Rancang Bangun Ekstensi Visual Studio Code untuk Otomatisasi Dokumentasi, Pengujian, dan Analisis Keamanan pada Application Programming Interface (API)

TUGAS AKHIR

Disusun Oleh :

Muhammad Afrizal

22106050078

PROGRAM STUDI S1 INFORMAATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI SUNAN KALIJAGA
YOGYAKARTA
2026

LEMBAR PERNYATAAN KEASLIAN

Yang bertanda tangan di bawah ini:

Nama	: Muhammad Afrizal
NIM	: 22106050078
Program Studi	: Informatika
Fakultas	: Sains dan Teknologi

Menyatakan dengan sesungguhnya, bahwa skripsi saya yang berjudul " **Rancang Bangun Ekstensi Visual Studio Code untuk Otomatisasi Dokumentasi, Pengujian, dan Analisis Keamanan pada Application Programming Interface (API)**" merupakan penelitian saya sendiri, tidak terdapat karya yang pernah diajukan untuk memperoleh gelar kesarjanaan di suatu perguruan tinggi dan bukan plagiasi karya orang lain kecuali secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka. Sebagai salah satu syarat untuk memperoleh gelar Sarjana Program Studi Informatika pada Fakultas Sains dan Teknologi, Universitas Islam Negeri Sunan Kalijaga.

Yogyakarta, 25 Februari 2026
Yang membuat pernyataan,

Muhammad Afrizal
22106050078

LEMBAR PERSETUJUAN SKRIPSI / TUGAS AKHIR

Hal : Persetujuan Skripsi / Tugas Akhir
Lamp : -

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

Assalamu'alaikum wr. wb.

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

Nama	:	Muhammad Afrizal
NIM	:	22106050078
Judul Skripsi	:	Rancang Bangun Ekstensi Visual Studio Code Untuk Otomatisasi Dokumentasi, Pengujian, Dan Analisis Keamanan Pada Application Programming Interface (API)

Sudah dapat diajukan 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 Saudari dapat segera dimunaqosyahkan. Atas perhatiannya saya ucapkan terima kasih.

Wassalamu'alaikum wr. wb.

Yogyakarta, 25 Februari 2026
Pembimbing,

Muhammad Didik Rohmad Wahyudi, S.T., M.T.
NIP. 19760812 200901 1 015

LEMBAR PENGESAHAN TUGAS AKHIR

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-500/Un.02/DST/PP.00.9/03/2026

Tugas Akhir dengan judul : Rancang Bangun Ekstensi Visual Studio Code untuk Otomatisasi Dokumentasi, Pengujian, dan Analisis Keamanan pada Application Programming Interface (API)

yang dipersiapkan dan disusun oleh:

Nama : MUHAMMAD AFRIZAL
Nomor Induk Mahasiswa : 22106050078
Telah diujikan pada : Rabu, 04 Maret 2026
Nilai ujian Tugas Akhir : A

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

TIM UJIAN TUGAS AKHIR

Ketua Sidang

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

Valid ID: 69a933e824a5d

Penguji I

Muhammad Mustakim, S.T. M.T.
SIGNED

Valid ID: 69a7d36f347c4

Penguji II

Dr. Ir. Sumarsono, S.T., M.Kom.
SIGNED

Valid ID: 69a8231a423ac

Yogyakarta, 04 Maret 2026
UIN Sunan Kalijaga
Dekan Fakultas Sains dan Teknologi

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

Valid ID: 69a9373452d1b

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.

Rancang Bangun Ekstensi Visual Studio Code untuk Otomatisasi Dokumentasi, Pengujian, dan Analisis Keamanan pada Application Programming Interface (API)

Muhammad Afrizal

22106050078

ABSTRAK

Fragmentasi alat kerja (*tool fragmentation*) dan tingginya intensitas perpindahan konteks (*context switching*) kerap menurunkan produktivitas pengembang dalam siklus pengembangan API, khususnya pada kerangka kerja Next.js. Penelitian ini bertujuan untuk merancang dan mengembangkan ekstensi *Visual Studio Code* bernama "NextJS API Inspector" guna mengotomatisasi dokumentasi, simulasi pengujian, dan analisis keamanan dini (*Shift-Left Testing*) menggunakan asisten kecerdasan buatan Google Gemini. Penelitian ini menerapkan metode *Design and Development* (D&D) dengan pengujian kelayakan bersandar pada model kualitas ISO/IEC 25010:2023, analisis statis SonarQube, dan evaluasi *System Usability Scale* (SUS) terhadap 30 responden. Hasil pengujian menunjukkan ekstensi ini berhasil memenuhi 9 karakteristik utama ISO 25010 dengan tingkat keberhasilan fungsional dan kompatibilitas 100%, skor efisiensi performa 100/100, serta memperoleh *Rating A* (0 *bugs*, 0 *vulnerabilities*) pada aspek keandalan, keamanan, dan keterpeliharaan. Evaluasi penerimaan pengguna menghasilkan skor SUS 78,75 yang masuk dalam kategori *Good/Acceptable*. Disimpulkan bahwa ekstensi ini terbukti fungsional, aman, dan secara signifikan mampu meningkatkan efisiensi alur kerja pengembang dengan mereduksi beban kognitif akibat perpindahan aplikasi.

Kata kunci : *VS Code, IDE, Ekstensi, Next.js, API, Generative AI, ISO/IEC 25010.*

Rancang Bangun Ekstensi Visual Studio Code untuk Otomatisasi Dokumentasi, Pengujian, dan Analisis Keamanan pada Application Programming Interface (API)

Muhammad Afrizal

22106050078

ABSTRACT

Tool fragmentation and the high intensity of context switching frequently reduce developer productivity during the API development lifecycle, particularly within the Next.js framework. This research aims to design and develop a Visual Studio Code extension named "NextJS API Inspector" to automate documentation, testing simulation, and early security analysis (Shift-Left Testing) utilizing the Google Gemini artificial intelligence assistant. The study applied the Design and Development (D&D) methodology, with feasibility testing based on the ISO/IEC 25010:2023 quality model, SonarQube static analysis, and a System Usability Scale (SUS) evaluation involving 30 respondents. The testing results demonstrated that the extension successfully met 9 main characteristics of ISO 25010 with a 100% functional and compatibility success rate, a 100/100 performance efficiency score, and achieved a Rating A (0 bugs, 0 vulnerabilities) in reliability, security, and maintainability aspects. The user acceptance evaluation yielded an SUS score of 78.75, falling into the Good/Acceptable category. In conclusion, this extension is proven to be functional, secure, and significantly enhances developer workflow efficiency by reducing the cognitive load caused by application switching.

Keyword : *VS Code, IDE, Ekstension, Next.js, API, Generative AI, ISO/IEC 25010.*

MOTTO

”Jangan Hinakan Batinmu Dengan Kepandaian Akal dan Pikiranmu”
(M.R H.S.M Irfa’i Nachrawi An - Naqsyabandie)

"Ilahi anta maqsudi waridhoka matlubi"

MOTTO

"Fortis Fortuna Adiuvat"

”Setelah kamu memutuskan untuk menjadi seperti itu, Jadi jika kamu mati karena berjuang untuk menjadi itu, Seharusnya kamu bahagia bukan”

Hanya butuh satu saja satu alasan, Untuk sudah menyerah atau terus berjalan
Lihat saja, takut saja, sudah sajalah. Lihat saja, jalan saja, sikat sajalah
Syarat pertama adalah percaya. Syarat kedua, lihat syarat pertama
(Syarat - FSTVLST)

Pelan pasti ku kabulkan, Segala catatan harapmu
Tentang masa depan, tentang masa terang, Kebul jalan kuterjang
(Gemilang – Perunggu)

Sampai kapan kau diam di kepala, Sedang dunia begitu luasnya
Jangan mati membusuk di sana, Biarkan kakimu mengembara
(Track 8 – The Jeblogs)

Take what you need, and be on your way. And stop crying your heart out.
(*Stop Crying Your Heart Out – Oasis*)

Alasan mengapa kita merasakan ketakutan adalah karena kita tahu ada harapan
(Kisuke Urahara - Bleach)

"Float like a Cadillac, sting like a Beemer."
(Lightning McQueen – Cars)

“Stay Courious”
(Xaviera Putri)

MOTTO

Figur terindah, pelengkap hidupku
Tanpa pamrih peluhmu yang takkan terdup
Membuat jiwa halus dengan cinta
Kasih, air mata, amarah yang sempurna
Tak patut ku pertanyakan di dalam batin
Mengapa pikiran sadar ini tak mau bergeming
Kenapa selalu kulakukan hal yang buntu
Yang selalu bertentangan dengan imajimu
Cinta yang kau punya terlalu istimewa
Tak mampu kubayar dengan bait kata bermakna
Mama, maafkan You know i love you
Mama, kaulah cinta abadiku
Kaulah cermin jiwaku, pelita hidupku
Mama, terima kasih untukmu, atas cinta yang tak henti
Mengalir, mengalun, dan pasti tak akan mati
(Pelita Hidup – Bondan Prakoso F2B)

HALAMAN PERSEMBAHAN

Atas kehendak dan karunia Allah SWT, skripsi ini penulis persembahkan kepada :

Inti Dian Lisnawati

Orang Tua Penulis

Dan

Almamater Tercinta

Program Studi Informatika

Fakultas Sains dan Teknologi

UIN Sunan Kalijaga Yogyakarta

KATA PENGANTAR

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

الْحَمْدُ لِلَّهِ رَبِّ الْعَالَمِينَ وَالصَّلَاةُ وَالسَّلَامُ عَلَى سَيِّدِنَا مُحَمَّدٍ وَعَلَى آلِهِ وَصَحْبِهِ أَجْمَعِينَ

اللَّهُمَّ ارْزُقْنَا فَهْمَ النَّبِيِّينَ وَحِفْظَ الْمُرْسَلِينَ وَإِلْهَامَ الْمَلَائِكَةِ الْمُقَرَّبِينَ، وَارْحَمْنَا بِرَحْمَتِكَ يَا أَرْحَمَ الرَّاحِمِينَ،

آمِينَ يَا مُجِيبَ السَّائِلِينَ

Tiada kata yang patut diucapkan selain puja dan puji syukur kepada Allah SWT, Yang Maha Mutlak, Maha Rahman dan Rahim, Sang penguasa Alam Semesta, yang berkehendak atas segala sesuatu. Karena dengan izin-Nya penyusunan skripsi yang berjudul **"Rancang Bangun Ekstensi Visual Studio Code untuk Otomatisasi Dokumentasi, Pengujian, dan Analisis Keamanan pada Application Programming Interface (API)"** ini dapat terselesaikan. Shalawat dan salam semoga tetap tercurahkan pada manusia sempurna, Nur Muhammadiyah, yang karenanya alam ini diciptakan, dan karenanya pula perdamaian dan kesejahteraan dunia tercapai.

Penulisan Skripsi ini ini disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana pada Program Studi Informatika, Fakultas Sains dan Teknologi, UIN Sunan Kalijaga Yogyakarta. Skripsi ini adalah hasil dari tulisan seseorang yang belum sempurna dalam segala hal, maka tentunya banyak kekurangan dan kesalahan dalam penulisan ini. Untuk itu, kritik dan saran serta nasehat-nasehat dari pembaca sangat diharapkan demi perbaikan dan kesempurnaan karya ini. Tiada sesuatupun yang dapat terselesaikan tanpa bantuan orang lain. Begitu pula karya ini. Oleh karena itu, penulis menyampaikan rasa terima kasih yang sedalam-dalamnya kepada :

1. Bapak Prof. Noorhaidi Hasan, S.Ag., M.A., M.Phil., Ph.D. selaku Rektor Universitas Islam Negeri Sunan Kalijaga Yogyakarta.
2. Ibu Prof. Dr. Dra. Hj. Khurul Wardati, M.Si. selaku Dekan Fakultas Sains dan Teknologi.

3. Bapak Dr. Muhammad Mustakim, S.T., M.T. sebagai Ketua Program Studi Informatika yang telah memberikan dukungan selama masa perkuliahan.
4. Bapak Muhammad Didik Rohmad Wahyudi, S.T., MT. sebagai dosen pembimbing akademik sekaligus dosen pembimbing skripsi penulis yang selalu membimbing dan mengarahkan selama menjalani perkuliahan.
5. Bapak Rio Nurtantyana, S.Pd., M.Pd., M.Sc., Ph.D. sebagai dosen pembimbing skripsi eksternal (BRIN) yang telah memberikan dukungan luar biasa, meluangkan waktu, bimbingan serta motivasi untuk menyelesaikan skripsi ini.
6. Badan Riset Inovasi Nasional (BRIN) khususnya Pusat Riset Data Sains dan Informasi (PRSDI) sebagai institusi yang telah memberikan kesempatan emas bagi penulis untuk melaksanakan penelitian. Terima kasih atas ruang, waktu, serta pengetahuan baru yang luar biasa sehingga penulis dapat menjalankan seluruh tahapan penelitian ini dengan lancar dan terarah.
7. Ibunda tercinta beserta kedua garis keluarga besar baik dari Pakis dan Blabak, yang selalu memberikan dukungan dan kekuatan doa saktinya, Terima kasih atas dukungan penuh baik berupa moril ataupun materil dalam menyelesaikan studi di UIN Sunan Kalijaga Yogyakarta, maka sudah selakyaknya tulisan sederhana ini penulis persembahkan kepada mereka.
8. Abah K.H Irfa'i Nachrawi sebagai orang tua rohani penulis, beserta putra putra guru yang membimbing jasad dan jiwa penulis dalam setiap langkah langkah kehidupan. Sudah selakyaknya juga tulisan ini penulis persembahkan kepada beliau.
9. Segenap keluarga besar Majlis Tarbiyah wa ta'lim Rubath Mubarak Qashrul 'Arifin Yogyakarta
10. Rekan - Rekan mahasiswa Informatika, yang telah menjadi teman seperjuangan selama masa studi, berbagi ilmu, pengalaman, serta semangat dalam menghadapi berbagai tantangan akademik.

11. Rekan perjuangan yang telah menyambung asa penulis untuk menjadi tangan, kaki dan pikirian penulis sehingga tercurahkan coretan yang tersusun.
12. Bumi Raya yang telah menjadi representasi dari isyarah keindahan yang menyimpan makna makna ayat ayat afaqi dan petunjuk intervensi Tuhan terhadap penulis.
13. FC Barcelona sebagai klub kebanggan penulis yang senantiasa mengajarkan bagian kata dari makna sabar, dan tabah dalam mencapai suatu arah untuk terus melangkah.
14. Seluruh Playslist Musik kehidupan yang sering penulis putar untuk membangkitkan semangat untuk terus melangkah kedepan, terimakasih atas karya karya yang abadi.
15. Seluruh pihak yang tidak dapat penulis sebutkan satu per satu dalam lembaran ini, terima kasih atas segala bentuk bantuan, doa, dan dukungan yang telah diberikan.
16. Terima kasih untuk penulis. Terima kasih karena tidak menyerah meski harus melewati berbagai tantangan yang menguras energi dan emosi.

Penulis berharap semoga Allah SWT senantiasa melimpahkan rahmat dan hidayah-Nya kepada semua pihak yang telah membantu. Semoga karya tulis ini dapat memberikan manfaat bagi kita semua.

Yogyakarta, 01 Maret 2026

Penulis,

Muhammad Afrizal

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam ekosistem pengembangan perangkat lunak modern, *Application Programming Interface (API)* telah bertransformasi menjadi tulang punggung infrastruktur teknologi global. Tidak lagi sekadar berfungsi sebagai alat pertukaran data teknis, API kini memegang peranan sentral dalam memastikan integrasi sistem dan skalabilitas. Menurut riset oleh Tupe dan Thube [1], API membentuk fondasi ekosistem digital modern yang memungkinkan komunikasi data antara berbagai sistem, aplikasi, dan layanan yang beragam. Keberadaan API menjadi persyaratan mutlak untuk menjamin interoperabilitas, yakni menjembatani *sistem legacy* (warisan) dengan aplikasi modern, serta menjaga kontinuitas aliran data di tengah arsitektur perangkat lunak pada perusahaan yang terus berkembang.

Seiring dengan meningkatnya adopsi arsitektur berbasis *microservices* dan integrasi platfrom *cloud*, kompleksitas ekosistem perangkat lunak secara cepat akan meningkat drastis. Pada penelitian Dhyani et al. [2] menekankan bahwa API memungkinkan komponen perangkat lunak yang berbeda untuk berkomunikasi satu sama lain dengan mudah, sebuah fungsi yang sangat krusial untuk mendukung kegunaan *(usability)* dan pemeliharaan arsitektur yang kompleks. Oleh karena itu, dalam ekosistem yang serba terhubung ini, API tidak hanya dipandang sebagai kode program semata, melainkan sebagai aset strategis yang memfasilitasi inovasi dan skalabilitas dalam pengembangan aplikasi skala besar.

Meskipun peran API sangat strategis dalam bidang pengembangan perangkat lunak, realitas operasional dalam siklus hidup pengembangannya sering kali terhambat oleh inefisiensi alur kerja. Tantangan utama yang dihadapi pengembang *(developer)* saat ini adalah banyak nya aplikasi yang harus di buka saat melakukan produktivitas pengembangan aplikasi, di mana proses pengembangan, dokumentasi, dan pengujian tersebar di berbagai aplikasi yang terpisah. Kondisi ini mengakibatkan pengembang dipaksa untuk melakukan perpindahan konteks *(context switching)* secara frekuentif yang terbukti secara empiris menurunkan konsentrasi dan produktivitas secara signifikan. Silva et al. [3] dalam penelitiannya

menemukan bahwa *context switching* merupakan salah satu faktor personal utama yang menghambat konsentrasi dan secara drastis menurunkan produktivitas pengembang akibat beban kognitif *(cognitive overload)* yang berlebihan. Hal ini sejalan dengan temuan Pettersson [4], yang menekankan bahwa friksi akibat aktivitas paralel dan penggunaan *software/tools* yang tidak terintegrasi menciptakan hambatan alur kerja yang serius. Dalam praktiknya, pengembang *(developer)* sering kali harus meninggalkan lingkungan pengembangan (*software IDE*) hanya untuk memperbarui dokumentasi atau menjalankan pengujian di aplikasi terpisah, sebuah proses repetitif yang menurut Coelho et al. [5] sering dipersepsikan sebagai aktivitas yang tidak produktif dan rentan terhadap kesalahan manusia. Dari hal tersebut dapat di jelaskan bahwa beban kognitif akibat *information overload* dari penggunaan alat yang terpisah-pisah ini tidak hanya menghambat alur kerja, tetapi juga meningkatkan risiko kesalahan manusia *(human error)*. Ketika dokumentasi dan kode dikelola secara terpisah tanpa integrasi yang memadai, pengembang rentan mengalami keterbatasan memori kerja *(working memory)* yang berujung pada munculnya bug atau celah keamanan dalam arsitektur sistem. Oleh karena itu, ketergantungan pada alat manual yang tidak terintegrasi kini menjadi salah satu penghambat dalam pengembangan perangkat lunak.

Dampak dari fragmentasi alat ini tidak hanya berhenti pada penurunan produktivitas pengembang, tetapi juga berimbas langsung pada kualitas akhir perangkat lunak dan keamanan sistem. Ketergantungan pada proses manual dan terpisah sering kali menyebabkan fenomena *'delayed defect discovery'* (keterlambatan penemuan cacat). Strategi pengujian tradisional di mana validasi dilakukan terpisah setelah fase penulisan kode sering kali gagal mengimbangi kecepatan alur kerja Agile modern. Akibatnya, celah keamanan dan kesalahan logika baru terdeteksi di fase akhir, yang secara drastis meningkatkan beban kerja ulang *(rework)* dan biaya perbaikan. Oleh karena itu, diperlukan pergeseran paradigma menuju pendekatan *Shift-Left*, di mana validasi dokumentasi, fungsionalitas, dan keamanan diintegrasikan sedini mungkin ke dalam lingkungan pengembangan (IDE) untuk memfasilitasi mitigasi celah keamanan lebih awal dan mencegah hambatan teknis. [6]

Meskipun urgensi efisiensi pengembangan telah banyak dibahas, terdapat celah penelitian (*research gap*) yang signifikan terkait integrasi alat pendukung dalam satu lingkungan kerja. Pada ekosistem pengembangan web modern berbasis Next.js. Svedas [7] menyoroti bahwa Next.js telah berevolusi menjadi kerangka kerja *full-stack* yang dominan, memungkinkan pengembang membangun antarmuka pengguna sekaligus menangani logika *backend* melalui fitur API Routes dan *Server-Side Rendering* (SSR). Namun, integrasi kemampuan *backend* ini membawa tantangan baru, yaitu berupa alat dokumentasi dan pengujian API konvensional yang tidak mengenali struktur direktori *file-system routing* yang digunakan oleh Next.js. Dhyani et al. [8] menyoroti bahwa metode pembuatan dokumentasi API saat ini cenderung stagnan dan masih sangat bergantung pada proses manual yang rentan kesalahan, hal ini menjadi sangat tidak efisien dalam arsitektur Next.js yang dinamis. Di sisi lain, Punz [9] dalam penelitian nya menekankan bahwa ketiadaan alat bantu yang terdedikasi *(dedicated tooling)* membuat proses pengembangan menjadi sulit dan tidak efisien. Punz membuktikan bahwa menyediakan umpan balik waktu nyata (*real-time feedback*) langsung di dalam editor kode adalah kunci untuk menurunkan hambatan teknis dan mendeteksi kesalahan lebih awal. Namun, solusi yang ada saat ini masih cenderung terfragmentasi, belum banyak penelitian yang menawarkan pendekatan holistik yang menyatukan otomatisasi dokumentasi, pengujian fungsional, dan pemindaian keamanan dalam satu ekstensi terintegrasi.

Merespons tantangan fragmentasi dan kebutuhan akan efisiensi tersebut, penelitian ini mengusulkan rancang bangun sebuah ekstensi pada *Visual Studio Code* (VS Code) yang dirancang khusus untuk ekosistem API. Pemilihan VS Code didasarkan pada posisinya sebagai *Integrated Development Environment* (IDE) yang paling mendominasi di kalangan pengembang web modern; tercatat lebih dari 73% pengembang perangkat lunak secara global menggunakannya sebagai editor utama mereka [10]. Di sisi lain, meskipun perangkat lunak pengujian dan dokumentasi API populer seperti Postman, Insomnia, maupun Swagger telah tersedia secara luas, alat-alat tersebut beroperasi sebagai aplikasi pihak ketiga yang mandiri (*standalone applications*). Ketiadaan integrasi langsung antara alat pengujian tersebut dengan IDE memaksa pengembang untuk terus-menerus

melakukan *context switching* saat berpindah antara menulis kode dan mengujinya, sebuah proses yang terbukti secara empiris memecah fokus dan menurunkan produktivitas [3]. Oleh karena itu, pendekatan pembuatan ekstensi ini sejalan dengan temuan Marklund [11] yang menyimpulkan bahwa pengembangan ekstensi kustom secara langsung di dalam editor mampu mengoptimalkan alur kerja pengembang secara signifikan. Melalui ekstensi ini, proses dokumentasi, pengujian, dan analisis keamanan yang sebelumnya tersebar di berbagai aplikasi akan diotomatisasi dan dipusatkan ke dalam satu antarmuka (IDE). Solusi terpadu ini diharapkan tidak hanya meminimalisir *context switching* dan *human error*, tetapi juga mempercepat siklus pengiriman perangkat lunak dengan kualitas dan keamanan yang lebih terjamin.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dipaparkan, dapat dirumuskan permasalahan penelitian sebagai berikut :

1. Bagaimana merancang dan membangun ekstensi Visual Studio Code yang mampu mengintegrasikan pembuatan dokumentasi otomatis dan pengujian API ke dalam satu lingkungan kerja (IDE) guna meminimalkan *context switching* pada pengembangan aplikasi Next.js?
2. Bagaimana mengintegrasikan fitur deteksi celah keamanan dan masalah skalabilitas dini (*Shift-Left Testing*) menggunakan bantuan AI, serta bagaimana proses distribusinya (*deployment*) agar ekstensi dapat diakses secara global?
3. Bagaimana tingkat kelayakan perangkat lunak secara teknis serta tingkat penerimaan penggunanya (*user acceptance*) setelah dievaluasi melalui serangkaian pengujian perangkat lunak yang komprehensif?

1.3 Batasan Masalah

Agar arah penelitian tetap konsisten dan mencapai sasaran yang diharapkan, peneliti merasa perlu untuk membatasi ruang lingkup kajian. Berikut batasan masalah dalam penelitian ini :

1. Penelitian ini hanya berfokus pada pengembangan ekstensi untuk Visual Studio Code (VS Code)
2. Ekstensi dikhususkan untuk memindai proyek berbasis Next.js (khususnya fitur *API Routes*)
3. Fitur pengujian terbatas pada validasi *endpoint* (HTTP Methods: GET, POST, PUT, DELETE) untuk memastikan respons status kode yang sesuai.
4. Analisis keamanan yang dilakukan bersifat *Static Application Security Testing* (SAST) dasar, yang berfokus pada deteksi kerentanan umum seperti eksposur *sensitive data (API Keys)* dan ketiadaan *security headers*

1.4 Tujuan Penelitian

Berangkat dari latarbelakang diatas, maksud dan tujuan dari penelitian ini mencakup beberapa hal, sebagai berikut :

1. Menghasilkan rancang bangun perangkat lunak berupa ekstensi Visual Studio Code yang menyatukan fitur dokumentasi dan pengujian API ke dalam satu lingkungan kerja untuk meminimalkan *context switching* pada kerangka kerja Next.js.
2. Mengimplementasikan fitur kecerdasan buatan (AI) untuk mendeteksi celah keamanan sedini mungkin (*Shift-Left Testing*) pada fase penulisan kode, serta mendistribusikan (*deployment*) ekstensi tersebut secara publik.
3. Mengukur dan mengevaluasi kualitas, kelayakan fungsional, serta tingkat penerimaan pengguna terhadap ekstensi yang dibangun melalui pengujian perangkat lunak yang holistik, guna memastikan alat bantu ini beroperasi secara optimal dan efektif.

1.5 Manfaat Penelitian

Adanya Penelitian ini diharapkan mampu untuk memberikan gambaran manfaat baik secara teoretis dalam memperkaya keilmuan riset, maupun secara praktis sebagai bahan pertimbangan dalam pengambilan keputusan bagi pihak-pihak terkait di dalamnya.

1.5.1 Manfaat Teoritis

Penelitian ini diharapkan dapat memberikan manfaat teoritis untuk beberapa pihak, diantaranya :

1. Memberikan kontribusi pada pengembangan ilmu di bidang *Software Engineering,* khususnya penerapan Ekstensi VS Code untuk Otomatisasi Dokumentasi, Pengujian, dan Analisis Keamanan pada Application Programming Interface (API)
2. Memperkaya literatur mengenai implementasi paradigma *Shift-Left Testing* di tingkat IDE, dalam hal ini memberikan bukti empiris baru bahwa pergeseran proses validasi (dokumentasi dan keamanan) dari fase pengembangan (Visual Studio Code) dapat secara efektif mengurangi fragmentasi alur kerja dan *context switching*.

1.5.2 Manfaat Praktis

Penelitian ini diharapkan dapat memberikan manfaat praktis untuk beberapa pihak, diantaranya:

1. Bagi Pengembang Perangkat Lunak (*Developer*), menyediakan alat bantu kerja yang mampu memangkas waktu administrasi teknis secara signifikan melalui otomatisasi dokumentasi dan pengujian langsung di dalam editor (VS Code). Hal ini memungkinkan pengembang untuk lebih fokus pada logika bisnis, mengurangi beban kognitif akibat *context switching*, serta menghindari kesalahan penulisan dokumentasi manual pada proyek berbasis Next.js.
2. Bagi Tim Teknis dan Industri, menghadirkan solusi efisiensi biaya dan penjaminan kualitas (*Quality Assurance*) dini melalui penerapan konsep *Shift-Left Testing*. Dengan adanya fitur pemindaian keamanan yang

berjalan *real-time*, tim dapat mendeteksi celah keamanan dan inkonsistensi API sejak fase penulisan kode, sehingga mencegah akumulasi kesalahan teknis dan mempermudah proses integrasi sistem di kemudian hari.

BAB V
KESIMPULAN DAN PENUTUP

5.1 Simpulan

Berdasarkan hasil penelitian yang telah dilakukan, berikut merupakan simpulan yang diperoleh dari penelitian Rancang Bangun Ekstensi Visual Studio Code untuk Otomatisasi Dokumentasi, Pengujian, dan Analisis Keamanan Application Programming Interface (API) :

1. Urgensi Pengembangan Ekstensi API :
   Ekstensi Visual Studio Code untuk pengembangan API (khususnya pada *framework* Next.js) masih sangat dibutuhkan untuk menyelesaikan permasalahan fragmentasi alat kerja (*tool fragmentation*) dan tingginya intensitas perpindahan konteks (*context switching*). Ketergantungan pada aplikasi pengujian eksternal sering kali menyebabkan beban kognitif yang berlebih, ketidaksinkronan dokumentasi, hingga keterlambatan deteksi celah keamanan. Kehadiran ekstensi yang mengintegrasikan otomatisasi dokumentasi dan pengujian langsung di dalam *Integrated Development Environment* (IDE), dalam hal ini *Visual Studio Code* terbukti menjadi solusi krusial untuk mengeliminasi inefisiensi tersebut melalui pendekatan *Shift-Left Testing*
2. Keberhasilan Distribusi (*Deployment*) :
   Penelitian ini telah berhasil mencapai tahap pengembangan akhir dengan mendistribusikan perangkat lunak secara langsung ke lingkungan produksi (*production deployment*). Ekstensi *"NextJS API Inspector"* telah lolos verifikasi keamanan dan berhasil dipublikasikan secara global di platform resmi *Visual Studio Code Marketplace*, selain itu ekstensi ini juga dapat di install menggunakan paket vsix yang ada dalam public repository girthub. Hal ini membuktikan bahwa perangkat lunak telah memenuhi standar distribusi

publik dan dapat diinstal dengan mudah oleh pengembang luas tanpa memerlukan konfigurasi prasyarat yang rumit

3. Kualitas Perangkat Lunak Terstandarisasi (ISO/IEC 25010) :

   Evaluasi kelayakan perangkat lunak telah dilakukan secara holistik menggunakan model kualitas internasional ISO/IEC 25010:2023. Penelitian ini berhasil mengadaptasi dan menguji keseluruhan 9 karakteristik kualitas utama (*Functional Suitability, Performance Efficiency, Compatibility, Interaction Capability, Reliability, Security, Maintainability, Flexibility,* dan *Safety*). Secara keseluruhan, hasil pengujian menunjukkan kualitas yang sangat unggul di kesembilan aspek tersebut, dibuktikan dengan tingkat keberhasilan fungsional dan kompatibilitas 100%, skor performa 100/100 pada instrumen pengukuran, hingga perolehan kualitas kode *Rating A* pada SonarQube. Secara menyeluruh, sistem terbukti sangat stabil, terstandarisasi, dan terbebas dari anomali fungsional

4. Penerimaan Pengguna (*User Perspective*) :

   Berdasarkan evaluasi *User Acceptance* yang melibatkan 30 responden dari kalangan akademisi maupun praktisi profesional IT, ekstensi ini berhasil memperoleh skor rata-rata *System Usability Scale* (SUS) sebesar 78,75. Pencapaian metrik tersebut menempatkan sistem pada kategori *Acceptable* (Dapat Diterima) dengan predikat *Good* (Baik), yang sekaligus menjadi bukti empiris atas tingkat kepuasan interaksi pengguna yang sangat tinggi. Pengguna mengonfirmasi bahwa fitur yang ditawarkan khususnya kemampuan otomatisasi dari asisten kecerdasan buatan (Gemini Flash 2.5) terbilang efektif dalam memangkas repetisi alur kerja manual (*Alt+Tab*), mencegah ketidaksinkronan kode (*documentation drift*), serta memfasilitasi mitigasi celah keamanan lebih awal. Secara keseluruhan, ekstensi ini dinilai berhasil menghadirkan pengalaman produktivitas yang jauh lebih fokus, modern, dan efisien bagi pengembang perangkat lunak.

5.2 Saran

Berdasarkan hasil penelitian dan simpulan yang diperoleh, berikut merupakan saran untuk pengembangan lebih lanjut.

1. Ekspansi Dukungan *Framework* Pengembangan :

   Penelitian dan pengembangan ekstensi saat ini masih difokuskan dan untuk ekosistem *framework* Next.js. Untuk pengembangan selanjutnya, kapabilitas *parser* algoritma deteksi rute disarankan untuk dilebarkan agar dapat mendukung kerangka kerja (*framework*) *backend* populer lainnya, seperti NestJS, Laravel, maupun Go Fiber. Perluasan dukungan ini akan secara signifikan meningkatkan utilitas dan jangkauan alat bagi komunitas pengembang perangkat lunak yang lebih luas.
2. Studi Komparatif *Large Language Model* (LLM) :

   Implementasi kecerdasan buatan pada sistem ini baru memanfaatkan satu ekosistem *Large Language Model* (LLM), yaitu Google Gemini, dan belum membandingkannya dengan model bahasa lain. Oleh karena itu, disarankan bagi penelitian mendatang untuk mengintegrasikan dan melakukan studi komparatif dengan berbagai penyedia LLM lainnya, seperti OpenAI (ChatGPT), Anthropic (Claude) atau lainnya. Evaluasi komparatif ini sangat penting untuk menemukan keseimbangan yang paling optimal antara tingkat akurasi analisis kode, efisiensi konsumsi token, dan latensi respons dari masing-masing model.
3. Adaptasi Lintas Lingkungan Pengembangan (IDE) :

   Arsitektur sistem yang dibangun masih terbatas pada lingkungan *Integrated Development Environment* (IDE) Visual Studio Code. Mengingat tren produktivitas pengembang yang terus berevolusi, disarankan agar fitur dan fungsionalitas ekstensi ini diadaptasi agar dapat beroperasi pada platform IDE modern lainnya. Integrasi pada editor kode berbasis AI seperti Cursor, Antigravity, maupun ekosistem JetBrains akan memberikan fleksibilitas tinggi dan memastikan bahwa inovasi *Shift-Left Testing* ini dapat dinikmati oleh pengembang dengan berbagai preferensi lingkungan kerja

4. Optimalisasi Skalabilitas pada Proyek Berskala Besar :

   Pengujian ekstensi saat ini masih difokuskan pada lingkup proyek berskala menengah. Pada skenario implementasi tingkat industri (*enterprise*) yang memiliki arsitektur monolitik atau ratusan *endpoint* API, proses pemindaian (*scanning*) dan generasi dokumen secara serentak dapat membebani kinerja editor. Oleh karena itu, penelitian selanjutnya disarankan untuk mengembangkan mekanisme *incremental parsing* (hanya memindai rute yang mengalami perubahan) dan sistem *caching* yang lebih mutakhir. Strategi ini sangat penting agar sistem dapat menangani proyek berskala besar tanpa memicu *timeout* pada API *Large Language Model* (LLM)

5. Standardisasi Gaya Penulisan Kode dalam Kolaborasi Tim :

   Mengingat proyek perangkat lunak umumnya dikerjakan secara kolaboratif, setiap pengembang sering kali memiliki kebiasaan atau gaya penulisan kode (*coding style*) yang bervariasi. Untuk menjaga konsistensi, ekstensi di masa depan disarankan memiliki kapabilitas untuk menyelaraskan keluaran AI dengan standar konvensi tim. Hal ini dapat diwujudkan dengan membuat ekstensi mampu membaca berkas konfigurasi *linter* lokal (seperti ESLint atau Prettier) atau memungkinkan tim mendefinisikan pedoman khusus (*custom prompt instructions*) di tingkat *workspace*.

6. Mitigasi dan Perhitungan Dampak Performa pada Skenario Kolaboratif :

   Pada lingkungan kerja tim yang sangat aktif, eksekusi analisis AI yang dilakukan secara terus-menerus dan bersamaan oleh banyak anggota tim dapat memicu lonjakan permintaan jaringan (*network congestion*) atau melampaui batas kuota API (*rate-limiting*). Penelitian selanjutnya perlu merumuskan perhitungan metrik beban (*load metrics*) dan strategi mitigasi performa yang presisi. Pendekatan yang dapat diteliti meliputi implementasi sistem antrean (*request queueing/throttling*) di dalam ekstensi, atau pemanfaatan basis data lokal berukuran ringan (*local vector database*) di dalam *repository*. Fitur ini memungkinkan hasil analisis dari satu pengembang disimpan dan dibagikan secara lokal (*shared context*), sehingga anggota tim lain tidak perlu melakukan pemanggilan API (AI *request*) yang repetitif untuk berkas yang belum berubah.

DAFTAR PUSTAKA

[1] V. Tupe and S. Thube, "AI Agentic workflows and Enterprise APIs: Adapting API architectures for the age of AI agents," *arXiv preprint arXiv:2502.17443*, 2025.

[2] P. Dhyani, S. Nautiyal, A. Negi, S. Dhyani, and P. Chaudhary, "Automated API Docs Generator using Generative AI," in *2024 IEEE International Students' Conference on Electrical, Electronics and Computer Science, SCEECS 2024*, Institute of Electrical and Electronics Engineers Inc., 2024. doi: 10.1109/SCEECS61402.2024.10482119.

[3] J. C. A. Silva *et al.*, "'Frustrating, Stressful, and Overwhelming': Insights into Software Practitioners' Productivity from Stack Exchange Discussions," in *Simpósio Brasileiro de Engenharia de Software (SBES)*, SBC, 2025, pp. 237–248.

[4] M. Pettersson, "Developer Productivity: Exploring factors that affect developer productivity and how they impact team performance and prioritization," 2025.

[5] M. Coelho, I. Reinbold, L. Sancho, M. Paixao, A. A. Araújo, and S. Freire, "Software Developers' Perceptions of Productivity: An Industry-focused Study," in *Simpósio Brasileiro de Qualidade de Software (SBQS)*, SBC, 2025, pp. 12–22.

[6] C. S. Pareek, "Driving Agile Excellence in Insurance Development through Shift-Left Testing," 2025. [Online]. Available: www.ijfmr.com

[7] E. Svedas, "Building an application using Next. js," 2024.

[8] P. Dhyani, S. Nautiyal, A. Negi, S. Dhyani, and P. Chaudhary, "Automated API docs generator using generative AI," in *2024 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*, IEEE, 2024, pp. 1–6.

[9] F. Punz, "MicroJava Extension for Visual Studio Code," 2025.

[10] Stack Overflow, "2023 Developer Survey." Accessed: Feb. 23, 2026. [Online]. Available: https://survey.stackoverflow.co/2023/

[11] A. Marklund, "Visual Studio Code Extension study and integration with Varvault: S0003D," 2024.

[12] A. Smardas and K. Kritikos, "Towards the Automatic Production of OpenAPI Specifications from Source Code," in *2025 12th International Conference on Future Internet of Things and Cloud (FiCloud)*, IEEE, 2025, pp. 342–349.

[13] S. Lee, J. Heo, and K. R. Dearstyne, “Can Llms Update Api Documentation?,” in *2025 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2025, pp. 1–12. doi: 10.1109/ICSME64153.2025.00048.

[14] N. Kamal, “API Documentation Generator,” 2022.

[15] A. Ali, H. A. Maghawry, and N. Badr, “AUTOMATION OF PERFORMANCE TESTING: A REVIEW.,” *International Journal of Intelligent Computing & Information Sciences*, vol. 22, no. 4, 2022.

[16] H. Anderstedt and M. Wifvesson, “Benchmarking and Load Testing a Dynamic CRM Architecture,” *LU-CS/HBG-EX*, 2025.

[17] D. B. Cruz, J. R. Almeida, and J. L. Oliveira, “Open source solutions for vulnerability assessment: A comparative analysis,” *IEEE Access*, vol. 11, pp. 100234–100255, 2023.

[18] E. Alor, S. Khatoonabadi, and E. Shihab, “Evaluating the Use of LLMs for Documentation to Code Traceability,” *arXiv preprint arXiv:2506.16440*, 2025.

[19] K. Lazar *et al.*, “SpeCrawler: Generating OpenAPI Specifications from API Documentation Using Large Language Models,” *arXiv preprint arXiv:2402.11625*, 2024.

[20] E. Lyster Golawski and J. Taylor, “Code Generation from Large API Specifications with Open Large Language Models: Increasing Relevance of Code Output in Initial Autonomic Code Generation from Large API Specifications with Open Large Language Models,” 2024.

[21] F. Punz, “MicroJava Extension for Visual Studio Code,” 2025.

[22] A. Marklund, “Visual Studio Code Extension study and integration with Varvault: S0003D,” 2024.

[23] S. B. Nettur, S. Karpurapu, U. Nettur, and L. S. Gajja, “Cypress copilot: Development of an AI assistant for boosting productivity and transforming web application testing,” *IEEE Access*, 2024.

[24] H. Ho, “Developing a full-stack E-commerce application with Next. js, JavaScript, React and MongoDB,” 2024.

[25] T. Kettunen, “REACT SERVER COMPONENTS RENDERING PATTERNS IN NEXT. JS,” 2024.

[26] Y. Mulyanto, E. Haryanti, and L. Lazarus, “IMPLEMENTASI REACT SERVER COMPONENT DAN SERVER ACTION UNTUK MENINGKATKAN PERFORMA APLIKASI WEB,” *TEKNIMEDIA: Teknologi Informasi dan Multimedia*, vol. 5, no. 1, pp. 17–24, 2024.

[27] M. Helenius and M. Vallius, “REST API SECURITY: TESTING AND ANALYSIS,” 2022.

[28] B. Marii and I. Zholubak, “Features of development and analysis of rest systems,” *Advances in Cyber-Physical Systems*, vol. 7, no. 2, pp. 121–129, 2022.

[29] P. Gowda and A. N. Gowda, “Best Practices in REST API Design for Enhanced Scalability and Security,” *Journal of Artificial Intelligence, Machine Learning and Data Science*, vol. 2, no. 1, pp. 827–830, 2024.

[30] S. Casas, D. Cruz, G. Vidal, and M. Constanzo, “Uses and applications of the OpenAPI/Swagger specification: a systematic mapping of the literature,” in *2021 40th International Conference of the Chilean Computer Science Society (SCCC)*, IEEE, 2021, pp. 1–8.

[31] T. H. TRAN NGUYEN, “Developing An End-To-End E-Commerce Web Solution With Next. js,” 2023.

[32] R. Kumar, “Standardizing API Contracts: Enabling Interoperability in Distributed Systems,” *ecosystems*, vol. 5, p. 19, 2022.

[33] F. Jansson, “Designing to Optimise Usability of API Documentation Interfaces: Applying a User-Centric Approach to Enhance Developer Experience,” 2024, *KTH Royal Institute of Technology*.

[34] J. Y. Khan and G. Uddin, “Automatic code documentation generation using gpt-3,” in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, 2022, pp. 1–6.

[35] W. Muhamad, Suhardi, and Y. Bandung, “Transforming OpenAPI Specification 3.0 documents into RDF-based semantic web services,” *J. Big Data*, vol. 9, no. 1, p. 55, 2022.

[36] P. L. Kaluarachchi, D. V Wadasinghe, E. T. M. Ranaweera, W. M. M. S. Weerasooriya, D. I. De Silva, and J. V. A. Amarasinghe, “A Comparative Analysis of Unit Testing and Integration Testing Based on Adding a New Feature in an E-commerce Application”.

[37] A. Godinho, J. Rosado, F. Sa, and F. Cardoso, “Performance Comparison of RESTful Web APIs using a Test Suite: .NET vs. Java Spring Boot,” *Journal of Software and Systems Development*, pp. 1–32, 2024.

[38] M. M. Olmez and E. Gehringer, “Automation of Test Skeletons Within Test-Driven Development Projects,” in *2024 36th International Conference on Software Engineering Education and Training (CSEE&T)*, IEEE, 2024, pp. 1–10.

[39] S. D. Sri, R. C. S. P. Raman, G. Rajagopal, and S. T. Chan, “Automating rest api postman test cases using llm,” *arXiv preprint arXiv:2404.10678*, 2024.

[40] D. Dalaq, K. F. Daya, A. Dalaq, M. N. Arefin, and M. K. Niazi, "A Systematic Literature Review on Static Application Security Testing (SAST) Tools: Evaluation, Benchmarks, Challenges, and Future Directions," in *Proceedings of the 2025 29th International Conference on Evaluation and Assessment in Software Engineering Companion*, 2025, pp. 162–168.

[41] L. Dencheva, "Comparative analysis of Static application security testing (SAST) and Dynamic application security testing (DAST) by using open-source web application penetration testing tools," Dublin, National College of Ireland, 2022.

[42] L. Moisei and L. Mocanu, "Mitigating OWASP API security top 5 risks through API gateway patterns," in *International Conference on Machine Intelligence & Security for Smart Cities (TRUST) Proceedings*, 2025, pp. 29–36.

[43] K. Reddy, "API Security and Authentication Mechanisms in Cloud Applications," 2026.

[44] V. S. Rani, A. R. Babu, K. Deepthi, and V. R. Reddy, "Shift-left testing in devops: A study of benefits, challenges, and best practices," in *2023 2nd International Conference on Automation, Computing and Renewable Systems (ICACRS)*, IEEE, 2023, pp. 1675–1680.

[45] S. A. Vaddadi, R. Thatikonda, A. Padthe, and P. R. R. Arnepalli, "Shift left testing paradigm process implementation for quality of software based on fuzzy," *Soft comput.*, pp. 1–13, 2023.

[46] S. Piciorea and T. Nguyen, "Bringing Invariant Analysis to modern IDEs: The DIG+ Extension for VS Code," in *Proceedings of the 34th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2025, pp. 51–55.

[47] L. Keating, "INTEGRATING SHIFT-LEFT AND SHIFT-RIGHT TESTING WITH AI-DRIVEN OBSERVABILITY IN DEVOPS PIPELINES," 2024.

[48] D. Nam, A. Macvean, V. Hellendoorn, B. Vasilescu, and B. Myers, "Using an llm to help with code understanding," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, 2024, pp. 1–13.

[49] A. Pande, R. Patil, R. Mukkemwar, R. Panchal, and S. Bhoite, "Comprehensive Study of Google Gemini and Text Generating Models: Understanding Capabilities and Performance," *Grenze International Journal of Engineering & Technology (GIJET)*, vol. 10, 2024.

[50] F. Punz, "MicroJava Extension for Visual Studio Code," 2025.

[51] L. Forstner, "Integrating GLSP based Tooling into Visual Studio Code," *Bachelor Thesis*, 2022.

[52] D. Bork and P. Langer, “Language server protocol: An introduction to the protocol, its use, and adoption for web modeling tools,” *Enterprise Modelling and Information Systems Architectures (EMISAJ)*, vol. 18, pp. 1–9, 2023.

[53] R. C. Richey and J. D. Klein, *Design and development research: Methods, strategies, and issues*. Routledge, 2014.

[54] N. Siagian, T. E. Tamba, H. H. O. Situmorang, and H. Samosir, “Aplikasi Apotek Berbasis Web Menggunakan Arsitektur Microservices (Studi Kasus Apotek Glen, Kab. Toba),” *Journal of Applied Technology and Informatics Indonesia*, vol. 1, no. 2, pp. 22–28, 2021.

[55] S. Nasiri, Y. Rhazali, M. Lahmer, and A. Adadi, “From user stories to UML diagrams driven by ontological and production model,” *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 6, 2021.

[56] P. Zhang, W. Dou, and H. Liu, “Hierarchical data structures for flowchart,” *Sci. Rep.*, vol. 13, no. 1, p. 5800, 2023.

[57] S. Al-Fedaghi, “Validation: Conceptual versus activity diagram approaches,” *arXiv preprint arXiv:2106.16160*, 2021.

[58] I. Joshi *et al.*, “Coprompter: User-centric evaluation of LLM instruction alignment for improved prompt engineering,” in *Proceedings of the 30th International Conference on Intelligent User Interfaces*, 2025, pp. 341–365.

[59] J. White, “A prompt pattern catalog to enhance prompt engineering with ChatGPT,” *arXiv preprint arXiv:2302.11382*, 2023.

[60] B. Mann *et al.*, “Language models are few-shot learners,” *arXiv preprint arXiv:2005.14165*, vol. 1, no. 3, p. 3, 2020.

[61] Z. Chen, M. M. Balan, and K. Brown, “Language models are few-shot learners for prognostic prediction,” *arXiv preprint arXiv:2302.12692*, 2023.

[62] M. Falco and G. Robiolo, “Building a catalogue of ISO/IEC 25010 quality measures applied in an industrial context,” in *Journal of Physics: Conference Series*, IOP Publishing, 2021, p. 012077.

[63] H. Panduwiyasa, M. Saputra, Z. F. Azzahra, and A. R. Aniko, “Accounting and smart system: functional evaluation of iso/iec 25010: 2011 quality model (a case study),” in *IOP Conference Series: Materials Science and Engineering*, IOP Publishing, 2021, p. 012065.

[64] iso25000.com, “ISO/IEC 25010.” Accessed: Feb. 09, 2026. [Online]. Available: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010

[65] S. A. Kinasih, R. Sahputra, and C. Anwar, "Functional Suitability Testing of Web-Based Warehouse Inventory Application Using Black Box Testing," *Ambidextrous Journal of Innovation Efficiency and Technology in Organization*, vol. 3, no. 02, pp. 138–154, 2025.

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

[67] C. Fricke, "Standalone web diagrams and lightweight plugins for web-IDEs such as Visual Studio Code and Theia," Bachelor's thesis, Kiel University, Department of Computer Science, 2021.

[68] T. Heričko, B. Šumak, and S. Brdnik, "Towards representative web performance measurements with google lighthouse," in *Proceedings of the 2021 7th student computer science research conference*, 2021.

[69] M. Nooraei Abadeh, "Genetic-based web regression testing: an ontology-based multi-objective evolutionary framework to auto-regression testing of web applications," *Service Oriented Computing and Applications*, vol. 15, no. 1, pp. 55–74, 2021.

[70] O. D. Segun-Falade, O. S. Osundare, W. E. Kedi, P. A. Okeleke, T. I. Ijomah, and O. Y. Abdul-Azeez, "Developing cross-platform software applications to enhance compatibility across devices and systems," *Computer Science & IT Research Journal*, vol. 5, no. 8, pp. 2040–2061, 2024.

[71] G. Aroush, "An Evaluation of Testing Frameworks for Beginners inJavaScript Programming: An evaluation of testing frameworks with beginners in mind," 2022.

[72] D. I. Carrión-León *et al.*, "Evaluating Interaction Capability in a Serious Game for Children with ASD: An Operability-Based Approach Aligned with ISO/IEC 25010: 2023," *Computers*, vol. 14, no. 9, p. 370, 2025.

[73] J. Brooke, "SUS-A quick and dirty usability scale," *Usability evaluation in industry*, vol. 189, no. 194, pp. 4–7, 1996.

[74] A. Bangor, P. T. Kortum, and J. T. Miller, "An empirical evaluation of the system usability scale," *Intl. Journal of Human–Computer Interaction*, vol. 24, no. 6, pp. 574–594, 2008.

[75] S. Kurniawan and A. Ependi, "USABILITY ANALYSIS OF C-ACCESS COMMUTERLINE APPLICATIONS USING THE SYSTEM USABILITY SCALE (SUS).," *Jurnal Syntax Admiration*, vol. 4, no. 7, 2023.

[76] A. Bangor, P. Kortum, and J. Miller, "Determining what individual SUS scores mean: Adding an adjective rating scale," *J. Usability Stud.*, vol. 4, no. 3, pp. 114–123, 2009.

[77] Sonar Community, “SonarQube Documentation: Sonarqube Comunity Build,” 2025, Accessed: Feb. 20, 2026. [Online]. Available: docs.sonarsource.com/sonarqube-community-build

[78] P. Bavadiya, “SONARQUBE-DRIVEN QUALITY GATES: IMPROVING SOFTWARE INTEGRITY THROUGH AUTOMATED CODE REVIEWS,” *International Journal of Applied Engineering & Technology*, vol. 6, no. 2, pp. 100–106, 2024.

[79] A. Abueshareik and A. Katbeh, “Empirical Evaluation of AI-Based Security Scanning andSonarQube in DevSecOps Pipelines,” 2025.

[80] R.-H. Pfeiffer and M. Lungu, “Technical Debt and Maintainability: How do tools measure it?,” *arXiv preprint arXiv:2202.13464*, 2022.

[81] J. Lenhard, S. Harrer, and G. Wirtz, “Measuring the installability of service orchestrations using the square method,” in *2013 IEEE 6th International Conference on Service-Oriented Computing and Applications*, IEEE, 2013, pp. 118–125.

[82] M. Wurster, U. Breitenbücher, O. Kopp, and F. Leymann, “Modeling and automated execution of application deployment tests,” in *2018 IEEE 22nd International Enterprise Distributed Object Computing Conference (EDOC)*, IEEE, 2018, pp. 171–180.

[83] S. Huang, “Load time optimization of JavaScript web applications,” 2019.

[84] M. A. Ali, N. K. Yap, A. A. A. Ghani, H. Zulzalil, N. I. Admodisastro, and A. A. Najafabadi, “A systematic mapping of quality models for AI systems, software and components,” *Applied Sciences*, vol. 12, no. 17, p. 8700, 2022.

[85] G. Oleksandr, G. Daria, R. Austen, G. Anatoliy, and T. Olga, “Quality Assessment of Artificial Intelligence Systems: A Metric-Based Approach,” *Electronics (Basel).*, vol. 15, no. 3, p. 691, 2026.

[86] Z. Xu, Z. Xie, S. Gandhi, and C. Kozyrakis, “FailSafe: High-performance Resilient Serving,” *arXiv preprint arXiv:2511.14116*, 2025.

[87] D. K. Zambou Zambou, “Large Language Models (LLMs) to Support Systematic Safety Assessments,” TU Clausthal, 2025.

[88] M. L. Siddiq, S. H. Majumder, M. R. Mim, S. Jajodia, and J. C. S. Santos, “An empirical study of code smells in transformer-based code generation techniques,” in *2022 IEEE 22nd International Working Conference on Source Code Analysis and Manipulation (SCAM)*, IEEE, 2022, pp. 71–82.

[89] G. Ilieva, T. Yankova, V. Hadzhieva, and Y. Iliev, “A Generative AI-Based Framework for Proactive Quality Assurance and Auditing,” 2026.

[90] R. V Hogg, E. A. Tanis, and D. L. Zimmerman, *Probability and statistical inference*, vol. 993. Macmillan New York, 1977.

[91] U. Sekaran and R. Bougie, *Research methods for business: A skill building approach*. john wiley & sons, 2016.

[92] R. Macefield, "How to specify the participant group size for usability studies: a practitioner's guide," *J. Usability Stud.*, vol. 5, no. 1, pp. 34–45, 2009.

[93] H. Taherdoost, "Sampling methods in research methodology; how to choose a sampling technique for research," *How to choose a sampling technique for research (April 10, 2016)*, 2016.

[94] N. Rane, S. Choudhary, and J. Rane, "Gemini versus ChatGPT: applications, performance, architecture, capabilities, and implementation," *Journal of Applied Artificial Intelligence*, vol. 5, no. 1, pp. 69–93, 2024.

[95] P. Walton, "Web Vitals: Essential metrics for a healthy site. web.dev (Google Developers).," 2020, Accessed: Feb. 21, 2026. [Online]. Available: https://web.dev/articles/vitals

[96] R. S. Pressman, *Software engineering: a practitioner's approach*. Palgrave macmillan, 2005.

[97] A. Joshi, S. Kale, S. Chandel, and D. K. Pal, "Likert scale: Explored and explained," *Br. J. Appl. Sci. Technol.*, vol. 7, no. 4, pp. 396–403, 2015.