RANCANG BANGUN VISUAL STUDIO CODE EXTENSION UNTUK DETEKSI CODE SMELL BERBASIS CLEAN ARCHITECTURE PADA PROYEK LARAVEL

TUGAS AKHIR

Disusun oleh :
Ahmad Fatih Hibatillah
22106050032

PROGRAM STUDI INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI SUNAN KALIJAGA
YOGYAKARTA
2026

LEMBAR PENGESAHAN

KEMENTERIAN AGAMA
UNIVERSITAS ISLAM NEGERI SUNAN KALIJAGA
FAKULTAS SAINS DAN TEKNOLOGI
Jl. Marsda Adisucipto Telp. (0274) 540971 Fax. (0274) 519739 Yogyakarta 55281

PENGESAHAN TUGAS AKHIR

Nomor : B-1192/Un.02/DST/PP.00.9/06/2026

Tugas Akhir dengan judul : Rancang Bangun Visual Studio Code Extension untuk Deteksi Code Smell Berbasis Clean Architecture pada Proyek Laravel

yang dipersiapkan dan disusun oleh:

Nama : AHMAD FATIH HIBATILLAH
Nomor Induk Mahasiswa : 22106050032
Telah diujikan pada : Senin, 25 Mei 2026
Nilai ujian Tugas Akhir : A

dinyatakan telah diterima oleh Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta

TIM UJIAN TUGAS AKHIR

Ketua Sidang

Dr. Fitri Wulandari, S.Si., M.Kom.
SIGNED

Valid ID: 6a2226cd960e2

Penguji I

Dr. Ir. Aulia Faqih Rifa'i, M.Kom.
SIGNED

Valid ID: 6a22224354 2ca

Penguji II

Ir. Muhammad Didik Rohmad Wahyudi, S.T., MT.
SIGNED

Valid ID: 6a210b5e41de1

Yogyakarta, 25 Mei 2026
UIN Sunan Kalijaga
Dekan Fakultas Sains dan Teknologi

Prof. Dr. Dra. Hj. Khurul Wardati, M.Si.
SIGNED

Valid ID: 6a2239b5ed6ec

LEMBAR PERNYATAAN

SURAT PERNYATAAN KEASLIAN

Yang bertanda tangan dibawah ini:

Nama : Ahmad Fatih Hibatillah
NIM : 22106050032
Program Studi : Informatika
Fakultas : Sains dan Teknologi

Dengan ini menyatakan bahwa isi skripsi ini tidak terdapat karya yang pernah diajukan untuk memperoleh gelar sarjana di suatu Perguruan Tinggi dan sesungguhnya skripsi ini merupakan hasil pekerjaan penulis sendiri sepanjang pengetahuan penulis, bukan duplikasi atau saduran dari karya orang lain kecuali bagian tertentu yang penulis ambil sebagai bahan acuan. Apabila terbukti pernyataan ini tidak benar, sepenuhnya menjadi tanggung jawab penulis.

Yogyakarta, 18 Mei 2026

Ahmad Fatih Hibatillah

LEMBAR PERSETUJUAN

Universitas Islam Negeri Sunan Kalijaga FM-UINSK-BM-05-03/R0

SURAT PERSETUJUAN SKRIPSI/TUGAS AKHIR

Hal : Persetujuan Skripsi / Tugas Akhir
Lamp :

Kepada
Yth. Dekan Fakultas Sains dan Teknologi
UIN Sunan Kalijaga Yogyakarta
di Yogyakarta

Assalamu'alaikum wr. wb.

Setelah membaca, meneliti, memberikan petunjuk dan mengoreksi serta mengadakan perbaikan seperlunya, maka kami selaku pembimbing berpendapat bahwa skripsi Saudara:

Nama : Ahmad Fatih Hibatillah
NIM : 22106050032
Judul Skripsi : Rancang Bangun Visual Studio Code Extension Untuk Deteksi Code Smell Berbasis Clean Architecture Pada Proyek Laravel

sudah dapat diajukan kembali kepada Program Studi Informatika Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta sebagai salah satu syarat untuk memperoleh gelar Sarjana Strata Satu dalam Program Studi Informatika.

Dengan ini kami mengharap agar skripsi/tugas akhir Saudara tersebut di atas dapat segera dimunaqasyahkan. Atas perhatiannya kami ucapkan terima kasih.

Wassalamu'alaikum wr. wb.

Yogyakarta, 18 Mei 2026
Pembimbing,

Dr. Fitri Wulandari, S.Si., M.Kom.
NIP. 19741016 200003 2 002

LEMBAR PEDOMAN PENGGUNAAN TUGAS AKHIR

Tugas Akhir ini tidak dipublikasikan, tetapi tersedia di perpustakaan dalam lingkungan UIN Sunan Kalijaga Yogyakarta, diperkenankan dipakai sebagai referensi kepustakaan, tetapi pengutipan harus seizin penyusun, dan harus menyebutkan sumbernya sesuai dengan kebiasaan ilmiah. Dokumen Tugas Akhir ini merupakan hak milik UIN Sunan Kalijaga Yogyakarta.

ABSTRAK

Pengembangan perangkat lunak modern menuntut kualitas struktur kode yang baik untuk menekan biaya pemeliharaan yang tinggi. Namun, pada proyek berbasis kerangka kerja Laravel, fleksibilitas implementasi sering kali memicu munculnya *code smell* seperti *fat controller*, *high complexity*, dan *direct database access* yang berdampak pada menurunnya kualitas struktur kode. Pengembangan ini bertujuan untuk merancang dan membangun ekstensi Visual Studio Code untuk mendeteksi *code smell* tersebut dan memfasilitasi restrukturisasi kode (*auto-refactoring*) berbasis *Clean Architecture*. Metode yang digunakan dalam pengembangan ekstensi ini adalah *Extreme Programming* (XP) dengan memanfaatkan teknologi *Tree-sitter* untuk menganalisis *Abstract Syntax Tree* (AST) dari kode sumber PHP secara *real-time*. Ekstensi ini secara otomatis memberikan peringatan diagnostik pada editor dan mengeksekusi *auto-refactoring* untuk memindahkan logika akses basis data dari *controller* ke lapisan *service*. Hasil pengujian melalui *Black-Box Testing* dan *User Acceptance Testing* (UAT) menunjukkan bahwa ekstensi ini berfungsi dengan baik dan mendapatkan tingkat penerimaan yang tinggi dari pengembang ahli dengan nilai rata-rata 4,18 (kategori Baik). Ekstensi ini dapat membantu pengembang mencegah erosi arsitektur, mengurangi utang teknis (*technical debt*), serta meningkatkan modularitas dan kemudahan pemeliharaan sistem dan kualitas struktur kode secara keseluruhan.

Kata Kunci: *Visual Studio Code Extension*, *Code Smell*, *Clean Architecture*, *Auto-refactoring*, Laravel.

ABSTRACT

Modern software development demands high-quality code structures to reduce high maintenance costs. However, in Laravel framework-based projects, implementation flexibility frequently triggers the emergence of code smells such as fat controllers, high complexity, and direct database access, which negatively impact the quality of the code structure. This development aims to design and build a Visual Studio Code extension to detect these code smells and facilitate code restructuring (auto-refactoring) based on Clean Architecture. The method used in developing this extension is Extreme Programming (XP), utilizing Tree-sitter technology to analyze the Abstract Syntax Tree (AST) of PHP source code in real-time. This extension automatically provides diagnostic warnings in the editor and executes auto-refactoring to migrate database access logic from controllers to the service layer. Test results through Black-Box Testing and User Acceptance Testing (UAT) indicate that the extension functions properly and has achieved a high level of acceptance from expert developers with an average score of 4.18 (Good category). This extension can help developers prevent architectural erosion, reduce technical debt, and improve system modularity, maintainability, and overall code structure quality.

Keywords: *Visual Studio Code Extension, Code Smell, Clean Architecture, Auto-refactoring, Laravel.*

MOTTO

"Trying to do better."

Peter Parker

"No one can win every battle, but no man should fall without struggle."

Spider-Man

"*With great power, comes great responsibility."*

Benjamin Parker

HALAMAN PERSEMBAHAN

*Alhamdulillahi Rabbil 'Alamin, d*engan penuh rasa syukur ke hadirat Allah SWT Yang Maha Pengasih dan Maha Penyayang. Atas segala rahmat serta kemudahan-Nya, Tugas Akhir ini dapat diselesaikan. Semoga karya sederhana ini dapat memberikan manfaat dan bernilai amal ibadah. Dengan segenap kerendahan hati, dan rasa cinta yang mendalam, Tugas Akhir ini saya persembahkan kepada:

1. Bapak dan Ibu saya tercinta, yang telah memberikan kasih sayang dan dukungan yang tak pernah putus. Pasti do'a kalian yang melancarkan segala upaya saya.
2. Adik-adik tersayang, yang suatu saat nanti akan berhadapan dengan skripsi, karya ini bisa kalian jadikan gambaran. Semoga bisa membantu atau minimal jadi pengingat kalau tugas akhir itu pasti bisa diselesaikan. Semangat berproses!
3. Almamater Tercinta, UIN Sunan Kalijaga, tempat saya berproses, menimba ilmu, dan mengembangkan diri.

Terima kasih banyak atas semua dukungan yang sudah diberikan. Semoga Allah SWT membalas setiap kebaikan kalian dengan rahmat dan berkah-Nya.

KATA PENGANTAR

Alhamdulillahi Rabbil 'Alamin, segala puji dan syukur penulis panjatkan ke hadirat Allah SWT atas rahmat dan karunia-Nya, sehingga penyusunan Tugas Akhir yang berjudul " Rancang Bangun Visual Studio Code Extension untuk Deteksi Code Smell Berbasis Clean Architecture Pada Proyek Laravel " ini dapat diselesaikan dengan baik. Selawat serta salam semoga senantiasa tercurah kepada junjungan kita, Nabi Muhammad SAW, beserta keluarga, sahabat, dan umatnya.

Penulis menyadari bahwa penyelesaian Tugas Akhir ini tidak lepas dari doa, bimbingan, serta dukungan dari berbagai pihak. Oleh karena itu, dengan segenap kerendahan hati, penulis ingin menyampaikan rasa terima kasih yang sebesar-besarnya kepada:

1. Prof. Noorhaidi Hasan S.Ag., M.A., M.Phil., Ph.D. selaku Rektor UIN Sunan Kalijaga Yogyakarta.
2. Prof. Dr. Dra. Hj. Khurul Wardati, M.Si. selaku Dekan Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta.
3. Bapak Dr. Muhammad Mustakim, S.T. M.T. selaku Ketua Program Studi Informatika Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta.
4. Bapak Dr. Agus Mulyanto, S.Si., M.Kom., ASEAN Eng. selaku dosen penasihat akademik penulis selama kuliah di UIN Sunan Kalijaga.
5. Ibu Dr. Fitri Wulandari, S.Si., M.Kom. yang telah memberikan masukan berharga dan dukungan penuh dalam penyusunan tugas akhir ini, serta selalu sabar mengarahkan penulis dalam setiap tahapan.
6. Kedua orang tua penulis, Bapak dan Ibu tercinta, yang telah memberikan kasih sayang dan dukungan yang tak pernah putus. Pasti doa kalianlah yang melancarkan setiap upaya penulis hingga mampu sampai pada tahap ini.
7. Teman-teman grup 5cm, yang senantiasa menjadi bagian dari huru-hara, tawa, canda, suka dan duka sejak SMA. Semoga kita selalu menjadi sebuah kisah klasik untuk masa depan.

8. Teman-teman Matdis dan Musang, yang selalu bersedia menjadi tempat singgah dan menciptakan suasana yang interaktif, kompetitif, ekspresif, investigatif, dan informatif. Terima kasih telah menjadi bagian dari kesombongan di masa muda yang indah.
9. Teman-teman Neroform (informatika angkatan 2022). Selamat berjuang mencari kerja, tapi jangan korbankan semua untuk dunia karena karir ini tak ada artinya.
10. Teman-teman Komplek H, yang telah menjadikan asrama bukan sekadar tempat singgah, tapi juga sebagai rumah kedua di bawah langit yang sama.
11. Teman-teman KKN Klepu, yang telah menjadi bagian dari cerita singkat yang berkesan. Terima kasih atas kerja sama dan memori baiknya.

Penulis menyadari bahwa Tugas Akhir ini masih jauh dari kata sempurna dan memiliki banyak ruang untuk perbaikan. Oleh karena itu, penulis sangat terbuka terhadap kritik dan saran yang membangun demi penyempurnaan karya ini. Akhir kata, semoga Tugas Akhir ini dapat memberikan manfaat serta bisa dikembangkan lebih baik lagi di masa depan.

Yogyakarta, 18 Mei 2026
Penulis,

Ahmad Fatih Hibatillah
NIM. 22106050032

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pengembangan perangkat lunak modern menuntut struktur kode yang fungsional sekaligus mudah diuji (*testability*) dan dipelihara (*maintainability*). Dalam praktik industri, tahap pemeliharaan sering kali menyerap sumber daya paling besar, yang dapat mencapai 70% hingga 90% dari total biaya pengembangan keseluruhan [1]. Membengkaknya alokasi pemeliharaan ini pada dasarnya berakar dari pengabaian standar kualitas penulisan kode pada fase awal proyek. Penelitian empiris yang dilakukan oleh Tornhill dan Borg terhadap puluhan basis kode komersial membuktikan bahwa penulisan kode yang buruk secara langsung menciptakan utang teknis (*technical debt*), di mana pengembang dapat kehilangan hingga 42% waktu produktif mereka hanya untuk mengurai dan memperbaiki logika kode yang berantakan dibandingkan membangun fitur baru [2]. Jika dibiarkan, akumulasi utang teknis ini akan memicu pelipatgandaan biaya dan tingkat kesulitan perbaikan di masa depan. Oleh karena itu, menjaga kualitas kode sejak fase awal penulisan menjadi faktor krusial agar perangkat lunak dapat terus berkembang tanpa menimbulkan kompleksitas yang sulit dikendalikan.

Untuk mencapai efisiensi dan kecepatan produksi, pengembang umumnya memanfaatkan kerangka kerja (*framework*) modern seperti Laravel. Namun, penggunaan *framework* tidak secara otomatis menjamin kualitas struktur kode yang baik. Studi empiris yang dilakukan oleh Rahmawati dan Susetyo pada berbagai proyek berbasis Laravel membuktikan bahwa pelanggaran kualitas kode masih sangat sering terjadi di lapangan. Melalui pengujian menggunakan perangkat bantu analisis statis, ditemukan berbagai permasalahan krusial seperti *code smell*, *bug*, serta duplikasi kode dalam jumlah yang signifikan di dalam ekosistem proyek tersebut [3].

Permasalahan kualitas tersebut sering kali berakar pada fleksibilitas arsitektur bawaan *framework* itu sendiri. Meskipun Laravel mengadopsi pola arsitektur *Model-View-Controller* (MVC) untuk memisahkan logika aplikasi,

implementasi praktisnya kerap tidak optimal. Rangkuti dkk. menemukan bahwa pola MVC pada Laravel tidak memberikan batasan yang ketat terhadap distribusi tempat penyimpanan logika bisnis [4]. Kondisi ketiadaan batasan ini membuka celah terjadinya penumpukan logika secara berlebihan pada komponen *controller* (*fat controller*). Selain itu, ketersediaan fitur instan yang sangat aksesibel seperti *Eloquent ORM* juga sering mendorong pengembang untuk melakukan praktik akses basis data secara langsung dari *controller* (*direct database access*), tanpa melewati lapisan abstraksi perantara tambahan seperti *service layer* [4].

Praktik-praktik penulisan kode yang buruk, seperti penumpukan tanggung jawab, akses basis data langsung, dan tingginya kompleksitas logika (*high complexity*), merupakan bentuk nyata dari *code smell*. *Code smell* adalah indikator kelemahan desain yang menyulitkan pembacaan kode, menghambat kolaborasi tim, dan memperbesar kemungkinan masuknya kesalahan sistem [5]. Secara empiris, penelitian pada aplikasi web berbasis PHP membuktikan bahwa keberadaan *code smell* berkorelasi kuat dengan menurunnya tingkat kemudahan pemeliharaan (*maintainability*) dan meningkatnya risiko kerentanan perangkat lunak [6]. Jika dibiarkan secara terus-menerus, kondisi ini akan memicu membesarnya utang teknis dan menyebabkan erosi arsitektur (*architectural erosion*), yaitu kondisi di mana implementasi kode nyata makin menyimpang dari rancangan desain awal yang ditetapkan.

Sebagai solusi untuk mengatasi *code smell* dan mencegah terjadinya erosi arsitektur, diperlukan penerapan dan kepatuhan terhadap pedoman desain yang lebih ketat, salah satunya adalah *Clean Architecture*. Pendekatan ini secara teoretis dan praktis terbukti mampu mempertahankan kualitas sistem dengan menekankan pemisahan tanggung jawab (*separation of concerns*) secara tegas [7], [8]. Dalam *Clean Architecture*, logika bisnis inti diisolasi sepenuhnya dari detail teknis seperti *framework* dan basis data, sehingga arah ketergantungan kode selalu terpusat ke dalam. Apabila prinsip ini diterapkan secara konsisten, kemunculan *code smell* struktural seperti *fat controller* dan *direct database access* dapat dicegah sejak dini karena logika aplikasi akan selalu didistribusikan ke lapisan-lapisan yang terisolasi.

Saat ini, berbagai penelitian telah mengembangkan pendekatan untuk mendeteksi *code smell*, salah satunya menggunakan alat analisis berbasis struktur hierarki pohon sintaks atau *Abstract Syntax Tree* (AST) [5]. Pendekatan AST terbukti jauh lebih efektif karena memungkinkan sistem memahami relasi dan struktur kode secara kontekstual dibandingkan sekadar pencarian pola berbasis teks. Namun demikian, sebagian besar pendekatan dan ekstensi pendeteksi *code smell* yang tersedia di pasaran saat ini masih bersifat pasif. Alat-alat tersebut umumnya hanya memberikan peringatan umum tanpa terintegrasi dengan parameter arsitektur spesifik seperti *Clean Architecture*, serta belum memfasilitasi mekanisme restrukturisasi kode (*auto-refactoring*) secara otomatis untuk membantu pengembang memperbaiki pelanggaran tersebut.

Berdasarkan celah penelitian (*research gap*) dan urgensi permasalahan di atas, diperlukan suatu alat bantu yang mampu mendeteksi *code smell* secara akurat dengan menggunakan prinsip *Clean Architecture* sebagai standar evaluasi struktur kodenya. Sejalan dengan hal tersebut, penulis bermaksud melaksanakan proyek pengembangan perangkat lunak yang dituangkan dalam bentuk tugas akhir dengan judul *"Rancang Bangun Visual Studio Code Extension untuk Deteksi Code Smell Berbasis Clean Architecture pada Proyek Laravel"*. Pengembangan ekstensi ini diharapkan dapat menjadi asisten aktif bagi pengembang guna mendeteksi *code smell* secara *real-time* berbasis AST, serta memfasilitasi eksekusi perbaikan (*auto-refactoring*) agar penerapan prinsip *Clean Architecture* tetap konsisten selama siklus pengembangan perangkat lunak berjalan.

1.2 Rumusan Masalah

Dari latar belakang yang telah diuraikan di atas, dapat dirumuskan sebuah rumusan masalah yaitu: "Bagaimana merancang dan membangun Visual Studio Code Extension yang dapat mendeteksi *code smell* secara akurat dengan mengacu pada prinsip *Clean Architecture* pada proyek perangkat lunak berbasis kerangka kerja *Laravel*?".

1.3 Batasan Masalah

Agar pengembangan tetap fokus pada tujuan yang telah ditetapkan, maka diberikan batasan-batasan masalah sebagai berikut:

1. Proyek yang dikembangkan berbentuk ekstensi yang dikhususkan untuk lingkungan *Integrated Development Environment* (IDE) *Visual Studio Code*.
2. Analisis kode hanya mendukung bahasa pemrograman PHP dan secara spesifik menargetkan proyek yang menggunakan kerangka kerja *Laravel*.
3. Proses analisis dan deteksi penyimpangan kode hanya dilakukan pada file yang berada di dalam folder *controller* (*app/Http/Controllers*), sehingga tidak mencakup lapisan lain seperti *Service, Repository*, maupun *Model.*
4. Proses deteksi penyimpangan kode dilakukan melalui analisis statis menggunakan teknologi *Tree-sitter* (*tree-sitter-php*) untuk membangun *Abstract Syntax Tree* (AST), dengan pendekatan berbasis aturan (*rule-based detection*) yang difokuskan pada *code smell* berupa *fat controller*, *high complexity*, dan *direct database access*.
5. Proses perbaikan otomatis (*auto-refactoring*) hanya dibatasi pada penanganan *direct database access* dengan memindahkan logika akses data ke lapisan *Service*, serta dilakukan menggunakan pendekatan berbasis pola (*pattern-based transformation*) yang hanya mendukung struktur kode sederhana.
6. Sistem memiliki keterbatasan dalam mengenali seluruh variasi model, struktur proyek, serta konteks bisnis aplikasi, sehingga hasil analisis dan *refactoring* yang dihasilkan masih memerlukan validasi dari *developer*.

1.4 Tujuan

Tujuan dari pengembangan ini adalah merancang dan membangun sebuah *Visual Studio Code Extension* yang mampu mendeteksi *code smell* serta melakukan restrukturisasi kode (*auto-refactoring*) pada proyek Laravel berdasarkan prinsip *Clean Architecture*, sehingga struktur kode menjadi lebih modular dan sesuai dengan pemisahan tanggung jawab yang baik.

1.5 Manfaat

Tugas akhir ini diharapkan memberikan manfaat sebagai berikut:

1. Membantu mengurangi akumulasi utang teknis (*technical debt*) yang disebabkan oleh praktik pengembangan tanpa perencanaan arsitektur yang terstruktur pada proyek *Laravel*.
2. Membantu meningkatkan *maintainability* (kemudahan pemeliharaan) dan *testability* (kemudahan pengujian) aplikasi dengan mendorong pemisahan logika bisnis dari lapisan *controller*.
3. Mendukung efisiensi pengembangan melalui fitur deteksi dan perbaikan otomatis (*auto-refactoring*) yang mampu memberikan umpan balik secara langsung (*real-time*) terhadap penyimpangan struktur kode.

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan serangkaian proses perancangan, implementasi, dan pengujian yang telah dilaksanakan untuk menjawab rumusan masalah penelitian, maka dapat ditarik beberapa simpulan sebagai berikut:

1. Keberhasilan Rancang Bangun Ekstensi

   Pengembangan ini telah berhasil merancang dan membangun sebuah *Visual Studio Code Extension* yang memanfaatkan teknologi *Tree-sitter* untuk pembentukan *Abstract Syntax Tree* (AST). Integrasi ini memungkinkan sistem melakukan analisis kode statis secara *real-time* dengan pemahaman mendalam terhadap struktur sintaksis PHP pada proyek *Laravel*.

2. Deteksi *Code Smell* Berbasis *Clean Architecture*

   Sistem mampu melakukan deteksi terhadap tiga jenis *code smell* utama pada lapisan *controller*, yaitu *fat controller*, *high complexity*, dan *direct database access*. Deteksi dilakukan menggunakan pendekatan *rule-based* yang diturunkan dari prinsip pemisahan tanggung jawab (*separation of concerns*) dalam *Clean Architecture*, sehingga pelanggaran struktur kode dapat diidentifikasi secara sistematis.

3. Implementasi Respons Sistem dan Pengelolaan Hasil Analisis

   Sistem berhasil mengimplementasikan mekanisme respons terhadap pelanggaran yang terdeteksi melalui fitur *Code Action* dan *auto-refactoring* untuk kasus *direct database access*. Selain itu, sistem juga menyediakan fitur *CSmell Summary* untuk menampilkan ringkasan pelanggaran dalam proyek, *CSmell Config* untuk mengelola nilai *threshold* setiap *rule* deteksi, serta *Ignore Violation* untuk mengabaikan pelanggaran tertentu sesuai kebutuhan pengguna. Fitur-fitur tersebut membantu pengguna dalam proses analisis, pengelolaan pelanggaran, dan penerapan prinsip *Clean Architecture* secara lebih fleksibel.

4. Validasi Capaian Tujuan melalui UAT

Hasil pengujian *User Acceptance Testing* (UAT) menunjukkan nilai rata-rata keseluruhan sebesar 4,18 (Kategori Baik). Responden memberikan skor tertinggi (4,5) pada kemampuan sistem dalam mengarahkan pemisahan logika, yang membuktikan bahwa tujuan untuk membangun alat bantu yang mendukung penerapan *Clean Architecture* secara konsisten pada proyek *Laravel* telah tercapai.

5.2 Saran

Untuk pengembangan sistem yang lebih komprehensif di masa mendatang, terdapat beberapa saran yang dapat dipertimbangkan:

1. Ekspansi Cakupan Analisis

   Mengingat batasan masalah penelitian ini yang berfokus pada folder *Controller*, pengembangan selanjutnya diharapkan dapat mencakup lapisan lain seperti *Service* dan *Repository* guna menjaga konsistensi arsitektur di seluruh ekosistem proyek.
2. Otomatisasi Transformasi Kompleks

   Menambahkan pola *auto-refactoring* untuk jenis pelanggaran lain seperti *high complexity* dengan algoritma ekstraksi fungsi yang lebih dinamis untuk meningkatkan produktivitas pengembang.

DAFTAR PUSTAKA

[1] S. Rochimah, T. R. Hadiningrum, B. D. Mardiana, D. O. Siahaan, J. A. Rizky, and M. S. Ary, "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.

[2] A. Tornhill and M. Borg, "Code Red: The Business Impact of Code Quality – A Quantitative Study of 39 Proprietary Production Codebases," 2022. [Online]. Available: https://arxiv.org/abs/2203.04374

[3] J. Penerapan Teknologi Informasi dan Komunikasi, A. Febriana Rahmawati, and Y. Alfa Susetyo, "IT-EXPLORE ANALISIS QUALITY CODE MENGGUNAKAN SONARQUBE DALAM SUATU APLIKASI BERBASIS LARAVEL".

[4] M. Y. Rangkuti, H. R. Ilhamsyah, R. Z. Mutaqin, and M. L. Khoirullah, "ANALISIS PENERAPAN ARSITEKTUR MODEL–VIEW–CONTROLLER (MVC) PADA PENGEMBANGAN APLIKASI WEB BERBASIS LARAVEL".

[5] Hamzan Wadi, Kasmawi, and Muhammad Asep Subandri, "Penggunaan Software Metrics Dan Abstract Syntax Tree Untuk Mendeteksi Code Smell Pada Bahasa Pemrograman Python," *JEKIN - Jurnal Teknik Informatika*, vol. 5, no. 1, pp. 61–69, Jan. 2025, doi: 10.58794/jekin.v5i1.900.

[6] A. Rio and F. Brito E Abreu, "PHP code smells in web apps: Evolution, survival and anomalies ☆," *J. Syst. Softw.*, vol. 200, p. 111644, 2023, doi: 10.5281/zenodo.

[7] R. C. Martin, "Clean Architecture."

[8] N. S. P. K. Yadati, "Architecture Design (MVVM + Clean Architecture)," *Journal of Artificial Intelligence, Machine Learning and Data Science*, vol. 1, no. 3, pp. 703–706, Sep. 2023, doi: 10.51219/jaimld/naga-satya-praveen-kumar-yadati/177.

[9] "Laravel - The clean stack for Artisans and agents." Accessed: Apr. 25, 2026. [Online]. Available: https://laravel.com/

[10] B. Moloudi, "Optimizing Architecture in iOS Development: A Comparative Study of MVVM and VIPER using SwiftUI."

[11] R. Li, P. Liang, M. Soliman, and P. Avgeriou, "Understanding software architecture erosion: A systematic mapping study," Mar. 01, 2022, *John Wiley and Sons Ltd*. doi: 10.1002/smr.2423.

[12] by Martin Fowler, K. Beck, J. Brant, W. Opdyke, and don Roberts, "Refactoring: Improving the Design of Existing Code."

[13] N. S. Filho, "Comparative Analysis between Cognitive Complexity and Cyclomatic Complexity in Software Development," *Leaders.Tec.Br*, vol. 2, no. 7, doi: 10.5281/zenodo.14879076.

[14] R. Hu, "An Automated Approach to Check Software Architecture Erosion."

[15] "PHP: Introduction - Manual." Accessed: Apr. 25, 2026. [Online]. Available: https://www.php.net/manual/en/introduction.php

[16] S. Kushwah, C. Bansal, S. Rathore, V. Kate, D. Bargal, and D. Vishwakarma, *TypeScript: An Open-Source Programming Language with Options for Robust Development and Large-Scale Applications*. 2024. doi: 10.1109/ACROSET62108.2024.10743426.

[17] A. Sergeyuk, I. Zakharov, E. Koshchenko, and M. Izadi, "Human-AI Experience in Integrated Development Environments: A Systematic Literature Review," Jan. 2026, doi: 10.1007/s10664-025-10793-0.

[18] "Static Code Analysis: Tools and Best Practices | Cycode." Accessed: Apr. 06, 2026. [Online]. Available: https://cycode.com/blog/static-code-analysis/

[19] "Extension API | Visual Studio Code Extension API." Accessed: Apr. 06, 2026. [Online]. Available: https://code.visualstudio.com/api

[20] A. Nuraminah and A. Ammar, "Damerau-Levenshtein Distance Algorithm Based on Abstract Syntax Tree to Detect Code Plagiarism," *Scientific Journal of Informatics*, vol. 11, no. 1, pp. 11–20, Dec. 2023, doi: 10.15294/sji.v11i1.48064.

[21] "Implementing Parser and PSI | IntelliJ Platform Plugin SDK." Accessed: Apr. 11, 2026. [Online]. Available: https://plugins.jetbrains.com/docs/intellij/implementing-parser-and-psi.html

[22] "Tree-sitter: An incremental parsing system for programming tools." Accessed: Apr. 07, 2026. [Online]. Available: https://tree-sitter.github.io/tree-sitter/

[23] J. J. Wijadi, B. Ghiffar, and I. B. K. Manuaba, "A Study on the Performance of WebAssembly Versus JavaScript in an SVG Editor Environment," in *2024*

International Symposium on Parallel Computing and Distributed Systems (PCDS), 2024, pp. 1–5. doi: 10.1109/PCDS61776.2024.10743854.

[24] M. Alharbi and M. Alshayeb, "A Comparative Study of Automated Refactoring Tools," *IEEE Access*, vol. 12, pp. 18764–18781, 2024, doi: 10.1109/ACCESS.2024.3361314.

[25] B. Nugroho, N. F. Azhar, and P. C. Subekti, "REFACTORING APLIKASI XYZ MENGGUNAKAN PENDEKATAN CLEAN ARCHITECTURE," *Information System Journal*, vol. 7, no. 01, pp. 20–33, Jun. 2024, doi: 10.24076/infosjournal.2024v7i01.1579.

[26] A. Supriyatna and D. Puspitasari, "Implementation of Extreme Programming Method in Web Based Digital Report Value Information System Design," *International Journal of Information System & Technology Akreditasi*, vol. 5, no. 1, pp. 67–75, 2021.

[27] D. Jean Cross Sihombing, "Enhancing Project Visibility: A Study on Construction Project Dashboard Development with Extreme Programming-Denny Jean Cross Sihombing Enhancing Project Visibility: A Study on Construction Project Dashboard Development with Extreme Programming," *Informatika dan Sains*, vol. 14, no. 01, p. 2024, doi: 10.54209/infosains.v14i01.3739.

[28] T. Haryati, W. Handini, and Y. M. Aprita, "Penerapan Extreme Programming dalam Pembangunan Sistem Informasi Pembayaran Tagihan PAM pada PAMDES Margakaya Sejahtera Karawang," *Jurnal Penelitian Inovatif*, vol. 4, no. 1, pp. 129–136, Feb. 2024, doi: 10.54082/jupin.278.

[29] S. Astiti, "RESOLUSI : Rekayasa Teknik Informatika dan Informasi Penerapan Metode Extreme Programming Pada Rancang Bangun Website Company Profile," *Media Online*, vol. 3, no. 3, pp. 114–124, 2023, [Online]. Available: https://djournals.com/resolusi

[30] I. G. B. Premana Putra, M. Sudarma, and I. B. G. Manuaba, "Penerapan Metode Extreme Programming pada Rancang Bangun Sistem Analisis Sentimen Portal Berita," *Jurnal Teknologi Informasi dan Ilmu Komputer*, vol. 10, no. 6, pp. 1369–1378, Dec. 2023, doi: 10.25126/jtiik.2023106904.

[31] Muhammad Helmi Satria Fedianto, Firza Prima Aditiawan, and Muhammad Muharrom Al Haromainy, "Pengujian Sistem Jaringan Dokumentasi Dan Informasi Menggunakan Black Box Testing Dan White Box Testing," *Jurnal Publikasi Sistem Informasi dan Manajemen Bisnis*, vol. 3, no. 1, pp. 213–221, Dec. 2023, doi: 10.55606/jupsim.v3i1.2447.

[32] M. T. Abdillah *et al.*, "Implementasi Black box Testing dan Usability Testing pada Website Sekolah MI Miftahul Ulum Warugunung Surabaya," *Jurnal Ilmu Komputer dan Desain Komunikasi Visual*, vol. 8, no. 1, 2023.

[33] M. Hennink and B. N. Kaiser, "Sample sizes for saturation in qualitative research: A systematic review of empirical tests," *Soc. Sci. Med.*, vol. 292, p. 114523, 2022, doi: https://doi.org/10.1016/j.socscimed.2021.114523.

[34] S. K. Ahmed, "Sample size for saturation in qualitative research: Debates, definitions, and strategies," *Journal of Medicine, Surgery, and Public Health*, vol. 5, p. 100171, 2025, doi: https://doi.org/10.1016/j.glmedi.2024.100171.

[35] Sugiyono, *Metode Penelitian Kuantitatif, Kualitatif, dan R&D*. Bandung: Alfabeta, 2013.