e-ISSN 2549-7472
Edumatic: Jurnal Pendidikan Informatika
Vol. 8 No. 2, Desember, 2024, **Hal. 545-554**
DOI: 10.29408/edumatic.v8i2.27709

Optimalisasi Aplikasi Financial Tracker berbasis Mobile dengan Penerapan Design Pattern MVVM untuk Mengelola Keuangan

Muhammad Naufal Hady Anshari Jaelani [1,*] , Yuli Asriningtias [1]

[1] Program Studi Informatika, Universitas Teknologi Yogyakarta, Indonesia
[*] Correspondence: m.naufalhadyaj@gmail.com

Received: 7 Oktober 2024 | Revised: 11 Oktober 2024 | Accepted: 30 Oktober 2024 | Published: 19 Desember 2024

Abstrak
Penggunaan pola desain yang tepat sangat penting dalam pengembangan aplikasi untuk menjaga konsistensi kode dan mempermudah pembagian tugas. Penelitian ini bertujuan mengoptimalkan pengembangan aplikasi *financial tracker* berbasis *mobile* dengan menerapkan pola desain *Model-View-ViewModel* (MVVM). Pola ini dipilih karena memisahkan logika bisnis (*Model*), tampilan (*View*), dan logika presentasi (*ViewModel*), yang membuat kode lebih terstruktur, mudah diuji, dan lebih mudah dipelihara. Penelitian ini termasuk jenis *Research and Development* (R&D), menggunakan model *waterfall*, melalui tahapan analisis, desain, implementasi, pengujian, dan pemeliharaan. Data dikumpulkan melalui studi literatur dan observasi, kemudian dianalisis untuk menguji efektivitas penerapan MVVM. Hasil temuan kami menunjukkan bahwa aplikasi *financial tracker* berbasis *mobile* dengan penerapan *design pattern* MVVM berhasil dikembangkan untuk membantu mengelola keuangan. Hasil pengujian aplikasi ini menunjukkan adanya peningkatan performa yang signifikan, dengan efisiensi kinerja CPU sebesar 74,8% pada *RenderThread* dan 27,8% pada *MainThread*. Penelitian ini berkontribusi dengan memperlihatkan bagaimana penerapan MVVM meningkatkan responsivitas, memudahkan sinkronisasi data *real-time*, serta membuat aplikasi lebih fleksibel dan efisien. Temuan ini mengisi kekosongan dalam studi sebelumnya yang kurang mengeksplorasi aspek teknis arsitektur aplikasi keuangan.

Kata kunci: *design pattern; financial tracker; mobile; mvvm*

Abstract
The use of an appropriate design pattern is crucial in application development to maintain code consistency and simplify task division. This research aims to optimize the development of a mobile-based financial tracker application by implementing the Model-View-ViewModel (MVVM) design pattern. MVVM was chosen because it separates business logic (Model), the interface (View), and presentation logic (ViewModel), making the code more structured, easier to test, and more maintainable. This study falls under the Research and Development (R&D) category, employing the waterfall model through the stages of analysis, design, implementation, testing, and maintenance. Data were collected through literature review and observation, and analyzed to evaluate the effectiveness of MVVM implementation. Our findings show that the mobile-based financial tracker application developed with MVVM design pattern successfully aids in financial management. Application testing results indicated significant performance improvements, with CPU efficiency at 74.8% on the RenderThread and 27.8% on the MainThread. This study contributes by demonstrating how MVVM improves responsiveness, simplifies real-time data synchronization, and enhances application flexibility and efficiency. These findings fill the gap in previous studies that have underexplored the technical aspects of financial application architecture.

Keywords: *design pattern; financial tracker; mobile; mvvm*

PENDAHULUAN

Aplikasi yang dirancang tanpa mempertimbangkan *design pattern* sering kali mengalami masalah performa. *Design pattern* yang tidak efisien dapat menyebabkan waktu respons yang lambat dan pengalaman pengguna yang buruk (Affandi et al., 2020). Dalam aplikasi keuangan yang mengelola *data real-time*, seperti transaksi dan saldo, pemilihan *design pattern* yang tepat sangat penting untuk menjaga kinerja optimal, terutama saat beban kerja meningkat (Trivaika, 2022). Selain itu, *design pattern* dapat membantu mengurangi kompleksitas kode (Anam & Anwar, 2020) dan juga mempermudah pemeliharaan serta pengembangan lanjutan aplikasi (Maulana et al., 2019).

Penelitian mengenai implementasi MERN *stack* yang dilakukan oleh Gunawan (2023) menunjukkan bahwa pola desain dapat meningkatkan skalabilitas dan kinerja aplikasi seiring bertambahnya pengguna. Aplikasi yang tidak menerapkan *design pattern* yang jelas sering kali mengalami penurunan performa yang signifikan. Tanpa struktur yang baik, aplikasi menjadi tidak responsif, dan waktu proses meningkat drastis, terutama ketika beban data bertambah (Affandi et al., 2020). Selain itu, kode yang tidak terstruktur cenderung sulit dibaca dan dipahami oleh pengembang lain, menyulitkan kolaborasi dan mempersulit pembaruan fitur serta pemeliharaan di masa depan (Fajri, 2022). Kondisi ini akan semakin parah jika terjadi perubahan dalam proses bisnis, karena kode yang kompleks akan memperlambat pengembangan dan pengujian.

Salah satu solusi untuk meningkatkan performa aplikasi dan menjaga keterbacaan kode karena memberikan struktur yang jelas dalam pengelolaan informasi (Rismayani et al., 2022) adalah dengan menerapkan *Model-View-ViewModel* (MVVM). Alasan utama mengapa aplikasi financial tracker harus menggunakan MVVM adalah karena aplikasi keuangan sering kali membutuhkan manajemen data *real-time*, seperti transaksi, saldo, dan laporan. Dalam konteks ini, MVVM memisahkan tanggung jawab logika bisnis (*Model*), tampilan (*View*), dan data yang menghubungkan keduanya (*ViewModel*).

Pada *design pattern* MVVM, *Model* bertanggung jawab untuk mengelola data dan logika bisnis, serta berinteraksi dengan sumber data seperti *database* atau API eksternal (Indrawan et al., 2023; Alfathar et al., 2024). *View* bertugas menampilkan data secara intuitif kepada pengguna, memastikan antarmuka mudah dipahami (Jastradaf & Asriningtias, 2023). *ViewModel* bertindak sebagai jembatan antara *Model* dan *View*, menangani logika presentasi dan memastikan data dari *Model* dapat ditampilkan dengan benar di *View* (Wiyana et al., 2021; Wang et al., 2024).

MVVM mendukung *data binding* dua arah, yang memastikan bahwa perubahan di *Model* secara otomatis tercermin di *View*, tanpa memerlukan intervensi manual (Vijaywargi & Boddapati, 2024; Epiloksa et al., 2022). Dengan memisahkan logika bisnis dari tampilan, pengujian dan pengembangan aplikasi menjadi lebih efisien, optimal, responsif, dan mudah dipelihara (Kusumawati et al., 2023; Lee et al., 2022). Sebagai perbandingan dengan pola desain seperti *Model-View-Controller* (MVC) atau *Model-View-Presenter* (MVP), MVVM unggul dalam memisahkan logika bisnis dari tampilan. MVC dan MVP sering kali membuat pengontrol atau presenter menjadi terlalu rumit, sedangkan MVVM memastikan *ViewModel* hanya fokus pada logika bisnis, sementara *View* menangani UI. Hal ini memungkinkan perubahan dalam logika bisnis tanpa mengganggu tampilan, yang sangat penting untuk fleksibilitas dan skalabilitas aplikasi keuangan (Riyadhi et al., 2023).

Penelitian sebelumnya yang dilakukan oleh Ramadhani (2024) menunjukkan bahwa penerapan *design pattern* secara konsisten dapat meningkatkan performa dan keterbacaan kode. Affandi et al. (2020) mengungkapkan bahwa aplikasi yang tidak menggunakan *design pattern* yang baik cenderung mengalami waktu respons yang lebih lama. Wiyana (2021) menunjukkan bahwa penerapan MVVM dalam sistem presensi daring mampu meningkatkan akurasi data serta mempermudah pengelolaan fitur aplikasi. Penelitian lain oleh Anam &

Anwar (2020) menegaskan bahwa *design pattern* membantu mengurangi kompleksitas kode dan meningkatkan responsivitas aplikasi secara keseluruhan. Meskipun banyak penelitian tentang teknologi keuangan, kajian teknis tentang penerapan MVVM dalam aplikasi keuangan masih terbatas (Belgacem et al., 2024; Ediagbonya & Tioluwani, 2023).

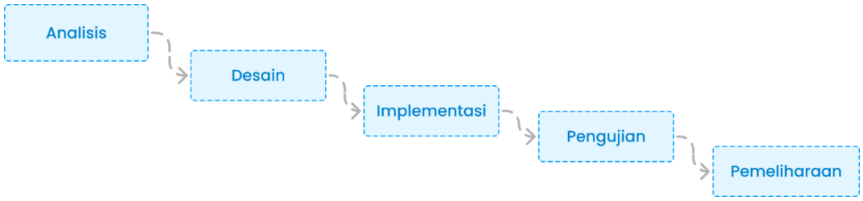
Sebagian besar penelitian sebelumnya seperti yang dilakukan oleh Belgacem (2024) beserta Ediagbonya & Tioluwani (2023) berfokus pada aspek inklusi keuangan dan inovasi layanan *fintech*. Namun, aspek teknis seperti penerapan MVVM dalam pengembangan aplikasi keuangan belum banyak dibahas. Seperti yang disampaikan pada paragraf pertama, aplikasi yang tidak menggunakan *design pattern* yang tepat cenderung mengalami masalah performa, seperti waktu respons yang lambat dan kesulitan dalam pemeliharaan kode (Affandi et al., 2020). Mengabaikan penerapan pola desain seperti MVVM dapat memperburuk masalah ini, terutama dalam aplikasi yang mengelola data *real-time*, di mana efisiensi dan responsivitas sangat penting (Tetiana et al., 2023).

Penelitian ini bertujuan untuk mengoptimalkan pengembangan aplikasi financial tracker berbasis *mobile* dengan menerapkan pola desain MVVM. Implementasi MVVM difokuskan pada peningkatan performa aplikasi, penyederhanaan kompleksitas kode, serta memfasilitasi sinkronisasi data secara *real-time* pada antarmuka pengguna. Diharapkan, penerapan MVVM dapat menghasilkan aplikasi yang lebih responsif, efisien dalam penggunaan sumber daya, optimal performanya serta mudah dipelihara. Selain itu, penerapan MVVM akan mempermudah pengembangan di masa mendatang karena arsitektur yang tertata dengan baik, memungkinkan konsistensi dalam pengembangan, meskipun dilakukan oleh *developer* yang berbeda.

METODE

Penelitian ini merupakan jenis penelitian *Research and Development* (R&D) yang bertujuan untuk menciptakan, menguji, serta mengevaluasi aplikasi pencatatan keuangan pribadi (*financial tracker*) yang dapat digunakan dalam manajemen keuangan pribadi atau bidang terkait lainnya. *Model* yang digunakan adala model *waterfall*. Pemilihan model *waterfall* dalam pengembangan aplikasi pencatatan keuangan cocok dengan *design pattern* MVVM karena keduanya membantu pembagian tugas yang jelas, bahkan untuk pengembang tunggal.

Waterfall menawarkan alur kerja terstruktur, memungkinkan fokus pada satu tahap dalam satu waktu, sementara MVVM memisahkan logika bisnis, tampilan, dan data. *Waterfall* dipilih karena persyaratan aplikasi financial tracker umumnya stabil dan jelas sejak awal, tidak seperti *agile* yang cocok untuk proyek dengan perubahan cepat atau *spiral* yang lebih sesuai untuk proyek kompleks. Dengan pendekatan linier, *waterfall* memastikan setiap tahap selesai sebelum melanjutkan, meminimalkan kesalahan dan membuat pengembangan lebih efisien serta terkontrol.



Gambar 1. Model *Waterfall*

Pada gambar 1 adalah model *waterfall* yang digunakan pada penelitian ini dan memiliki lima tahapan utama: analisis, desain, implementasi, pengujian, dan pemeliharaan. Pada tahap analisis, dilakukan pengumpulan data melalui studi literatur dan observasi untuk memahami

kebutuhan pengguna. Data ini digunakan untuk menetapkan fitur seperti pencatatan transaksi dan pembuatan laporan keuangan.

Pada tahap desain, *draw.io* digunakan untuk membuat diagram UML, seperti *use case*, *activity*, dan *sequence* diagram, serta merancang skema basis data, mengatur struktur tabel, dan relasi untuk pencatatan keuangan. *Figma* digunakan untuk merancang *wireframe* UI agar antarmuka pengguna mudah digunakan. *Design pattern* MVVM diterapkan dengan membagi aplikasi menjadi *Model* (logika bisnis), *View* (tampilan), dan *ViewModel* (pengelolaan data).

Pada tahap implementasi, aplikasi dikembangkan dengan *Kotlin* di *Android Studio* menggunakan arsitektur MVVM, yang membagi kode menjadi *Model*, *View*, dan *ViewModel*. *Kotlin* dipilih karena mendukung *null safety* untuk mencegah kesalahan *null pointer*, dan *coroutines* yang memudahkan pengelolaan tugas asinkron, seperti pengambilan data dari API atau basis data. Dalam konteks penerapan MVVM, *Kotlin* membantu membuat kode lebih aman dan efisien. *Model* mengelola logika bisnis dan data, seperti pemrosesan transaksi. *View* mengatur UI melalui file XML, *Activity*, atau *Fragment*. *ViewModel* menghubungkan *Model* dan *View*, menggunakan *LiveData* agar UI otomatis diperbarui saat data berubah.

Selanjutnya, tahap pengujian menggunakan *black-box testing* untuk memastikan semua fitur berjalan sesuai spesifikasi. Pengujian *create, read, update, delete* (CRUD) memastikan aplikasi mencatat transaksi, menghasilkan laporan akurat, dan setiap fitur berfungsi baik. Jika ditemukan *bug* maka akan dicatat dan akan diperbaiki pada fase pemeliharaan. Selain itu, uji performa juga dilakukan menggunakan *Android Profiler* untuk memantau penggunaan sumber daya seperti CPU, memastikan aplikasi efisien tanpa lag dan responsif. Evaluasi performa melibatkan perbandingan sebelum dan sesudah penerapan *design pattern* MVVM. Peningkatan performa dihitung menggunakan persamaan 1.

$$\text{Peningkatan Performa (\%)} = \left(\frac{\text{Tanpa MVVM} - \text{Dengan MVVM}}{\text{Tanpa MVVM}}\right) \times 100 \quad (1)$$

Hasil dari perhitungan tersebut akan menunjukan persentase peningkatan performa setelah diterapkan *design pattern* MVVM. Pada tahap pemeliharaan, *bug* yang ditemukan dicatat di sistem manajemen proyek seperti *git* atau *trello* dan diberi prioritas berdasarkan keparahannya. Pengembang melakukan *debugging* untuk mencari penyebab dan memperbaiki *bug*. Setelah perbaikan, bagian yang terpengaruh diuji ulang agar tidak muncul masalah baru. Semua perubahan kode terdokumentasi di *git* untuk memudahkan pelacakan, dan *rollback* bisa dilakukan jika dibutuhkan.

HASIL DAN PEMBAHASAN

Hasil

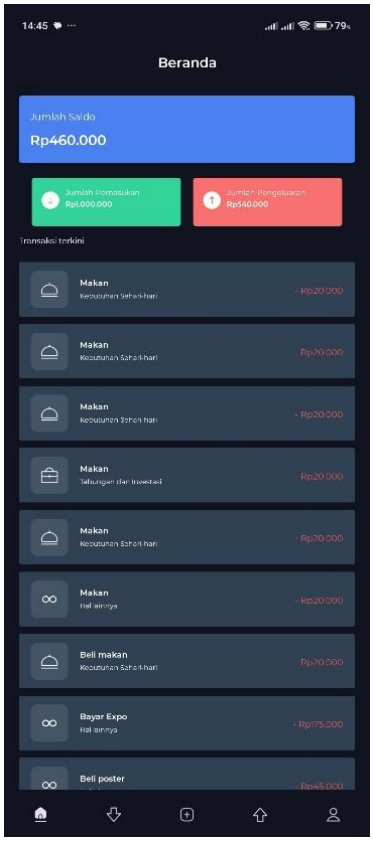
Pada tahap analisis, kami menemukan bahwa kebutuhan aplikasi dibagi menjadi fungsional dan non-fungsional. Kebutuhan fungsional mencakup fitur menambah transaksi, menampilkan data transaksi, memperbarui data transaksi dan menghapus data transaksi, yang digunakan untuk menguji optimalisasi *design pattern*. Kebutuhan non-fungsional meliputi perangkat seperti laptop dengan *Windows* 10 dan RAM minimal 8 GB serta *smartphone* dengan penyimpanan 64 GB dan RAM 4 GB. Perangkat ini berperan penting dalam memastikan aplikasi berjalan lancar selama pengembangan dan pengujian, serta memudahkan penggunaan karena sesuai dengan spesifikasi umum di pasaran.

Setelah menentukan spesifikasi perangkat, peneliti juga menemukan bahwa diperlukan data *dummy* untuk mensimulasikan pencatatan keuangan sehari-hari dalam pengujian aplikasi. Data *dummy* ini berisi data catatan transaksi keuangan sehari hari yang digunakan untuk menguji fitur-fitur aplikasi. Data tersebut diperoleh dari skenario pencatatan keuangan harian yang didasarkan pada masukan dari pengguna yang berpartisipasi dalam penelitian ini.

Pada tahap desain, *design pattern* MVVM dirancang untuk memisahkan logika aplikasi dari antarmuka pengguna secara jelas. Dalam aplikasi financial tracker ini, pengguna memasukkan data transaksi melalui antarmuka (*View*). Data yang dimasukkan kemudian dikirim ke *ViewModel*, yang bertanggung jawab mengelola logika bisnis. Misalnya, jika pengguna memasukkan jumlah transaksi, *ViewModel* akan memprosesnya, memvalidasi input tersebut, lalu berkomunikasi dengan *repository* untuk menyimpan atau mengambil data. *Repository* berfungsi sebagai perantara antara *ViewModel* dan *Data Access Object* (DAO), yang bertugas menyimpan atau mengambil data dari basis data.

Gambar 2. Arsitektur model

Model pada gambar 2 merupakan arsitektur model sistem yang berjalan. Saat pengguna menambahkan transaksi, data tersebut diterima oleh *View*, dikirim ke *ViewModel* untuk diproses, kemudian disimpan melalui DAO ke dalam basis data. Setelah itu, *ViewModel* memperbarui *View* untuk menampilkan informasi terbaru secara *real-time*, seperti saldo yang diperbarui. Pemisahan ini tidak hanya memudahkan pemeliharaan tetapi juga memastikan antarmuka tetap responsif dan data sinkron dengan baik.

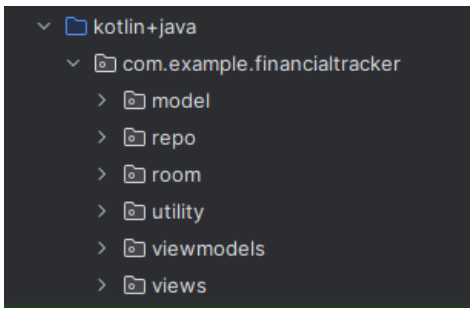


Gambar 3. Tampilan beranda

Gambar 3 adalah tampilan dari laman beranda aplikasi financial tracker berbasis mobile. Dalam arsitektur MVVM, setiap elemen di antarmuka ini berhubungan langsung dengan *ViewModel*, yang bertugas memproses logika bisnis dan berkomunikasi dengan *Model* (basis data). Pengguna dapat melihat saldo dan transaksi terbaru secara *real-time*. Yang dimana data ini berasal dari fitur tambah transaksi yang memungkinkan pengguna memasukkan data baru, kemudian langsung diproses oleh *ViewModel* dan disimpan ke basis data. Urgensi desain ini adalah memastikan antarmuka tetap intuitif dan responsif, sehingga pengguna dapat mengelola keuangan dengan efisien, sementara arsitektur MVVM menjaga pemisahan logika, tampilan, dan data untuk memudahkan pengembangan dan pemeliharaan aplikasi.

Pada tahapan implementasi, kami menulis kode aplikasi *financial tracker* berbasis *mobile* dengan mengikuti aturan *design pattern* MVVM. Peneliti memulai dengan membangun setiap bagian secara terpisah, seperti *Model* untuk mengelola data keuangan, *View* untuk menampilkan antarmuka pengguna, dan *ViewModel* sebagai penghubung yang menangani logika bisnis.

Tantangan utama selama pengembangan adalah pengelolaan *context*, terutama saat sinkronisasi data *real-time* antara *ViewModel* dan *Model*. Karena MVVM memisahkan komponen, pengelolaan *context* menjadi tantangan bagi pengembang yang baru pertama kali menggunakan pola ini. Masalah ini diatasi dengan menggunakan *LiveData* dan *data binding*, yang memungkinkan tampilan diperbarui secara otomatis saat data berubah, serta memastikan penggunaan *context* yang tepat di tiap komponen aplikasi.



Gambar 4. Struktur *folder design pattern* mvvm pada android studio

Hasil pada gambar 4 menunjukkan bahwa struktur *folder* yang rapi, dengan pemisahan antara *View*, *ViewModel*, dan *Model*, memudahkan pengembangan aplikasi. *View* bertugas menampilkan data, sedangkan *ViewModel* mengelola logika bisnis, dan *Model* mengelola data. Pemisahan ini mempercepat pengembangan dan pemeliharaan aplikasi. Selain itu, pemisahan ini juga membuat *debugging* lebih efisien.

Tabel 1. *Black-box testing*

Fitur	Langkah Pengujian	Hasil yang Diharapkan	Hasil Pengujian
Menambah transaksi	Input data transaksi baru (jumlah, kategori, tanggal)	Transaksi berhasil ditambahkan dan tampil di daftar transaksi	Berhasil
Melihat transaksi	Lihat daftar transaksi	Daftar transaksi muncul dengan benar	Berhasil
Mengedit transaksi	Pilih transaksi, ubah detail transaksi	Data transaksi berubah sesuai input	Berhasil
Menghapus transaksi	Pilih transaksi, tekan hapus	Transaksi dihapus dari daftar	Berhasil

Pada tahap pengujian, aplikasi diuji menggunakan *black-box testing* untuk memastikan fitur CRUD berfungsi dengan baik. Pengujian ini dilakukan tanpa memeriksa kode internal, fokus hanya pada input dan output. Selain itu, pemisahan ini juga membantu dalam mengidentifikasi kesalahan dengan lebih cepat dan memastikan setiap komponen bekerja sesuai perannya masing-masing. Hasil dari pengujian ini memberikan keyakinan bahwa sistem berjalan sesuai spesifikasi yang diharapkan.

Hasil pengujian bisa dilihat pada tabel 1, pengujian dimulai dengan fitur menambah transaksi memungkinkan data transaksi baru ditambahkan dan tampil di daftar transaksi dengan benar. Fitur melihat transaksi menampilkan daftar riwayat transaksi dengan tepat. Pada fitur mengedit transaksi, perubahan data sesuai input pengguna, sedangkan fitur menghapus transaksi berhasil menghapus transaksi dari aplikasi. Semua fitur ini berfungsi sesuai hasil yang diharapkan, menunjukkan sistem berjalan dengan baik.

Pengujian performa dilakukan dengan *android profiler*, membandingkan aplikasi sebelum dan sesudah penerapan MVVM. Hasilnya menunjukkan peningkatan efisiensi, terutama dalam manajemen memori dan responsivitas. Dengan *design pattern* MVVM, aplikasi menjadi lebih optimal, stabil dan mudah untuk di-*debug*, terutama pada perangkat dengan spesifikasi rendah.

Tabel 2. Perbandingan waktu eksekusi antara sebelum dan sesudah menggunakan mvvm

Method	Waktu Eksekusi Sebelum MVVM (µs)	Waktu Eksekusi Setelah MVVM (µs)
RenderThread	318,011,766	80,137,318
MainThread	42,145,851	30,418,278

Tabel 2 menyoroti perbandingan waktu eksekusi yang menunjukkan peningkatan performa signifikan setelah penerapan *design pattern* MVVM. Sebelum menggunakan MVVM, *RenderThread* membutuhkan waktu 318.011.766 µs, dan setelah penerapan MVVM, waktu ini berkurang menjadi 80.137.318 µs, menghasilkan peningkatan performa sebesar 74,8%. Pada *MainThread*, waktu eksekusi berkurang dari 42.145.851 µs menjadi 30.418.278 µs, yang berarti peningkatan sebesar 27,8%. Hasilnya menunjukkan bahwa penerapan *design pattern* MVVM membawa peningkatan performa yang sangat signifikan, dengan efisiensi yang lebih baik pada kedua *thread*. Ini membuktikan bahwa MVVM tidak hanya membuat aplikasi lebih responsif, tetapi juga mengoptimalkan penggunaan CPU secara keseluruhan.

Tahap pemeliharaan dilakukan jika ada *bug*, dan penyelesaiannya dilakukan secara sistematis. Jika *error* terjadi pada data, pemeriksaan dilakukan pada *Model*, yang bertanggung jawab atas pengelolaan logika dan pengambilan data dari basis data atau API. Jika *error* terkait tampilan, maka yang diperiksa adalah *View*, untuk memastikan elemen UI seperti *layout* atau *binding* data berfungsi dengan baik. Setiap *bug* atau *error* yang ditemukan akan dicatat dalam sistem manajemen versi seperti *Git*, termasuk deskripsi lengkap tentang perbaikan yang dilakukan. Versi pembaruan dicatat untuk memudahkan pelacakan perubahan dan *rollback* jika diperlukan di masa mendatang.

Pembahasan

Pada tahap analisis, kebutuhan aplikasi dibagi menjadi fungsional dan non-fungsional. Secara fungsional, fitur-fitur utama seperti menambah transaksi, menampilkan data transaksi, memperbarui data transaksi dan menghapus data transaksi yang digunakan untuk menguji optimalisasi *design pattern* MVVM. Fitur ini dirancang untuk memungkinkan pengguna mengelola transaksi keuangan sehari-hari dengan lebih efisien dan terstruktur. Sementara itu, kebutuhan non-fungsional seperti spesifikasi perangkat keras, seperti laptop dengan RAM minimal 8 GB dan *smartphone* dengan RAM 4 GB, diperlukan agar aplikasi berjalan lancar

selama proses pengembangan dan pengujian. Spesifikasi ini juga bertujuan untuk memastikan aplikasi tetap berfungsi optimal pada perangkat dengan spesifikasi rendah.

Setelah menentukan spesifikasi perangkat, penggunaan data *dummy* dalam pengujian menjadi bagian penting dalam simulasi pencatatan keuangan harian. Data *dummy* ini mencakup berbagai fitur CRUD yakni menambah transaksi, menampilkan data transaksi, memperbarui data transaksi dan menghapus data transaksi. Penggunaan data ini memastikan setiap fitur aplikasi berfungsi sesuai dengan kebutuhan pengguna nyata, menciptakan skenario penggunaan realistis untuk menguji relevansi konten aplikasi.

Pada tahap desain, penerapan *design pattern* MVVM terbukti efektif dalam memisahkan *Model*, *View*, dan *ViewModel*. Pemisahan ini memberikan keuntungan dalam hal struktur kode dan pemeliharaan aplikasi. Misalnya, ketika pengguna menambahkan transaksi, *ViewModel* menangani pengelolaan data dan memvalidasi input sebelum disimpan melalui DAO ke basis data. Dengan pemisahan ini, pengembangan dapat dilakukan lebih independen dan modular, sehingga setiap perubahan logika bisnis atau tampilan tidak mengganggu bagian lain dari aplikasi.

Pada tahap implementasi, pengembang menulis kode aplikasi mengikuti *design pattern* MVVM. Setiap komponen dibangun secara terpisah, sehingga pengembangan dapat dilakukan secara paralel dan lebih efisien. Tantangan utama dalam implementasi adalah pengelolaan *context* dan sinkronisasi data *real-time* antara *ViewModel* dan *Model*. Masalah ini diatasi dengan menggunakan *LiveData* dan data *binding*, yang memungkinkan antarmuka diperbarui otomatis ketika data berubah, memastikan penggunaan *context* yang tepat di setiap komponen aplikasi.

Pengujian *black-box* terhadap fitur CRUD menunjukkan hasil yang sangat memuaskan. Fitur menambah transaksi, pengujian dilakukan dengan menginput data transaksi baru seperti jumlah, kategori, dan tanggal. Hasil yang diharapkan adalah transaksi berhasil ditambahkan dan muncul di daftar transaksi, dan ini terbukti berjalan dengan baik. Selanjutnya, pada fitur melihat transaksi, daftar transaksi berhasil ditampilkan dengan benar sesuai yang diharapkan. Fitur mengedit transaksi juga diuji dengan mengubah detail transaksi, dan hasilnya sesuai dengan input pengguna, menunjukkan bahwa data transaksi dapat diperbarui dengan lancar. Terakhir, fitur menghapus transaksi diuji dengan memilih dan menghapus transaksi dari daftar, dan hasil pengujiannya membuktikan bahwa transaksi tersebut berhasil dihapus. Semua langkah pengujian ini berjalan sukses, menunjukkan bahwa aplikasi mampu menangani pencatatan keuangan pengguna dengan baik.

Hasil pengujian performa dengan *android profiler* menunjukkan peningkatan signifikan setelah penerapan MVVM. Optimalisasi waktu eksekusi terlihat jelas, dengan peningkatan performa *RenderThread* sebesar 74,8% dan *MainThread* sebesar 27,8%. Peningkatan ini terjadi karena adanya pemisahan logika aplikasi dari tampilan (*View*) dalam arsitektur MVVM. Pemisahan ini memungkinkan logika bisnis dan tampilan bekerja secara mandiri namun sinkron, sehingga sistem dapat mengeksekusi proses lebih cepat dan efisien.

Secara teknis, penerapan MVVM tidak hanya meningkatkan performa dan pemeliharaan aplikasi, tetapi juga memaksimalkan efisiensi penggunaan sumber daya. Data *binding* dua arah memastikan bahwa tampilan diperbarui otomatis ketika data berubah, mempercepat proses *rendering* dan mengurangi potensi kesalahan dalam sinkronisasi data. Hal ini membuat aplikasi lebih efisien dalam pengunaan sumber daya yang ada.

Pada tahap pemeliharaan, struktur *folder* yang rapi dengan pemisahan antara *View*, *ViewModel*, dan *Model* mempermudah proses *debugging* dan pemeliharaan aplikasi. Pemisahan ini memungkinkan pengembang untuk melacak *bug* dan memperbaikinya tanpa mengganggu komponen lain dari aplikasi, membuat pengembangan lebih terukur dan berkelanjutan, bahkan ketika aplikasi dikembangkan oleh pengembang yang berbeda.

Perbandingan dengan MVC menunjukkan bahwa MVVM unggul dalam hal memisahkan logika bisnis dari tampilan. Sementara pada MVC, logika sering kali menumpuk di *Controller*, menyebabkan kode menjadi lebih rumit, dalam MVVM, *ViewModel* bertanggung jawab penuh atas interaksi data, sehingga UI bebas dari logika bisnis yang kompleks, membuat kode lebih bersih dan terstruktur. Namun, penerapan MVVM juga menghadapi tantangan, terutama dalam pengelolaan *context* antar komponen, terutama bagi pengembang yang belum familiar dengan pola ini. Manajemen *state* dan *lifecycle* komponen membutuhkan pemahaman yang lebih dalam, terutama dalam aplikasi yang membutuhkan interaksi data *real-time*.

Penelitian ini juga menutupi celah yang belum dieksplorasi oleh studi sebelumnya, seperti yang ditunjukkan oleh Ediagbonya & Tioluwani (2022) dan Belgacem (2024), yang fokus pada *fintech* tetapi tidak mendalami aspek teknis penerapan MVVM. Penelitian yang kami lakukan menunjukkan bahwa MVVM tidak hanya meningkatkan performa aplikasi secara signifikan, tetapi juga mengurangi *lag* dan memperbaiki pengalaman pengguna dalam mengelola keuangan secara *real-time*. Meskipun hasil pengujian performa menunjukkan peningkatan yang signifikan, tantangan dalam manajemen *context* tetap ada. Kedepannya, pengembangan aplikasi dapat ditingkatkan dengan fitur-fitur baru seperti pengalokasian dana dan pengingat otomatis untuk membantu pengguna mengelola keuangan dengan lebih baik.

SIMPULAN

Penelitian ini menunjukkan bahwa penerapan *design pattern* MVVM secara signifikan meningkatkan performa dan efisiensi aplikasi *mobile financial tracker*, dengan peningkatan efisiensi sebesar 74,8% pada *RenderThread* dan 27,8% pada *MainThread*. Peningkatan ini membuktikan bahwa MVVM efektif mengurangi beban kerja sistem, mempercepat *rendering*, dan memudahkan sinkronisasi data *real-time* yang tentunya mengoptimalkan kinerja sistem. Pemisahan komponen *Model*, *View*, dan *View Model* tidak hanya membuat kode lebih terstruktur dan mudah diuji, tetapi juga meningkatkan responsivitas aplikasi. Penelitian ini berkontribusi dengan mengaplikasikan MVVM pada aplikasi pencatatan keuangan, yang belum banyak dijelajahi sebelumnya, dan membuka peluang baru bagi pengembangan aplikasi mobile yang lebih efisien dan responsif

REFERENSI

Affandi, L., Apriyani, M. E., & Putra, A. M. (2020). Analisis Response Metrics Terhadap Arsitektur Monolithic dan Microservices dalam Implementasi Aplikasi Kompen. *Jurnal Teknologi Informasi*, *11*(2), 48–53. https://doi.org/10.36382/jti-tki.v11i2.495

Alfathar, M. I., Pamungkas, B. C., Darwaman, B. S., Syafei, A. F., Dwinanto, M. R., Ghifari, M. A., Septiani, N. W. P., & Lestari, M. (2024). Penerapan MVVM (Model View Viewmodel) Pada Pengembangan Aplikasi Bank Sampah Digital. *Jurnal Riset Dan Aplikasi Mahasiswa Informatika (JRAMI)*, *5*(2). 406-414. https://doi.org/10.30998/jrami.v5i2.11071

Anam, M. K., & Anwar, R. (2020). Penerapan Aplikasi Pendukung Touring pada Komunitas Motor Berbasis Android. *Edumatic: Jurnal Pendidikan Informatika*, *4*(1), 1–10. https://doi.org/10.29408/edumatic.v4i1.1980

Belgacem, S. B., Khatoon, G., Bala, H., & Alzuman, A. (2024). The Role of Financial Technology on the Nexus Between Demographic, Socio-economic, and Psychological Factors, and the Financial Literacy Gap. *SAGE Open*, *14*(2), 1-12. https://doi.org/10.1177/21582440241255678

Ediagbonya, V., & Tioluwani, C. (2023). The role of fintech in driving financial inclusion in developing and emerging markets: issues, challenges and prospects. *Technological Sustainability*, *2*(1), 100–119. https://doi.org/10.1108/TECHS-10-2021-0017

Epiloksa, H. A., Kusumo, D. S., & Adrian, M. (2022). Effect Of MVVM Architecture Pattern on Android Based Application Performance. *Jurnal Media Informatika Budidarma*, *6*(4), 1949-1955. https://doi.org/10.30865/mib.v6i4.4545

Fajri, A. R., (2022). *Penerapan Design Pattern MVVM dan Clean Architecture Pada Pengembangan Aplikasi Android (Studi Kasus: Aplikasi Agree).* DSpace Repository. https://dspace.uii.ac.id/handle/123456789/40624

Gunawan, D., Cahyo Utomo, I., Yasin Al Irsyadi, F., Afriantari Puspa Putri, D., Imaduddin, H., Zainal Abidin, A., Aziz Bima Anggita, N., & Sasika Rani, D. (2023). Implementasi MERN Stack pada Pengembangan Sistem Penerimaan Peserta Didik Baru. *Jurnal Swabumi*, *11*(2), 102–110. https://doi.org/https://doi.org/10.31294/swabumi.v11i2.15965

Indrawan, D., Kusumo, D. S., & Puspitasari, S. Y. (2023). Analysis of the Implementation of MVVM Architecture Pattern On Performance Of IOS Mobile-Based Applications. *JIPI (Jurnal Ilmiah Penelitian Dan Pembelajaran Informatika)*, *8*(1), 59–65. https://doi.org/10.29100/jipi.v8i1.3293

Jastradaf, M. L. S. K., & Asriningtias, Y. (2023). Aplikasi Teknologi Augmented Reality untuk Media Pembelajaran Olahraga Renang. *Edumatic: Jurnal Pendidikan Informatika*, *7*(2), 406–415. https://doi.org/10.29408/edumatic.v7i2.23234

Kusumawati, N. P. A., Pramuki, N. M. W. A., Pratiwi, N. P. T. W., Yuliantari, N. P. Y., & Suputra, G. A. (2023). Pelatihan Aplikasi Keuangan Digital Pada Kube Sari Jaya Di Desa Sumerta Kauh. *Jurnal Pengabdian Masyarakat Akademisi*, *2*(4), 164–169. https://doi.org/10.54099/jpma.v2i4.768

Lee, D. C., Seo, K. M., Park, H. M., & Kim, B. S. (2022). Simulation Testing of Maritime Cyber-Physical Systems: Application of Model-View-ViewModel. *Complexity*, *2022*, 1-14. https://doi.org/10.1155/2022/1742772

Maulana, R., & Arivianti, D. (2019). Prototipe sistem informasi pelelangan barang berbasis web sebagai media pengolah informasi data pelelangan. *Jurnal Khatulistiwa Informatika*, *7*(2). 134–140.

Ramadhani, I. H., Suharso, W. & Rizki, D. (2024). Penerapan Desain Pattern Observer Pada Pengembangan Aplikasi Android (Studi Kasus: Aplikasi KataFilm). *Jurnal Repositor*, *6*(1), 1-10. https://doi.org/10.22219/repositor.v6i1.31202

Rismayani., Patasik, M., Layuk, N. S., Saputra, S., & Muhajir, A. (2022). Aplikasi tracking rekreasi dan aktivitas menggunakan model view viewmodel di provinsi sulawesi selatan. *Csrid (Computer Science Research and Its Development Journal)*, *14*(2), 176-187. https://doi.org/10.22303/csrid.14.2.2022.176-187

Riyadhi, I. M., Purnamasari, I. & Prihandani, K. (2023). Penerapan Pola Arsitektur Mvvm Pada Perancangan Aplikasi Pengaduan Masyarakat Berbasis Android. *Infotech Journal*, *9*(1), 147–158. https://doi.org/10.31949/infotech.v9i1.5246

Trivaika, E. (2022). Perancangan aplikasi pengelola keuangan pribadi berbasis android. *Nuansa Informatika*, 16(1), 33-40. https://doi.org/10.25134/nuansa.v16i1.4670

Vijaywargi, A., & Boddapati, U. K. (2024). Architectural Patterns in Android Development: Comparing MVP, MVVM, and MVI. *International Journal for Research in Applied Science and Engineering Technology*, *12*(4), 4611–4616. https://doi.org/10.22214/ijraset.2024.60762

Wang, J., Ji, M., Han, Y., & Wu, Y. (2024). Development and Usability Testing of a Mobile App–Based Clinical Decision Support System for Delirium: Randomized Crossover Trial. *JMIR Aging*, *7*(e51264), 1-12. https://doi.org/10.2196/51264

Wiyana, A. S., Putera, M. I. A., & Natasia, S. R. (2021). Sistem Presensi Online Menggunakan Arsitektur Pengembangan Perangkat Lunak Model-View-Viewmodel. *Teknika*, *10*(3), 214–224. https://doi.org/10.34148/teknika.v10i3.398