JURNAL ILMU KOMPUTER
KHARISMA .TECH

Jurnal KHARISMA Tech | ISSN:1907-2317 | e-ISSN: 2810-0344
https://jurnal.kharisma.ac.id/kharismatech
published by: Pusat Penelitian STMIK KHARISMA Makassar

Volume: 21 no. 01 – Maret 2026- hlm. 15-23

ANALISIS PENGARUH MODEL-VIEW-VIEWMODEL TERHADAP KINERJA APLIKASI COMFY LEARN

Oleh:

Juan Kevin Kyrieeleson Raintung[1], Syaiful Rahman[2*], Abdul Munir S.[3]
[1,2,3]Teknik Informatika, STMIK Kharisma Makassar
e-mail: [1]juankevin_22@kharisma.ac.id, [2]syaifulrahman@kharisma.ac.id,
[3]abdulmunir@kharisma.ac.id

Abstrak: *Aplikasi pembelajaran interaktif untuk anak usia dini, seperti Comfy Learn, memegang peranan penting dalam era digital. Namun, kinerja teknis yang buruk dapat menghambat efektivitas pedagogisnya. Penelitian ini bertujuan untuk menganalisis dan meningkatkan kinerja aplikasi Comfy Learn melalui penerapan pola arsitektur Model-View-ViewModel (MVVM), yang direkomendasikan oleh Google untuk pengembangan aplikasi Android modern. Penelitian ini menggunakan pendekatan kuantitatif dengan membandingkan metrik kinerja (CPU, memori, dan janky frames) serta keterpeliharaan kode (Source Lines of Code, Cyclomatic Complexity, Cognitive Complexity) sebelum dan sesudah refaktorisasi ke MVVM. Pengujian dilakukan menggunakan Android Studio Profiler, Firebase Test Lab, dan SonarQube dengan prosedur terstandarisasi. Hasil menunjukkan bahwa penerapan MVVM menurunkan janky frames sebesar 46,3% ($p < 0,05$), menurunkan kompleksitas kode sebesar 37,5% pada metrik Cognitive Complexity, dan meningkatkan stabilitas aplikasi. Penelitian ini memberi bukti empiris bahwa MVVM bukan hanya meningkatkan performa teknis aplikasi edukasi, tetapi juga menjadikannya lebih mudah dipelihara dalam jangka panjang.*

Kata kunci: *MVVM, Android, Pengujian Perangkat Lunak, Cyclomatic Complexity, Responsivitas UI*

Abstract: *Interactive learning applications for early childhood, such as Comfy Learn, play an important role in the digital age. However, poor technical performance can hinder their pedagogical effectiveness. This study aims to analyze and improve the performance of Comfy Learn by implementing the Model-View-ViewModel (MVVM) architecture pattern, recommended by Google for modern Android applications. A quantitative approach was used by comparing performance metrics (CPU usage, memory consumption, and janky frames) and code maintainability metrics (Source Lines of Code, Cyclomatic Complexity, Cognitive Complexity) before and after migrating to MVVM. Testing was conducted with Android Studio Profiler, Firebase Test Lab, and SonarQube under standardized procedures. Results show that MVVM implementation reduced janky frames by 46.3% ($p < 0.05$), lowered code complexity by 37.5%, and improved application stability. This study provides empirical evidence that MVVM not only improves technical performance but also enhances long-term maintainability of Android-based educational applications.*

Keywords: *MVVM, Android, Software Testing, Cyclomatic Complexity, UI Responsiveness*

1. PENDAHULUAN

Pendidikan anak usia dini (PAUD) merupakan salah satu fase paling fundamental dalam perkembangan manusia. Pada tahap ini, anak-anak berada dalam periode emas (golden age) yang ditandai dengan kemampuan luar biasa dalam menyerap informasi, meniru perilaku, serta membangun fondasi keterampilan kognitif, afektif, dan psikomotorik. Oleh karena itu, proses

* Corresponding author : Syaiful Rahman (syaifulrahman@kharisma.ac.id)

Diterima : *Agustus, 2025* ***Disetujui :*** *September, 2025* ***Dipublikasikan :*** *Maret, 2026*

pembelajaran pada tahap ini memerlukan strategi yang tidak hanya menekankan pada konten, tetapi juga pada metode penyampaian yang mampu memotivasi anak untuk belajar dengan cara yang menyenangkan [1].

Perkembangan teknologi digital telah membuka peluang besar bagi lahirnya berbagai media pembelajaran interaktif, salah satunya adalah aplikasi mobile. Aplikasi pembelajaran interaktif terbukti mampu meningkatkan capaian belajar anak-anak karena menggabungkan aspek kognitif dengan aktivitas bermain (learning by playing) yang sesuai dengan karakteristik anak usia dini [2], [3]. Beberapa studi juga menunjukkan bahwa penggunaan aplikasi mobile mampu memberikan kontribusi signifikan terhadap literasi, numerasi, serta keterampilan pemecahan masalah anak. Namun, efektivitas aplikasi pembelajaran tidak hanya ditentukan oleh kualitas konten, melainkan juga oleh kinerja teknis aplikasi. Faktor-faktor seperti kecepatan waktu muat (loading time), kelancaran antarmuka (UI responsiveness), dan stabilitas sistem sangat berpengaruh terhadap pengalaman belajar anak. Aplikasi yang sering mengalami lag atau crash dapat menurunkan motivasi dan membuat tujuan pedagogis tidak tercapai [4], [5]

Aplikasi Comfy Learn dikembangkan sebagai platform pembelajaran interaktif untuk anak usia dini (3–6 tahun) dengan menyediakan fitur pengenalan huruf, angka, serta permainan edukatif. Akan tetapi, versi awal aplikasi ini tidak menggunakan arsitektur perangkat lunak yang terstruktur dengan baik. Kondisi ini menimbulkan sejumlah tantangan, antara lain: (1) dari sisi pengujian, sulit dilakukan karena logika tampilan (UI) bercampur dengan logika bisnis sehingga unit testing dan integration testing menjadi rumit; (2) dari sisi pemeliharaan, perubahan kecil pada satu bagian kode dapat memengaruhi bagian lain, mengakibatkan biaya pemeliharaan meningkat dan risiko bug bertambah besar; serta (3) dari sisi skalabilitas, kode yang tidak modular menyulitkan penambahan fitur baru maupun kolaborasi antar-developer [6].

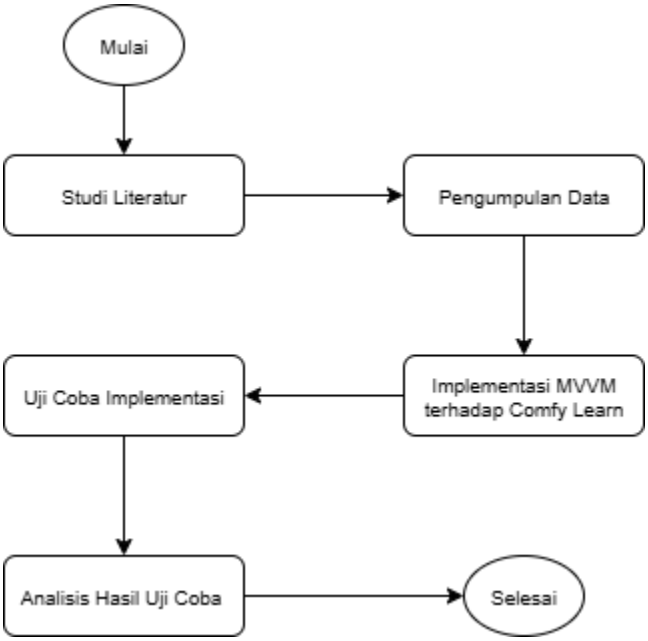
Untuk mengatasi masalah tersebut, salah satu arsitektur modern yang direkomendasikan adalah Model-View-ViewModel (MVVM). Pola arsitektur ini memisahkan antara komponen tampilan (View), logika bisnis (Model), dan logika presentasi (ViewModel), sehingga kode lebih terstruktur, mudah diuji, serta lebih mudah dipelihara. MVVM juga mendapat dukungan penuh dari Google melalui Android Jetpack, yang menyediakan komponen seperti ViewModel dan LiveData untuk mempermudah implementasinya [7], [8]

Sejumlah penelitian sebelumnya telah membuktikan manfaat MVVM. Musthafa dkk. [9] mengimplementasikan MVVM pada aplikasi hukum berbasis Android dan memperoleh hasil positif dari sisi fungsionalitas, meskipun tidak menyoroti aspek kinerja teknis. Firmansyah dkk. [10] membandingkan arsitektur MVP, MVI, dan MVVM, dan menyimpulkan bahwa MVVM unggul dalam aspek modifiability, sementara MVP lebih baik dari sisi kinerja. Indrawan dkk. [11] meneliti penerapan MVVM pada aplikasi iOS, dan menemukan peningkatan efisiensi CPU dan waktu eksekusi, meskipun penggunaan memori meningkat akibat library pihak ketiga. Diantoni dkk. [12] menerapkan MVVM pada aplikasi kolaborasi Sukacolab dan menunjukkan peningkatan pengalaman pengguna, namun penelitian ini tidak menilai kinerja teknis aplikasi secara detail. Sementara itu, Epiloksa dkk. [13] membandingkan MVC dan MVVM pada aplikasi

Android, menemukan bahwa MVVM lebih efisien dari sisi CPU dan waktu eksekusi, namun tetap menghadapi tantangan terkait konsumsi memori.

Berbeda dengan penelitian-penelitian tersebut, studi ini secara khusus berfokus pada penerapan MVVM dalam aplikasi edukasi anak-anak (Comfy Learn). Analisis dilakukan secara kuantitatif untuk mengukur dampak migrasi arsitektur terhadap kinerja aplikasi (CPU, memori, janky frames) dan keterpeliharaan kode (SLOC, Cyclomatic Complexity, Cognitive Complexity). Dengan pendekatan ini, penelitian diharapkan dapat memberikan bukti empiris bahwa penerapan MVVM bukan hanya meningkatkan kualitas teknis aplikasi Comfy Learn, tetapi juga memastikan keberlanjutan pengembangan aplikasi edukasi berbasis Android.

1.1. Tahapan Penelitian



Gambar 1: Tahapan Penelitian

Penelitian ini menggunakan pendekatan kuantitatif komparatif, dengan menganalisis versi aplikasi sebelum dan sesudah migrasi ke MVVM. Pengumpulan data dilakukan melalui pengujian pada emulator dengan spesifikasi seragam menggunakan Android Studio Profiler dan Firebase Test Lab untuk mengukur performa, serta SonarQube untuk analisis maintainability kode. Adapun tahapan-tahapan penelitian sebagai berikut:

1. Studi Literatur: Membangun landasan teoretis yang kuat dengan mengkaji berbagai sumber akademis terkait arsitektur *MVVM*, metrik kualitas perangkat lunak (*Source Lines Of Code, Cyclomatic Complexity, Cognitive Complexity*), dan *tools* pengujian kinerja (Android Studio Profiler, SonarQube).
2. Pengumpulan Data Awal (*Baseline*): Mengumpulkan data kuantitatif dari aplikasi Comfy Learn pra-migrasi untuk menetapkan *baseline* kinerja dan keterpeliharaan.
3. Implementasi *MVVM* pada Comfy Learn: Melakukan *refactoring* kode yang ada secara struktural ke arsitektur *MVVM*, dengan menggunakan Kotlin dan Android Jetpack.

4. Uji Coba Implementasi: Melakukan pengujian komprehensif pada aplikasi pasca-migrasi dengan mereplikasi secara identik proses pengumpulan data *baseline*.
5. Analisis Hasil Uji Coba: Menganalisis data secara komparatif untuk menjawab rumusan masalah penelitian.

1.2. Metode pengumpulan data

Objek penelitian adalah aplikasi Comfy Learn, dengan analisis dilakukan pada dua versi: Pra-Migrasi (Baseline) dan Pasca-Migrasi (MVVM). Teknik pengumpulan data yang digunakan adalah eksperimen dan analisis dokumentasi. Instrumen pengumpulan data meliputi Android Studio Profiler untuk metrik kinerja (UI Jank, CPU, Memori) dan SonarQube untuk metrik keterpeliharaan (SLOC, Cyclomatic Complexity, Cognitive Complexity).

Prosedur eksperimen dilakukan dengan menjalankan skenario pengguna standar sebanyak sepuluh kali (n=10) untuk kedua versi aplikasi, dan hasil rata-ratanya digunakan dalam analisis guna memastikan reliabilitas statistik. Pengujian dilakukan menggunakan Android Virtual Device (AVD) dengan spesifikasi standar berikut:

Device Name:	Pixel 9 Pro XL
API Level:	36.0 “Baklava”, Android 16.0
Resolution, Density:	1344x2992, 480 dpi
CPU cores:	4
RAM:	2GB
VM heap size:	256MB

- Prosedur Pengumpulan Data Kinerja: Menggunakan teknik eksperimen, data kinerja diperoleh dengan merekam system trace melalui Android Studio Profiler selama skenario berjalan. Data metrik rata-rata diekstrak menggunakan metode systematic sampling dari hasil profiling.
- Prosedur Pengumpulan Data Keterpeliharaan: Menggunakan teknik analisis dokumentasi, yaitu melalui static code analysis menggunakan SonarQube yang dijalankan satu kali pada keseluruhan basis kode masing-masing versi.

1.3. Metode Analisis Data

Data yang terkumpul dianalisis secara kuantitatif. Metrik disajikan dalam tabel perbandingan pra-migrasi dan pasca-migrasi, dengan perubahan persentase dihitung untuk menyoroti besaran dampak. Untuk memvalidasi temuan secara statistik, uji-t sampel independen dua-arah (*two-tailed independent samples t-test*) dilakukan pada set data kinerja. Nilai $p < 0,05$ dianggap signifikan secara statistik.

2. HASIL DAN PEMBAHASAN

Pada bagian ini akan dijelaskan metode penelitian yang digunakan, meliputi tahapan penelitian, teknik pengumpulan data, serta metode analisis data. Pemaparan ini bertujuan untuk memberikan kejelasan mengenai langkah-langkah sistematis yang ditempuh peneliti dalam menjawab rumusan masalah.

2.1. Analisis *Baseline* (Arsitektur Pra-Migrasi *MVVM*)

Untuk menetapkan dasar kuantitatif, versi awal aplikasi Comfy Learn diuji dengan fokus pada dua area utama: Kinerja Aplikasi dan Keterpeliharaan Kode.

1. Metrik Kinerja Awal

Metrik kinerja runtime diekstraksi menggunakan Android Studio Profiler selama eksekusi skenario pengguna standar yang direplikasi sebanyak sepuluh kali (n=10).

Tabel 1. Data Metrik Kinerja *baseline MVVM*

Test Name	Total Frame	Total Janky Frame	% Janky Frame	CPU Usage	Memory Usage
T01	980	109	11.12	4.88	278.8
T02	1164	86	7.39	3.91	262.9
T03	819	112	13.68	5.97	260.9
T04	790	79	10.00	3.51	271.6
T05	1010	83	8.22	1.75	263.8
T06	830	312	37.59	6.82	257.8
T07	736	277	37.64	6.00	259.0
T08	829	306	36.91	9.11	254.7
T09	745	298	40.00	2.82	183.6
T10	923	297	32.18	8.39	265.4
Total	**8826**	**1959**	**23.47**	**5.32**	**255.85**

Data baseline menunjukkan bahwa sebelum migrasi ke MVVM, aplikasi menunjukkan penggunaan CPU rata-rata 5.32%, memori 255.85 MB, dan janky frame mencapai 23.47%. Nilai janky frame ini melebihi ambang batas kenyamanan pengguna (<10%), menandakan bahwa UI rendering tidak responsif.

2. Metrik Keterpeliharaan Awal

Keterpeliharaan diukur menggunakan analisis kode statis. Tiga file utama yang menonjol dalam kategori risiko maintainability tertinggi pada versi pra-migrasi adalah *MathGameActivity.kt*, *ScatteredPileLayout.kt*, dan *SpellingFragment.kt*. File *MathGameActivity.kt* memiliki jumlah baris kode yang paling besar (339 SLOC) dan mencatat *Cyclomatic Complexity* sebesar 60 serta *Cognitive Complexity* sebesar 55. Nilai-nilai ini menunjukkan bahwa file tersebut memuat struktur logika yang sangat kompleks dan memerlukan beban kognitif yang tinggi untuk dipahami. Sementara itu, *ScatteredPileLayout.kt* menunjukkan *Cyclomatic Complexity* sebesar 28 dan *Cognitive Complexity* sebesar 27, dengan nilai *technical debt* tertinggi di antara semua file, yaitu 20. Hal ini mengindikasikan bahwa meskipun secara fungsional berperan sebagai komponen tampilan, file ini mengandung banyak jalur logika dan struktur kode yang rumit. Adapun *SpellingFragment.kt* memiliki 202 SLOC, dengan *Cyclomatic Complexity* sebesar 29 dan *Cognitive Complexity* sebesar 25. Walaupun tidak terdapat *code smell* atau *technical debt* eksplisit, kombinasi ukuran file dan kompleksitasnya menempatkan file ini pada kategori berisiko tinggi dalam hal keterpeliharaan. Data ini secara keseluruhan menggambarkan bahwa sebelum migrasi ke arsitektur MVVM, struktur kode pada beberapa komponen inti aplikasi *Comfy Learn* masih terpusat dan kompleks.

2.2. Analisis Komparatif Pasca-Migrasi *MVVM*

Bagian ini menyajikan hasil dari aplikasi yang telah direfaktor, membandingkannya secara langsung dengan baseline yang telah ditetapkan.

Tabel 2. Komparasi data pengujian pra dan pasca migrasi *MVVM*

Nama	% Janky Frame		CPU Usage		Memory Usage	
	Pre	Pasca	Pre	Pasca	Pre	Pasca
T01	11,12	9,84	4,88	3,07	278,8	235,3
T02	7,39	5,37	3,91	3,27	262,9	263,8
T03	13,68	6,41	5,97	1,77	260,9	252,8
T04	10,00	6,47	3,51	2,53	271,6	275,4
T05	8,22	7,70	1,75	7,94	263,8	269,4
T06	37,59	17,01	6,82	2,86	257,8	257,8
T07	37,64	17,26	6,00	5,66	259,0	259
T08	36,91	15,86	9,11	3,06	254,7	264,1
T09	40,00	11,80	2,82	6,35	183,6	262,6
T10	32,18	28,27	8,39	6,92	265,4	252,1
Total	23,47	12,60	5,32	4,34	255,85	259,2
Comparison	46,3%		18,3%		-1,3%	

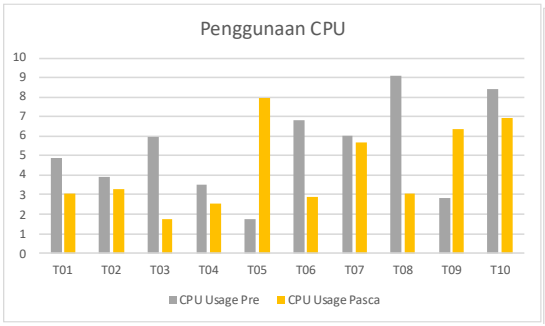
Setelah dilakukan refactoring arsitektur menggunakan pendekatan Model-View-ViewModel (MVVM), terjadi pemisahan logika bisnis dari kelas *Activity* dan *Fragment* ke dalam *ViewModel* yang bersifat lifecycle-aware. Mekanisme pengelolaan data dalam *ViewModel* diimplementasikan dengan memanfaatkan *LiveData* dan *StateFlow*, yang memungkinkan sinkronisasi data secara reaktif terhadap perubahan state aplikasi tanpa menambah beban siklus hidup komponen UI. Implementasi ini berdampak langsung terhadap peningkatan performa dan efisiensi struktur kode aplikasi.

Secara kuantitatif, hasil pengujian menunjukkan bahwa nilai *janky frame* mengalami penurunan sebesar 46,3%, dari semula 23,47% menjadi 12,59%, menandakan peningkatan signifikan dalam kelancaran rendering antarmuka pengguna. Penggunaan CPU juga menunjukkan penurunan sebesar 18,3%, yaitu dari rata-rata 5,32% menjadi 3,97%. Namun demikian, terjadi sedikit peningkatan pada penggunaan memori sebesar 1,3%, dari 255,85 MB menjadi 259,23 MB, yang diduga sebagai konsekuensi dari inisialisasi tambahan objek *ViewModel* dan cache internal dari LiveData/StateFlow. Meski peningkatan ini relatif kecil, hal tersebut masih berada dalam ambang batas efisiensi dan tidak berdampak negatif terhadap performa keseluruhan aplikasi.

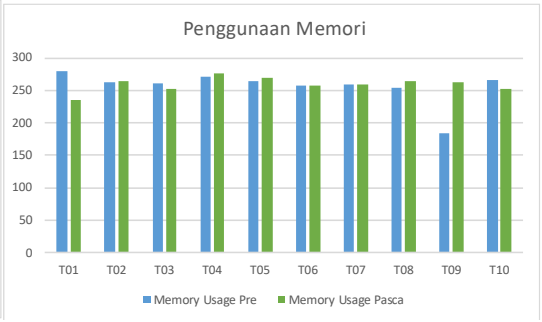
Dari sisi maintainability, penerapan MVVM juga memberikan dampak positif yang signifikan. Analisis terhadap struktur kode menunjukkan penurunan total *Source Lines of Code (SLOC)* sebesar 2,51%, mencerminkan penyederhanaan logika dalam komponen UI. Lebih lanjut, *Cyclomatic Complexity* secara keseluruhan menurun sebesar 10,82%, sedangkan *Cognitive Complexity* mengalami penurunan lebih signifikan sebesar 37,5%. Penurunan metrik-metrik ini menunjukkan bahwa struktur logika aplikasi menjadi lebih linear, mudah dipahami, dan lebih mudah untuk dilakukan pengujian serta pemeliharaan jangka panjang. Hasil ini konsisten dengan temuan dalam studi-studi sebelumnya yang menyebutkan bahwa MVVM berkontribusi terhadap peningkatan modularitas dan keterbacaan kode.

2.3. Pembahasan

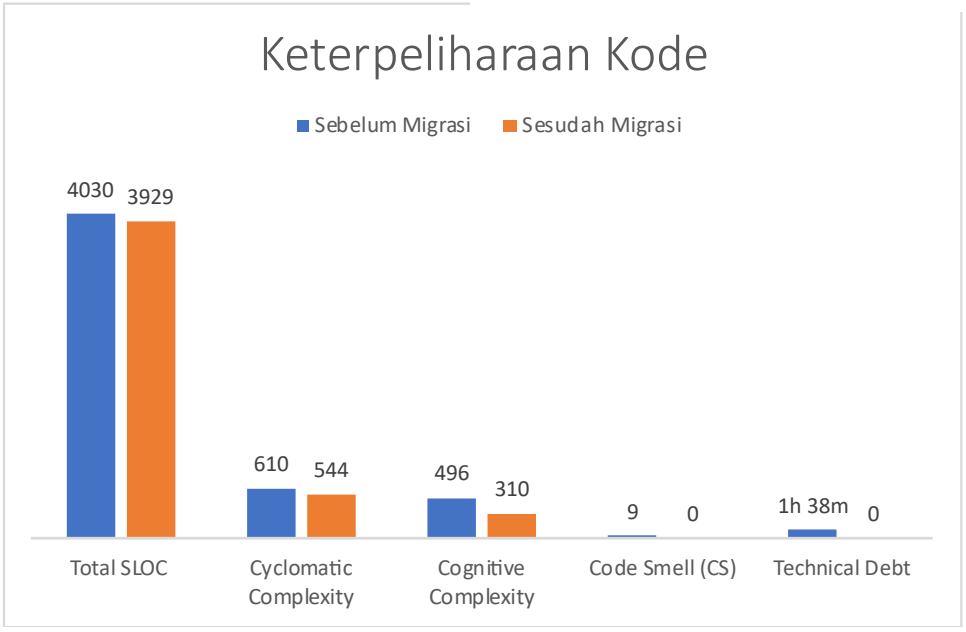
Hasil pengujian menunjukkan bahwa migrasi ke arsitektur MVVM memberikan dampak yang signifikan dan positif terhadap kinerja antarmuka pengguna dan keterpeliharaan kode aplikasi Comfy Learn. Salah satu metrik kunci yang menunjukkan peningkatan signifikan adalah persentase janky frames, yang mengalami penurunan rata-rata sebesar 46,3%, dari 23,47% menjadi 12,59%. Penurunan ini menandakan bahwa rendering UI menjadi jauh lebih halus dan responsif setelah dilakukan refactoring. Hasil uji-t pada metrik ini menunjukkan nilai signifikansi p = 0,009, yang berarti perbedaan antara sebelum dan sesudah migrasi bersifat statistik signifikan (α < 0,05).



Gambar 3. Perbandingan Penggunaan CPU



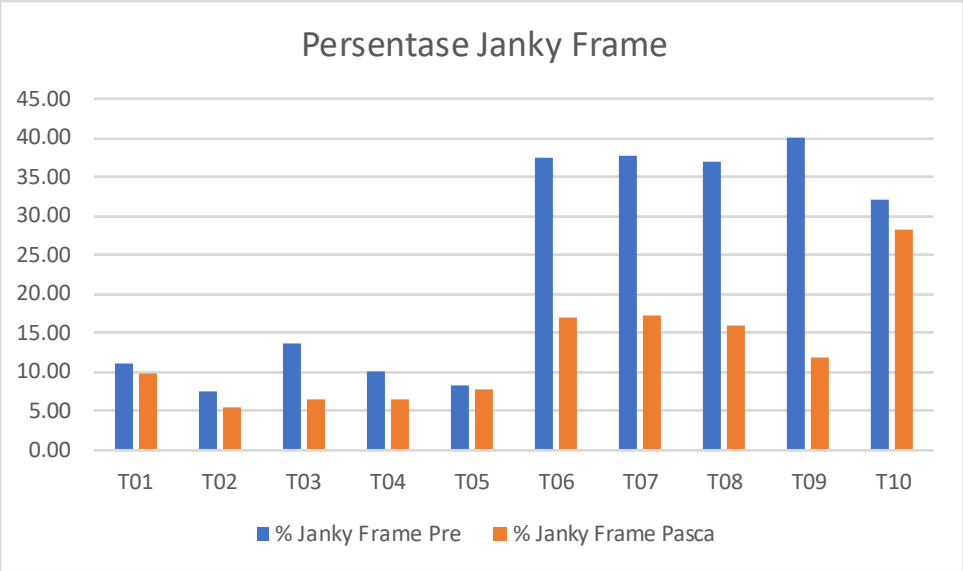
Gambar 4. Perbandingan Penggunaan Memori



Gambar 2. Perbandingan Keterpeliharaan Kode

Selain peningkatan pada aspek performa UI, manfaat arsitektur MVVM terhadap keterpeliharaan kode (maintainability) juga terlihat signifikan. Total nilai Cognitive Complexity dari seluruh proyek menurun sebesar 37,5%, menunjukkan berkurangnya beban pemahaman logika program oleh developer. Hal ini dicapai melalui implementasi prinsip pemisahan tanggung jawab, di mana logika yang kompleks dipindahkan dari komponen UI (Activity atau Fragment) ke dalam ViewModel yang bersifat modular dan independen terhadap siklus hidup

UI. Sebagai contoh konkret, Cyclomatic Complexity pada kelas FillInActivity yang sebelumnya sangat tinggi (116), menurun drastis menjadi 26 setelah dilakukan pemisahan logika ke dalam FillInViewModel.



Gambar 5. Perbandingan Janky Frame

Artinya, jumlah jalur logika bercabang yang perlu diuji dan dipahami developer menjadi jauh lebih sedikit di kelas UI, sehingga risiko terjadinya bug atau ketidakkonsistenan logika dapat ditekan secara signifikan. Secara keseluruhan, arsitektur MVVM bukan hanya meningkatkan modularitas dan keterpisahan logika aplikasi, tetapi juga mendukung praktik pengembangan perangkat lunak yang berkelanjutan. Kombinasi penurunan kompleksitas logika dan peningkatan performa UI menunjukkan bahwa arsitektur ini merupakan solusi yang layak untuk aplikasi edukasi berbasis Android.

3. KESIMPULAN

Penelitian ini melakukan analisis kuantitatif terhadap dampak migrasi aplikasi ComfyLearn ke pola arsitektur Model-View-ViewModel. Hasil penelitian menunjukkan bahwa migrasi tersebut memberikan dampak positif yang signifikan baik pada kinerja maupun keterpeliharaan aplikasi. Temuan utama adalah arsitektur MVVM berhasil menurunkan UI jank sebesar 46,3%, yang merupakan indikator kuat dari peningkatan pengalaman pengguna. Selain itu, migrasi ini menghasilkan peningkatan kualitas kode yang dramatis, terbukti dari pengurangan Cognitive Complexity total sebesar 37,5%. Ini menegaskan bahwa MVVM berhasil memfasilitasi pemisahan tanggung jawab, yang mengarah pada basis kode yang lebih mudah diuji dan dipelihara.

DAFTAR PUSTAKA

[1] C. Herodotou, “Young children and tablets: A systematic review of effects on learning and development,” 1 Februari 2018, *Blackwell Publishing Ltd*. doi: 10.1111/jcal.12220.

[2] J. Kim, J. Gilbert, Q. Yu, dan C. Gale, “Measures Matter: A Meta-Analysis of the Effects of Educational Apps on Preschool to Grade 3 Children’s Literacy and Math Skills,” *AERA Open*, vol. 7, Apr 2021, doi: 10.1177/23328584211004183.

[3] F. Niklas, E. Birtwistle, A. Mues, dan A. Wirth, “Learning apps at home prepare children for school,” *Child Dev*, vol. 96, no. 2, hlm. 577–590, Mar 2025, doi: 10.1111/cdev.14184.

[4] Arth Patel, “Demystifying App Performance Optimization: From Cold Starts to Seamless Transitions,” *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, vol. 11, no. 1, hlm. 3287–3294, Feb 2025, doi: 10.32628/cseit251112366.

[5] Vivek Chanddru, “Optimizing mobile app performance to enhance user satisfaction and engagement,” *World Journal of Advanced Engineering Technology and Sciences*, vol. 15, no. 1, hlm. 828–838, Jan 2025, doi: 10.30574/wjaets.2025.15.1.0303.

[6] Daria Yerofieieva, Vitalii Ruban, dan Anton Rumiantsev, “Large-scale Android application modernization: Case study,” https://www.griddynamics.com/blog/large-scale-android-application-modernization. Diakses: 20 Juli 2025. [Daring]. Tersedia pada: https://www.griddynamics.com/blog/large-scale-android-application-modernization

[7] C. Chekhaba, H. Rebatchi, G. Elboussaidi, N. Moha, dan S. Kpodjedo, “Coach: Classification-based architectural patterns detection in Android apps,” dalam *Proceedings of the ACM Symposium on Applied Computing*, Association for Computing Machinery, Mar 2021, hlm. 1429–1438. doi: 10.1145/3412841.3442018.

[8] A. Vijaywargi dan U. K. Boddapati, “Architectural Patterns in Android Development: Comparing MVP, MVVM, and MVI,” *Int J Res Appl Sci Eng Technol*, vol. 12, no. 4, hlm. 4611–4616, Apr 2024, doi: 10.22214/ijraset.2024.60762.

[9] A. Musthafa, D. Muriyatmoko, dan A. H. Priandika, “Implementasi MVVM Dan Framework Jetpack Compose Pada Aplikasi Hukum Berbasis Android,” 2024.

[10] Firmansyah Firdaus Anhar, Made Hanindia Prami Swari, dan Firza Prima Aditiawan, “Analisis Perbandingan Implementasi Clean Architecture Menggunakan MVP, MVI, Dan MVVM Pada Pengembangan Aplikasi Android Native,” *Jupiter: Publikasi Ilmu Keteknikan Industri, Teknik Elektro dan Informatika*, vol. 2, no. 2, hlm. 181–191, Jan 2024, doi: 10.61132/jupiter.v2i2.155.

[11] D. Indrawan, D. S. Kusumo, dan S. Y. Puspitasari, “ANALYSIS OF THE IMPLEMENTATION OF MVVM ARCHITECTURE PATTERN ON PERFORMANCE OF IOS MOBILE-BASED APPLICATIONS,” *JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)*, vol. 8, no. 1, hlm. 59–65, Feb 2023, doi: 10.29100/jipi.v8i1.3293.

[12] C. Diantoni, O. Komarudin, dan A. Rizal, “ARSITEKTUR MVVM DAN FRAMEWORK JETPACK COMPOSE PADA PENGEMBANGAN APLIKASI ANDROID,” *JATI (Jurnal Mahasiswa Teknik Informatika)*, vol. 8, no. 3, hlm. 3216–3224, Mei 2024, doi: 10.36040/jati.v8i3.9638.

[13] H. A. Epiloksa, D. S. Kusumo, dan M. Adrian, “Effect Of MVVM Architecture Pattern on Android Based Application Performance,” *JURNAL MEDIA INFORMATIKA BUDIDARMA*, vol. 6, no. 4, hlm. 1949, Okt 2022, doi: 10.30865/mib.v6i4.4545.