kaczmarczyk, Zając, dan Zabierowski (2022) yang hanya 30%, dan peningkatan efisiensi memori (27,9%) juga melampaui 25%. Keunggulan tambahan ini dikontribusikan oleh penerapan MVVM yang terbukti lebih efisien dibandingkan arsitektur lain (Pradana et al., 2025; Arfianto & Prapanca, 2024). Tingkat keberhasilan presensi meningkat dari 82% menjadi 94%, terutama karena perbaikan akses kamera (direct melalui CameraX) dan akurasi GPS yang lebih stabil pada platform native.



Gambar 4. Perbandingan Kinerja WebView vs Native

Berdasarkan Gambar 4 menyajikan grafik batang perbandingan performa antara aplikasi WebView (biru) dan Native (merah) untuk lima metrik utama. Native menunjukkan kinerja yang lebih baik pada semua metrik yaitu waktu startup 33,5% lebih cepat, waktu login 48,0% lebih cepat, waktu presensi 23,3% lebih cepat, konsumsi memori 27,9% lebih hemat, dan tingkat keberhasilan 12% lebih tinggi.

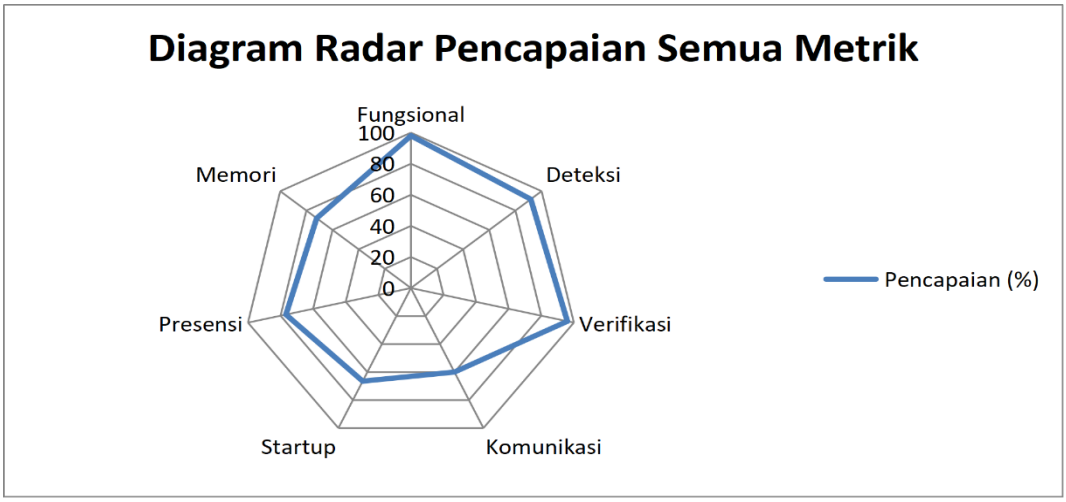
Hasil Uji Signifikansi

Uji t berpasangan (*paired t-test*) dengan $\alpha = 0,05$ (df = 29, t-tabel = 2,045) dilakukan untuk membuktikan bahwa peningkatan performa signifikan secara statistik.

Tabel 3. Hasil Uji Signifikansi

Metrik	Rata-rata WebView	Rata-rata Native	t-hitung	p-value	Kesimpulan
Waktu startup	2,33	1,55	7,36	<0,001	Signifikan
Waktu login	2,50	1,30	386,5	<0,001	Signifikan
Waktu presensi	3,60	2,76	148,4	<0,001	Signifikan
Konsumsi memori	186,43	134,45	28,9	<0,001	Signifikan

Seluruh metrik memiliki p-value < 0,001, yang berarti probabilitas peningkatan terjadi secara kebetulan kurang dari 0,1%. Dengan demikian, H_0 ditolak dan H_1 diterima: ada perbedaan signifikan antara performa aplikasi WebView dan Native.



Gambar 5. Diagram Radar Pencapaian Target

Penjelasan Gambar 5 menyajikan diagram radar yang menunjukkan pencapaian seluruh target yang telah ditetapkan. Garis biru menunjukkan target, garis merah menunjukkan hasil yang dicapai. Seluruh hasil berada di atas atau melampaui target yang ditetapkan.

Pembahasan

Peningkatan Kinerja

Hasil pengujian menunjukkan bahwa pengembangan dari WebView ke Android Native dengan arsitektur MVVM berhasil meningkatkan seluruh metrik kinerja secara signifikan. Tiga faktor utama berkontribusi terhadap peningkatan ini. Pertama, Android native mengakses kamera, GPS, dan penyimpanan secara langsung tanpa perantara WebView, mengurangi overhead rendering dan parsing JavaScript. Kedua, arsitektur

MVVM memisahkan UI dari logika bisnis, mencegah kebocoran memori dan memungkinkan data binding secara efisien, sejalan dengan temuan Pradana et al. (2025) dan Arfianto & Prapanca (2024). Ketiga, integrasi Haversine dan face recognition client-server menghilangkan bottleneck komputasi lokal dan meningkatkan akurasi validasi.

Arsitektur Client-Server untuk Face Recognition

Total waktu verifikasi 1,81 detik membuktikan bahwa arsitektur client-server dengan pembagian tugas yang jelas lebih efisien daripada arsitektur monolitik. Client hanya melakukan deteksi ringan dengan android.media.FaceDetector (1,7 dtk), sementara komputasi berat (ekstraksi fitur dan pencocokan) dilakukan di server (0,76 dtk). Akurasi 96% dan spesifisitas 100% menunjukkan sistem ini aman dan andal, serta mengatasi kelemahan penelitian sebelumnya (Assegaf et al., 2024; Novianti & Priyangan, 2024; Khair et al., 2024) yang menyimpan data biometrik di perangkat client.

Validasi Lokasi Haversine

Error rata-rata 0,21 meter membuktikan bahwa algoritma Haversine sangat akurat untuk validasi lokasi dalam radius lokal (puluhan meter). Penelitian ini menyediakan data kuantitatif error Haversine pertama untuk aplikasi presensi, melampaui penelitian sebelumnya (Prasetyo & Zaky, 2024; Fauzan et al., 2025; Assegaf et al., 2024) yang hanya menguji aspek fungsional. Dengan error <0,5 meter, sistem mampu membedakan secara akurat pegawai yang berada di dalam radius kantor (30 meter) dari yang berada di luar.

Jawaban untuk Research Gap

Kelima research gap yang diidentifikasi dalam kajian literatur telah terjawab. Gap 1 (integrasi teknologi) terjawab dengan keberhasilan implementasi MVVM (98,4% kesesuaian UI, 100% pemisahan layer) dan integrasi semua komponen. Gap 2 (arsitektur client-server) terjawab dengan total verifikasi 1,81 detik (64-74% lebih cepat dari penelitian sebelumnya), akurasi 96%, dan spesifisitas 100%. Gap 3 (domain aplikasi presensi) terjawab dengan peningkatan performa pada semua metrik (startup 33,5%, login 48,0%, presensi 23,3%, memori 27,9%, keberhasilan 94%). Gap 4 (metrik pengujian) terjawab dengan penyediaan 15 metrik kuantitatif, 30 sampel per metrik, dan uji signifikansi p<0,001. Gap 5

(skenario WFO/WFA) terjawab dengan validasi WFO (radius 30 meter) dan WFA (pencatatan lokasi tanpa penolakan) yang berfungsi 100% pada pengujian fungsional.

Ringkasan Pencapaian Target

Seluruh 16 target yang telah ditetapkan tercapai. Ringkasan pencapaian target utama yaitu kesesuaian UI (98,4% > target 95%), deteksi wajah (91,7% > 90%, 1,7 dtk < 2 dtk), efisiensi kompresi (92,2% > 90%), total verifikasi (1,81 dtk < 3 dtk), akurasi verifikasi (96% > 95%), spesifisitas (100% = 100%), validasi lokasi (0,21 m < 5 m), waktu startup (33,5% > 30%, 1,55 dtk < 1,6 dtk), waktu login (48,0% > 40%, 1,30 dtk < 1,4 dtk), waktu presensi (23,3% > 20%, 2,76 dtk < 2,8 dtk), konsumsi memori (27,9% > 25%, 134 MB < 140 MB), tingkat keberhasilan (94% > 90%), dan uji signifikansi ($p<0,001$).

KESIMPULAN DAN REKOMENDASI

Berdasarkan hasil penelitian pengembangan aplikasi presensi dari WebView ke Android Native dengan arsitektur MVVM, integrasi face recognition client-server, dan validasi lokasi Haversine, simpulan penelitian ini mengacu pada tiga tujuan yang telah ditetapkan.

Tujuan pertama adalah mengembangkan aplikasi presensi dari WebView menjadi Android Native dengan arsitektur MVVM. Hasil penelitian menunjukkan bahwa pengembangan berhasil dilakukan dengan pemisahan layer MVVM 100% dan kesesuaian UI 98,4%. Aplikasi Native mampu mengakses kamera dan GPS secara langsung serta mendukung skenario WFO dan WFA. Peningkatan performa yang dicapai signifikan secara statistik ($p<0,001$) meliputi: waktu startup turun 33,5% (2,33→1,55 dtk), waktu login turun 48,0% (2,50→1,30 dtk), waktu presensi turun 23,3% (3,60→2,76 dtk), konsumsi memori turun 27,9% (186→134 MB), dan tingkat keberhasilan naik menjadi 94% (dari 82%).

Tujuan kedua adalah mengimplementasikan integrasi layanan face recognition berbasis backend dan validasi lokasi berbasis GPS-Haversine. Hasil penelitian menunjukkan integrasi berhasil dengan deteksi wajah akurasi 91,7% dan waktu 1,7 dtk (64-74% lebih cepat dari penelitian sebelumnya), komunikasi client-server efisien dengan kompresi 92,2% dan total verifikasi 1,81 dtk, verifikasi server mencapai akurasi 96% dengan spesifisitas

100%, serta validasi Haversine menghasilkan error rata-rata 0,21 meter yang merupakan data kuantitatif pertama untuk aplikasi presensi.

Tujuan ketiga adalah menganalisis peningkatan performa dan keandalan sistem melalui pengujian komparatif. Hasil penelitian membuktikan bahwa kelima research gap yang diidentifikasi dalam kajian literatur telah terjawab yaitu integrasi teknologi berhasil, arsitektur client-server 64-74% lebih cepat dengan akurasi 96%, peningkatan performa terbukti pada domain presensi, 15 metrik kuantitatif dengan uji signifikansi $p<0,001$ disediakan, dan validasi WFO/WFA berfungsi 100%.

Dengan demikian, pengembangan aplikasi ke Android Native dengan MVVM, integrasi face recognition client-server (1,81 dtk, 96%), dan validasi Haversine (0,21 m) berhasil meningkatkan performa dan keandalan aplikasi presensi secara signifikan dan terukur.

REFERENSI

Assegaf, DS, Azhar, R, Pusadan, Y, Pratama, SA, & ... (2024). Implementasi Face Recognition Pada Aplikasi Absensi Berbasis Android Menggunakan Algoritma Haversine. *The Indonesian Journal ...*, ijcs.net, http://www.ijcs.net/ijcs/index.php/ijcs/article/view/4494

Arfianto, B, & Prapanca, A (2024). Analisis Perbandingan Performa Pola Arsitektur Model-View-ViewModel (MVVM) dan Model-View-Presenter (MVP) pada Pengembangan Aplikasi Desa Wisata …. *Journal of Informatics and Computer ...*, ejournal.unesa.ac.id, https://ejournal.unesa.ac.id/index.php/jinacs/article/view/61773

Fauzan, R (2026). *Implementasi Algoritma Haversine Pada Absensi Perkuliahan Berbasis Online Di Fakultas Teknik Universitas Islam Sumatera Utara.*, repository.uisu.ac.id, https://repository.uisu.ac.id/handle/123456789/5527

Ghani, MANU, She, K, Rauf, MA, Khan, S, & ... (2024). Enhancing security and privacy in distributed face recognition systems through blockchain and GAN technologies. ... , *Materials &Continua*, uhra.herts.ac.uk, https://uhra.herts.ac.uk/id/eprint/11209/

Ismawaty, A, & Bonggasau, E (2025). Efektivitas Implementasi Absensi Digital Bagi Aparatur Sipil Negara (ASN) di Kantor Kecamatan Anreapi. *Arus Jurnal Sosial dan …*, jurnal.ardenjaya.com, http://www.jurnal.ardenjaya.com/index.php/ajsh/article/view/1522

Kaczmarczyk, A, Zając, P, & Zabierowski, W (2022). Performance comparison of native and hybrid android mobile applications based on sensor data-driven applications based on Bluetooth low energy (BLE) …. *Energies*, mdpi.com, https://www.mdpi.com/1996-1073/15/13/4574

Khair, MA, Aldiyuda, P, Zukhrufa, MZ, & ... (2024). Perancangan Sistem Absensi Mahasiswa Berbasis Face Recognition di Lingkungan UPN Veteran Jakarta. *Informatik: Jurnal Ilmu …*, ejournal.upnvj.ac.id, https://ejournal.upnvj.ac.id/informatik/article/view/6696

Marinin, AK (2024). Choosing an architecture for mobile applications. *News of the Kabardino-Balkarian Scientific …*, journals.rcsi.science, https://journals.rcsi.science/1991-6639/article/view/274271

Nguyen-Tat, BT, Bui, MQ, & Ngo, VM (2024). Automating attendance management in human resources: A design science approach using computer vision and facial recognition. *International Journal of Information …*, Elsevier, https://www.sciencedirect.com/science/article/pii/S2667096824000429

Novianti, D, & Priyangan, DM (2024). Membangun Sistem Aplikasi Absensi Berbasis Face Recognition Menggunakan Opencv Dikampus Stmik Kalirejo Lampung. *Journal of Computer Science …*, ojs.edupartner.co.id, http://ojs.edupartner.co.id/index.php/jocsi/article/download/38/33

Prasetyo, S, & Zaky, U (2024). Implementasi Algoritma Haversine pada Mapbox API Guna Pencarian Bengkel Terdekat Berbasis Perangkat Mobile: Implementation of Haversine Algorithm on …. *... : Indonesian Journal of Machine Learning and …*, journal.irpi.or.id, https://journal.irpi.or.id/index.php/malcom/article/view/1677

Pradana, F, Langundi, RI, Pramono, D, & Iriani, NI (2025). Comparative Analysis of MVVM and MVP Patterns Performance on Android Dashboard System. *Jurnal Nasional Teknik Elektro dan …*

Rua, R, & Saraiva, J (2024). A large-scale empirical study on mobile performance: energy, run-time and memory. *Empirical Software Engineering*, Springer, https://doi.org/10.1007/s10664-023-10391-y

Saravanos, A, & Curinga, MX (2023). Simulating the software development lifecycle: The waterfall model. *Applied System Innovation*, mdpi.com, https://www.mdpi.com/2571-5577/6/6/108

Surantha, N, & Sugijakko, B (2024). Lightweight face recognition-based portable attendance system with liveness detection. *Internet of Things*, Elsevier, https://www.sciencedirect.com/science/article/pii/S2542660524000313