RANCANG BANGUN VS CODE EXTENSION UNTUK IDENTIFIKASI KODE YANG BERISIKO MENGHAMBAT MAINTAINABILITY MENGGUNAKAN SOFTWARE METRICS DAN LARGE LANGUAGE MODEL

TUGAS AKHIR



DISUSUN OLEH:

EKASARI AMALIA

NIM. 21106050023

PROGRAM STUDI INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI SUNAN KALIJAGA

YOGYAKARTA

2026

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

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-2382/Un.02/DST/PP.00.9/09/2026

Tugas Akhir dengan judul : Rancang Bangun VS Code Extension untuk Identifikasi Kode yang Berisiko Menghambat Maintainability Menggunakan Software Metrics dan Large Language Model

yang dipersiapkan dan disusun oleh:

Nama : EKASARI AMALIA
Nomor Induk Mahasiswa : 21106050023
Telah diujikan pada : Jumat, 28 Agustus 2026
Nilai ujian Tugas Akhir : A-

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

TIM UJIAN TUGAS AKHIR

Ketua Sidang

Muhammad Galih Wonoseto, M.T.
SIGNED

Valid ID: 6ab8c9744bd4f

Penguji I

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

Valid ID: 6ab7914be2220

Penguji II

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

Valid ID: 6ab89ea24ee88

Yogyakarta, 28 Agustus 2026
UIN Sunan Kalijaga
Dekan Fakultas Sains dan Teknologi

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

Valid ID: 6ab9cf9b30c8a

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

LEMBAR PERNYATAAN

SURAT PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Ekasari Amalia
NIM : 21106050023
Program Studi : Informatika
Fakultas : Sains dan Teknologi
Universitas : Universitas Islam Negeri Sunan Kalijaga

Menyatakan dengan sesungguhnya bahwa skripsi saya yang berjudul "Rancang Bangun VS Code Extension untuk Identifikasi Kode yang Berisiko Menghambat Maintainability Menggunakan Software Metrics dan Large Language Model" merupakan hasil karya saya sendiri. Tidak terdapat karya yang pernah diajukan untuk memperoleh gelar kesarjanaan di suatu perguruan tinggi, dan tidak terdapat plagiasi karya orang lain, kecuali yang secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka.

Karya ini disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana Program Studi Informatika pada Fakultas Sains dan Teknologi, Universitas Islam Negeri Sunan Kalijaga.

Yogyakarta, 26 Agustus 2026
Penulis,



Ekasari Amalia
NIM.21106050023

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

LEMBAR PERSETUJUAN TUGAS AKHIR

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	: Ekasari Amalia
NIM	: 21106050023
Judul Skripsi	: Rancang Bangun VS Code Extension untuk Identifikasi Kode yang Berisiko Menghambat Maintainability Menggunakan Software Metrics dan Large Language Model

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, 24 Agustus 2026
Pembimbing

Muhammad Galih Wonoseto, M.T.
NIP. 19901113 201903 1 012

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

INTISARI

Maintainability merupakan atribut kualitas perangkat lunak yang berkaitan dengan kemudahan kode untuk dipahami, diperbaiki, dimodifikasi, dan dikembangkan. Penelitian ini bertujuan untuk mengembangkan *Maintainability Risk Analyzer* sebagai VS Code *Extension* untuk mengidentifikasi risiko *maintainability* pada kode JavaScript menggunakan *software metrics* serta memberikan penjelasan kontekstual melalui *Large Language Model* (LLM). Sistem dikembangkan menggunakan *Extreme Programming* (XP) yang meliputi *planning, design, coding,* dan *testing*. Analisis statis memanfaatkan *Abstract Syntax Tree* (AST) untuk memperoleh *Halstead Volume, Cyclomatic Complexity,* dan *Lines of Code* (LOC), yang digunakan dalam perhitungan *Visual Studio Maintainability Index* (MIVS) dan penentuan tingkat risiko. Hasil analisis ditampilkan secara langsung pada lingkungan pengembangan melalui *Diagnostic, Highlight, Hover,* dan *Show Summary*. Fitur *Explain Maintainability Risk* menyediakan penjelasan kontekstual mengenai faktor penyebab risiko serta saran *refactoring* berbasis LLM. Hasil implementasi dan pengujian menunjukkan bahwa *extension* dapat menjalankan analisis *maintainability* pada kode JavaScript, menampilkan tingkat risiko beserta metrik yang digunakan, serta menyediakan penjelasan dan saran perbaikan melalui fitur berbasis LLM. Evaluasi pengguna terhadap fitur berbasis LLM menggunakan *Extended TAM* pada 101 responden valid menunjukkan rata-rata keseluruhan sebesar 3,49 dalam kategori sangat tinggi.

Kata Kunci: Visual Studio Code*, Extension, Maintainability, Software Metrics,* JavaScript*, Large Language Model*

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

ABSTRACT

Maintainability is a software quality attribute related to the ease of understanding, repairing, modifying, and developing code. This study aims to develop the Maintainability Risk Analyzer as a VS Code *Extension to identify maintainability risks in* JavaScript *code using software metrics and provide contextual explanations through a Large Language Model* (LLM). *The system was developed using Extreme Programming (XP), consisting of planning, design, coding, and testing. Static analysis utilizes the Abstract Syntax Tree (AST) to obtain Halstead Volume, Cyclomatic Complexity, and Lines of Code (LOC), which are used to calculate the Visual Studio Maintainability Index (MIVS) and determine the risk level. The analysis results are displayed directly within the development environment through Diagnostic, Highlight, Hover, and Show Summary. The Explain Maintainability Risk feature provides contextual explanations of risk factors and LLM-based refactoring suggestions. The implementation and testing results show that the extension can perform maintainability analysis on* JavaScript *code, display risk levels and the corresponding metrics, and provide explanations and improvement suggestions through the LLM-based feature. User evaluation of the LLM-based feature using the Extended TAM among 101 valid respondents resulted in an overall mean score of 3.49, which falls into the very high category.*

Keywords: Visual Studio Code*, Extension, Maintainability, Software Metrics,* JavaScript*, Large Language Model*

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

MOTTO

"It always seems impossible until it's done."

Nelson Mandela

"Success is not final, failure is not fatal: it is the courage to continue that counts."

Winston Churchill

"The best way to predict the future is to create it."

Peter Drucker

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

HALAMAN PERSEMBAHAN

Tugas Akhir ini saya persembahkan kepada kedua orang tua saya yang selalu menjadi tempat saya kembali, sumber kekuatan, dan alasan untuk terus melangkah hingga sampai di tahap ini. Terima kasih atas doa yang tidak pernah putus, kasih sayang, dukungan, dan kepercayaan yang selalu diberikan selama perjalanan pendidikan saya. Di balik setiap langkah yang saya jalani, ada kerja keras, pengorbanan, dan ketulusan yang mungkin tidak selalu dapat saya ungkapkan, tetapi selalu saya rasakan. Ketika perjalanan ini terasa berat, dukungan dan doa dari kalian menjadi pengingat bagi saya untuk tetap bertahan dan menyelesaikan apa yang telah saya mulai. Pencapaian ini bukan hanya hasil dari usaha saya sendiri, tetapi juga bagian dari doa dan perjuangan yang telah kalian berikan sejak awal. Tugas Akhir ini juga saya persembahkan kepada kedua adik saya yang selalu menghadirkan semangat dan keceriaan dengan caranya masing-masing, serta kepada teman-teman seperjuangan yang telah menemani perjalanan perkuliahan, berbagi cerita, dan saling menguatkan hingga proses ini dapat diselesaikan.

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

KATA PENGANTAR

Alhamdulillahirabbil'alamin, segala puji dan syukur penulis panjatkan ke hadirat Allah SWT atas segala rahmat, hidayah, dan karunia-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul "Rancang Bangun VS Code Extension untuk Identifikasi Kode yang Berisiko Menghambat Maintainability Menggunakan Software Metrics dan Large Language Model" dengan baik.

Penyusunan Tugas Akhir ini merupakan salah satu syarat untuk memperoleh gelar Sarjana pada Program Studi Informatika, Fakultas Sains dan Teknologi, Universitas Islam Negeri Sunan Kalijaga Yogyakarta. Dalam proses penyusunan ini, penulis menyadari bahwa banyak tantangan dan kendala yang dihadapi. Namun, berkat bantuan, arahan, dan dukungan dari berbagai pihak, tugas akhir ini dapat diselesaikan.

Pada kesempatan ini, 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. Dr. Aulia selaku dosen penasihat akademik yang telah memberikan arahan selama penulis menempuh masa studi.
5. Muhammad Galih Wonoseto, M.T. selaku dosen pembimbing yang telah memberikan bimbingan dalam tahapan penyusunan Tugas Akhir ini.
6. Seluruh dosen dan karyawan Program Studi Informatika UIN Sunan Kalijaga Yogyakarta yang telah memberikan ilmu dan pengalaman selama penulis menempuh pendidikan.
7. Kedua orang tua penulis, Ibu Armina Sihak dan Bapak Muri, selalu memberikan doa, dukungan, serta kepercayaan kepada penulis dalam setiap proses yang penulis jalani.

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

8. Kedua adik penulis, Muhammad Rafi'ul Anam dan Muhammad Fahrisalam, yang telah memberikan semangat dan menjadi motivasi penulis dalam menyelesaikan Tugas Akhir ini.
9. Teman-teman seperjuangan yang telah memberikan dukungan, berbagi pengalaman, serta menemani proses belajar hingga penyusunan Tugas Akhir penulis. Serta semua pihak yang telah membantu penulis, baik secara langsung maupun tidak langsung, yang tidak dapat disebutkan satu per satu.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki keterbatasan. Oleh karena itu, penulis sangat mengharapkan kritik dan saran yang membangun sebagai bahan perbaikan di masa mendatang. Semoga Tugas Akhir ini dapat memberikan manfaat bagi pembaca serta menjadi referensi untuk pengembangan penelitian selanjutnya.

Yogyakarta, 24 Agustus 2026
Penulis,

Ekasari Amalia
NIM.21106050023

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

DAFTAR ISI

LEMBAR PENGESAHAN i
LEMBAR PERNYATAAN ii
LEMBAR PERSETUJUAN TUGAS AKHIR iii
INTISARI iv
ABSTRACT v
MOTTO vi
HALAMAN PERSEMBAHAN vii
KATA PENGANTAR viii
DAFTAR ISI x
DAFTAR TABEL xiii
DAFTAR GAMBAR xv
DAFTAR LAMPIRAN xvi
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 3
1.3 Batasan Masalah 3
1.4 Tujuan 3
1.5 Manfaat 4
BAB II KAJIAN PUSTAKA 5
2.1 *Maintainability* 5
2.2 JavaScript 6
2.3 Analisis Statis dan *Abstract Syntax Tree* (AST) 6
2.3.1 *Abstract Syntax Tree* pada Kode JavaScript 6
2.3.2 Ekstraksi *Software Metrics* Berbasis AST 7
2.4 *Software Metrics* 8
2.4.1 *Lines of Code* (LOC) 10
2.4.2 *Cyclomatic Complexity* (CC) 11
2.4.3 *Halstead Volume* (HV) 11
2.5 *Maintainability Index* 12

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

2.5.1 Konsep dan Perhitungan *Maintainability Index* 12
2.5.2 Interpretasi Nilai dan Kategori Risiko 17
2.6 Visual Studio Code *Extension* 18
2.7 *Large Language Model* 19
2.8 *Extreme Programming* 21
2.8.1 Perencanaan (*Planning*) 22
2.8.2 Perancangan (*Design*) 22
2.8.3 Pengkodean (*Coding*) 24
2.8.4 Pengujian (*Testing*) 24
2.9 *Extended Technology Acceptance Model* 25
2.10 Penelitian Terdahulu 27
BAB III METODE PENGEMBANGAN SISTEM 31
3.1 Lokasi dan Waktu Penelitian 31
3.2 Alat dan Bahan 31
3.3 Alur Pengembangan Sistem 32
3.3.1 Studi Literatur 33
3.3.2 Perencanaan (*Planning*) 33
3.3.3 Perancangan (*Design*) 34
3.3.4 Implementasi (*Coding*) 34
3.3.5 Pengujian (*Testing*) 35
3.3.6 Evaluasi Sistem Menggunakan *Extended TAM* 36
BAB IV PERANCANGAN DAN IMPLEMENTASI SISTEM 39
4.1 Perencanaan Sistem (*Planning*) 39
4.1.1 *User Stories* 39
4.1.2 Kebutuhan Fungsional 41
4.1.3 Kebutuhan Non-Fungsional 41
4.2 Perancangan Sistem (*Design*) 42
4.2.1 *Component Diagram* 42
4.2.2 *Use Case Diagram* 45
4.3 Implementasi Sistem (*Coding*) 46
4.3.1 Implementasi Identifikasi Fungsi Berisiko terhadap *Maintainability* .. 46

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

4.3.2 Implementasi Penyajian Hasil Analisis pada VS Code 58
4.3.3 Implementasi Ringkasan Hasil Analisis *Maintainability* 59
4.3.4 Implementasi LLM 60
4.4 Pengujian Sistem 71
4.4.1 *Unit Testing* 71
4.4.2 *Black Box Testing* 75
4.5 Evaluasi Pengguna Menggunakan *Extended TAM* 79
4.6 Dokumentasi 83
BAB V PENUTUP 85
5.1 Kesimpulan 85
5.2 Saran 85
DAFTAR PUSTAKA 86
LAMPIRAN 91

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

DAFTAR TABEL

Tabel 2.1 Fungsi *calculateCustomerRisk* 14
Tabel 2.2 Komponen HV dari fungsi *calculateCustomerRisk*() 16
Tabel 2.3 Interpretasi Risiko *Visual Studio Maintainability Index* 17
Tabel 2.4 Kajian Pustaka yang Relevan dengan Penelitian 27
Tabel 3.1 Skala Likert 37
Tabel 3.2 Interpretasi Skor Skala Likert 38
Tabel 4.1 *User Stories* 39
Tabel 4.2 Kebutuhan Fungsional 41
Tabel 4.3 Kebutuhan Non-Fungsional 41
Tabel 4.4 Implementasi Pembentukan AST Menggunakan Babel Parser 47
Tabel 4.5 Identifikasi Jenis Fungsi 48
Tabel 4.6 Pengumpulan Fungsi dari AST 48
Tabel 4.7 Perhitungan *Lines of Code* 49
Tabel 4.8 Perhitungan *Cyclomatic Complexity* 50
Tabel 4.9 Inisialisasi Operator dan Operand untuk *Halstead* 52
Tabel 4.10 Potongan Kode Identifikasi Operator *Halstead* 53
Tabel 4.11 Potongan Kode Identifikasi Operand *Halstead* 54
Tabel 4.12 Perhitungan Parameter *Halstead Volume* 55
Tabel 4.13 Implementasi Perhitungan MIVS 56
Tabel 4.14 Implementasi Klasifikasi Risiko Berdasarkan MIVS 56
Tabel 4.15 Penggabungan Hasil Analisis Fungsi 57
Tabel 4.16 Potongan Kode Pemanggilan Fitur LLM 61
Tabel 4.17 Potongan Kode Pembentukan Konteks Analisis 62
Tabel 4.18 Potongan Kode Pengiriman Permintaan ke *Backend* 63
Tabel 4.19 Potongan Kode Instruksi Utama LLM 64
Tabel 4.20 Potongan Kode Instruksi Berdasarkan Tingkat Risiko 65
Tabel 4.21 Potongan Kode Pembentukan *Request* OpenAI 66
Tabel 4.22 Potongan Kode Pengiriman *Request* ke OpenAI API 67
Tabel 4.23 Potongan Kode Pengambilan Respons OpenAI 67
Tabel 4.24 Potongan Kode Pemrosesan Respons pada *Extension* 68
Tabel 4.25 Potongan Kode *Fallback* ketika LLM Tidak Tersedia 70

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

Tabel 4.26 Hasil *Unit Testing* 72
Tabel 4.27 Hasil *Black Box Testing* Iterasi Pertama 75
Tabel 4.28 Hasil *Black Box Testing* Iterasi Kedua 77
Tabel 4.29 Karakteristik Responden 79
Tabel 4.30 Hasil Uji Reliabilitas Instrumen *Extended TAM* 81
Tabel 4.31 Distribusi Jawaban dan Rata-rata Item *Extended TAM* 82
Tabel 4.32 Rata-rata dan Interpretasi Setiap Konstruk *Extended TAM* 83

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

DAFTAR GAMBAR

Gambar 2.1 Representasi Struktur AST pada Fungsi *calculateCustomerRisk* 7
Gambar 2.2 Alur Ekstraksi *Software Metrics* Berbasis AST 8
Gambar 2.3 Atribut *Maintainability* pada Model *Software Quality* 9
Gambar 2.4 Integrasi LLM dalam Analisis *Maintainability* 20
Gambar 2.5 Siklus Metode *Extreme Programming* (XP) 21
Gambar 2.6 Simbol *Component Diagram* 23
Gambar 2.7 Simbol *Use Case Diagram* 24
Gambar 2.8 Model *Extended TAM* pada Adopsi *AI Tools* 26
Gambar 3.1 Alur Pengembangan Sistem Berdasarkan *Extreme Programming*. .. 32
Gambar 4.1 *Component Diagram Maintainability Risk Analyzer* 43
Gambar 4.2 *Use Case Diagram Maintainability Risk Analyzer* 45
Gambar 4.3 Hasil Penyajian Risiko pada VS Code 59
Gambar 4.4 Hasil *Summary* Analisis *Maintainability* 60
Gambar 4.5 Hasil Penjelasan dan Saran *Refactoring* Berbasis LLM 69
Gambar 4.6 Hasil *Test Case* pada *Unit Testing* 74
Gambar 4.7 Distribusi Pendidikan Terakhir Responden 79

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan perangkat lunak yang semakin cepat menyebabkan aplikasi terus mengalami perubahan untuk menyesuaikan kebutuhan pengguna, penambahan fitur, perbaikan kesalahan, dan perubahan lingkungan teknologi. Dalam kondisi tersebut, kode sumber tidak berhenti digunakan setelah perangkat lunak selesai dikembangkan, tetapi terus menjadi bagian dari proses pengembangan dan pemeliharaan. Semakin besar ukuran dan kompleksitas perangkat lunak, semakin penting kemampuan untuk memahami, memperbaiki, dan memodifikasi kode secara efisien agar perubahan yang dilakukan tidak menimbulkan permasalahan baru [1][2]. Kondisi tersebut menjadikan *maintainability* sebagai salah satu aspek kualitas perangkat lunak yang relevan dalam pengembangan perangkat lunak saat ini. *Maintainability* merupakan aspek penting dalam kualitas perangkat lunak karena berkaitan dengan kemudahan perangkat lunak untuk diperbaiki, disesuaikan, dan dikembangkan setelah digunakan [1].

Salah satu pendekatan untuk menilai kondisi *maintainability* adalah menggunakan *software metrics* untuk mengukur karakteristik kode secara kuantitatif [8]. JavaScript banyak digunakan dalam pengembangan perangkat lunak dan karakteristik kodenya dapat dianalisis menggunakan berbagai *software metrics* [4][14]. Salah satu pendekatan yang dapat digunakan adalah *Maintainability Index* (MI), yang menggabungkan *Halstead Volume, Cyclomatic Complexity,* dan *Lines of Code* (LOC) untuk menghasilkan nilai yang menggambarkan tingkat *maintainability* kode [2].

Namun, hasil pengukuran berupa nilai atau kategori risiko belum secara langsung menjelaskan penyebab risiko maupun kemungkinan perbaikannya. Penelitian terdahulu telah mengkaji berbagai pendekatan yang berkaitan dengan pengukuran *maintainability* dan pemanfaatan LLM dalam pengembangan perangkat lunak. Welker, Oman, dan Atkinson [3] mengembangkan *Maintainability Index* (MI) untuk memberikan ukuran kuantitatif terhadap

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

maintainability perangkat lunak. Zozas et al. [4] mengkaji karakteristik kode dan *maintenance effort* pada aplikasi JavaScript, sedangkan Heričko dan Šumak [1] menunjukkan bahwa pemilihan varian *Maintainability Index* dapat memengaruhi hasil pengukuran *maintainability*. Penelitian lain oleh Alomari et al. [26] menunjukkan pemanfaatan *Large Language Model* (LLM) dalam aktivitas peningkatan kualitas kode, termasuk *refactoring* dan *code smell detection*. Sementara itu, Nam et al. [25] menunjukkan penggunaan LLM yang terintegrasi melalui *plugin* pada IDE untuk mendukung aktivitas *code understanding*.

Meskipun penelitian-penelitian tersebut telah memberikan kontribusi pada aspek pengukuran *maintainability*, analisis kode JavaScript, maupun pemanfaatan LLM dalam lingkungan pengembangan, masing-masing penelitian masih memiliki fokus yang berbeda. Khususnya, hasil pengukuran *maintainability* berupa nilai atau kategori belum secara langsung memberikan penjelasan kontekstual mengenai faktor yang menyebabkan suatu bagian kode memiliki risiko *maintainability* tertentu maupun kemungkinan perbaikannya. Di sisi lain, pemanfaatan LLM dalam penelitian terkait belum secara khusus diarahkan untuk memberikan penjelasan berdasarkan hasil analisis risiko *maintainability* yang diperoleh dari *software metrics.* Dengan demikian, masih terdapat kebutuhan untuk mengintegrasikan analisis struktur kode, *software metrics* dan *Maintainability Index*, klasifikasi risiko, serta penjelasan kontekstual berbasis LLM dalam satu alat bantu pada lingkungan pengembangan.

Berdasarkan kesenjangan tersebut, penelitian ini mengembangkan VS Code *Extension* yang mengintegrasikan analisis statis berbasis *Abstract Syntax Tree* (AST), *software metrics*, dan *Large Language Model* (LLM) untuk menganalisis risiko *maintainability* pada kode JavaScript. *Software metrics* digunakan untuk menghasilkan analisis risiko, sedangkan LLM digunakan untuk memberikan penjelasan kontekstual dan saran *refactoring* berdasarkan hasil analisis [26]. Hasil analisis tersebut kemudian diintegrasikan ke dalam VS Code agar dapat diberikan sebagai umpan balik langsung kepada pengembang. Pengembangan sistem dilakukan menggunakan metode *Extreme Programming* (XP) melalui *planning, design, coding,* dan *testing* berdasarkan kebutuhan dan *user stories.* Setelah sistem

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

dikembangkan dan diuji, penerimaan pengguna terhadap fitur berbasis LLM dievaluasi menggunakan *Extended Technology Acceptance Model* dengan mengadaptasi konstruk yang relevan dari penelitian Mustofa et al. [35].

1.2 Rumusan Masalah

1. Bagaimana mengidentifikasi bagian kode JavaScript yang berisiko menghambat *maintainability* menggunakan pendekatan *software metrics*?
2. Bagaimana mengintegrasikan hasil analisis *software metrics* ke dalam VS Code sehingga dapat memberikan umpan balik secara langsung kepada pengembang?
3. Bagaimana memanfaatkan *Large Language Model* (LLM) untuk memberikan penjelasan kontekstual terhadap hasil analisis risiko *maintainability* pada kode?

1.3 Batasan Masalah

Penelitian ini memiliki beberapa batasan. Analisis *maintainability* dibatasi pada kode JavaScript dengan ekstensi .js, .mjs, dan .cjs serta menggunakan *Halstead Volume, Cyclomatic Complexity,* dan *Lines of Code* sebagai *software metrics* dalam perhitungan MIVS. LLM digunakan sebagai komponen pendukung untuk memberikan penjelasan dan saran *refactoring* berdasarkan hasil analisis, sehingga penelitian ini tidak mengevaluasi kebenaran atau efektivitas saran *refactoring* secara khusus. Evaluasi penerimaan pengguna menggunakan kuesioner *Extended TAM* sehingga konstruk *Actual Use* merepresentasikan penggunaan yang dilaporkan oleh responden dan bukan data telemetri penggunaan *extension*.

1.4 Tujuan

Tujuan dari penelitian ini adalah sebagai berikut:

1. Mengintegrasikan *Abstract Syntax Tree* (AST) sebagai dasar ekstraksi karakteristik kode dan *software metrics* untuk menganalisis risiko *maintainability* pada kode JavaScript.

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

2. Mengimplementasikan hasil analisis *software metrics* ke dalam VS Code sehingga dapat memberikan umpan balik secara langsung kepada pengembang.
3. Menerapkan *Large Language Model* (LLM) untuk memberikan penjelasan kontekstual atas hasil analisis risiko *maintainability* pada kode.

1.5 Manfaat

Penelitian ini diharapkan dapat memberikan manfaat dalam bidang rekayasa perangkat lunak melalui pengembangan VS Code *Extension* yang menerapkan *Maintainability Index* untuk membantu mengidentifikasi risiko *maintainability* pada kode JavaScript. Secara praktis, *extension* diharapkan dapat membantu pengguna memahami risiko berdasarkan nilai metrik dan kategori risiko serta memperoleh penjelasan kontekstual melalui LLM. Penelitian ini juga diharapkan dapat menjadi referensi bagi pengembangan alat bantu analisis kode yang mengintegrasikan analisis statis, *software metrics*, dan LLM.

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan, dapat disimpulkan bahwa:

1. Identifikasi risiko *maintainability* pada kode JavaScript dapat dilakukan menggunakan *software metrics*. Analisis berbasis AST menghasilkan *Halstead Volume, Cyclomatic Complexity,* dan *Lines of Code* yang digunakan dalam perhitungan MIVS dan dipetakan menjadi kategori *Low Risk*, *Medium Risk*, dan *High Risk*, sehingga bagian kode yang berisiko dapat diidentifikasi.
2. Hasil analisis *software metrics* berhasil diintegrasikan ke dalam VS Code sebagai umpan balik langsung melalui *diagnostic, highlight, hover,* dan *Show Summary*, sehingga pengembang dapat mengetahui kondisi *maintainability* kode tanpa berpindah dari lingkungan pengembangan.
3. *Large Language Model* (LLM) berhasil diintegrasikan pada fitur *Explain Maintainability Risk* untuk menghasilkan penjelasan kontekstual dan saran *refactoring* berdasarkan hasil analisis maintainability. Evaluasi *Extended TAM* terhadap 101 responden valid memperoleh rata-rata 3,49 dalam kategori Sangat Tinggi. Penelitian ini tidak mengevaluasi kebenaran atau efektivitas saran *refactoring* yang dihasilkan LLM.

5.2 Saran

Pengembangan selanjutnya dapat memperluas *software metrics* dan metode *Maintainability Index*, meningkatkan kualitas fitur LLM melalui konteks analisis dan evaluasi beberapa model, serta melakukan pengujian pada proyek JavaScript yang lebih beragam dan pengguna dengan tingkat pengalaman berbeda. Evaluasi pengguna juga dapat dilakukan dalam periode yang lebih panjang untuk memperoleh gambaran penerimaan fitur *Explain Maintainability Risk* berdasarkan penggunaan dalam kondisi pengembangan perangkat lunak yang lebih nyata.

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

DAFTAR PUSTAKA

[1] T. Heričko and B. Šumak, "Exploring Maintainability Index Variants for Software *Maintainability* Measurement in Object-Oriented Systems," *Applied Sciences*, vol. 13, no. 5, Art. no. 2972, 2023, doi: 10.3390/app13052972. [Online]. Available: https://doi.org/10.3390/app13052972.

[2] B. Idrisov and T. Schlippe, "Program code generation with generative AIs," *Algorithms*, vol. 17, no. 2, Art. no. 62, 2024. [Online]. Available: https://doi.org/10.3390/a17020062

[3] K. D. Welker, P. W. Oman, and G. G. Atkinson, "Development and application of an automated source code maintainability index," *Journal of Software Maintenance: Research and Practice*, vol. 9, no. 3, pp. 127–159, 1997. [Online]. Available: https://doi.org/10.1002/(SICI)1096-908X(199705)9:3<127::AID-SMR149>3.0.CO;2-S

[4] I. Zozas, S. Bibi, A. Ampatzoglou, and P. G. Sarigiannidis, "Estimating the maintenance effort of JavaScript applications," in *Proc. 45th Euromicro Conf. Software Engineering and Advanced Applications (SEAA)*, pp. 212–219, 2019, doi: 10.1109/SEAA.2019.00042. [Online]. Available: https://ieeexplore.ieee.org/document/8906670/

[5] ISO/IEC, *ISO/IEC 25010:2023 Systems and Software Engineering—Systems and Software Quality Requirements and Evaluation (SQuaRE)—Product Quality Model*, 2023. [Online]. Available: https://www.iso.org/standard/78176.html

[6] A.-J. Molnar and S. Motogna, "Longitudinal evaluation of open-source software maintainability," in *Proc. 15th Int. Conf. Evaluation of Novel Approaches to Software Engineering (ENASE)*, vol. 1, pp. 120–131, 2020. [Online]. Available: https://doi.org/10.5220/0009393501200131

[7] M. D. Papamichail and A. L. Symeonidis, "A generic methodology for early identification of non-maintainable source code components through analysis of software releases," *Information and Software Technology*, vol. 118, Art. no. 106218, 2020. [Online]. Available: https://doi.org/10.1016/j.infsof.2019.106218

[8] I. Heitlager, T. Kuipers, and J. Visser, "A practical model for measuring maintainability," in *Proc. 6th International Conference on Quality of Information and Communications Technology (QUATIC)*, pp. 30–39, 2007, doi: 10.1109/QUATIC.2007.8. [Online]. Available: https://doi.org/10.1109/QUATIC.2007.8

[9] S. Burger and O. Hummel, "Lessons learnt from gauging software metrics of cabin software in a commercial airliner," *International Scholarly Research Notices*, vol. 2012, Art. no. 162305, 2012, doi:

STATE ISLAMIC UNIVERSITY SUNAN KALIJAGA YOGYAKARTA

10.5402/2012/162305. [Online]. Available: https://doi.org/10.5402/2012/162305

[10] A. C. M. Matos, M. M. Barros, R. A. B. Freitas, and A. R. C. da Silva, "A maintainability model for software product quality assessment," in *Proc. 13th International Conference on Enterprise Information Systems*, 2011. [Online]. Available: https://www.scitepress.org/PublishedPapers/2011/34492/

[11] A. A. Ghani and R. Hunter, "An attribute grammar approach to specifying Halstead's metrics," *Malaysian Journal of Computer Science*, vol. 9, no. 1, pp. 56–67, 1996. [Online]. Available: https://ejournal.um.edu.my/index.php/MJCS/article/view/2987

[12] A. M. Teixeira, A. A. A. Silva Filho, and J.-R. Bourguet, "Revisiting Halstead's metrics through abstract syntax trees in C," in *Proc. IEEE/ACM 2nd International Workshop on Advancing Static Analysis for Researchers and Industry Practitioners in Software Engineering*, pp. 5–11, 2026. [Online]. Available: https://doi.org/10.1145/3786173.3788356

[13] Microsoft, "Code Metrics—Maintainability Index Range and Meaning," *Microsoft Learn*. [Online]. Available: https://learn.microsoft.com/en-us/visualstudio/code-quality/code-metrics-maintainability-index-range-and-meaning

[14] I. Zozas, S. Bibi, A. Ampatzoglou, E.-M. Arvanitou, P. Angelidis, and M. Tsipouras, "A mapping study on JavaScript quality attributes and metrics," *Journal of Software: Evolution and Process*, vol. 37, no. 12, Art. no. e70073, 2025, doi: 10.1002/smr.70073. [Online]. Available: https://doi.org/10.1002/smr.70073

[15] S. Mohammad, A. Al Jobair, and I. Abedeen, "Decoding code quality: A software metric analysis of open-source JavaScript projects," in *Proc. 19th Int. Conf. Evaluation of Novel Approaches to Software Engineering (ENASE)*, pp. 63–74, 2024, doi: 10.5220/0012618800003687. [Online]. Available: https://doi.org/10.5220/0012618800003687

[16] Ecma International, *ECMA-262: ECMAScript 2026 Language Specification*, 17th ed., Jun. 2026. [Online]. Available: https://ecma-international.org/publications-and-standards/standards/ecma-262/

[17] MDN Web Docs, "Functions — JavaScript," *Mozilla Developer Network*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions

[18] NIST Computer Security Resource Center, "Static Code Analyzer," *NIST CSRC Glossary*. [Online]. Available: https://csrc.nist.gov/glossary/term/static_code_analyzer

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

[19] Babel, "@babel/parser," Babel *Documentation*. [Online]. Available: https://babeljs.io/docs/babel-parser

[20] L. Ardito, L. Barbato, M. Castelluccio, R. Coppola, C. Denizet, S. Ledru, and M. Valsesia, "rust-code-analysis: A Rust library to analyze and extract maintainability information from source codes," *SoftwareX*, vol. 12, Art. no. 100635, 2020, doi: 10.1016/j.softx.2020.100635. [Online]. Available: https://doi.org/10.1016/j.softx.2020.100635

[21] Microsoft, "Extension API," VS Code Extension API. [Online]. Available: https://code.visualstudio.com/api

[22] Microsoft, "Programmatic Language Features," VS Code Extension API. [Online]. Available: https://code.visualstudio.com/api/language-extensions/programmatic-language-features

[23] Microsoft, "Webview API," VS Code Extension API. [Online]. Available: https://code.visualstudio.com/api/extension-guides/webview

[24] T. B. Brown *et al.*, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html

[25] D. Nam, A. Macvean, V. J. Hellendoorn, B. Vasilescu, and B. A. Myers, "Using an LLM to help with code understanding," in *Proc. IEEE/ACM 46th International Conference on Software Engineering (ICSE)*, 2024, doi: 10.1145/3597503.3639187. [Online]. Available: https://doi.org/10.1145/3597503.3639187

[26] N. Alomari, M. Redah, A. Ashraf, and M. Alshayeb, "Using LLMs to enhance code quality: A systematic literature review," *Information and Software Technology*, vol. 190, Art. no. 107960, 2026, doi: 10.1016/j.infsof.2025.107960. [Online]. Available: https://doi.org/10.1016/j.infsof.2025.107960

[27] OpenAI, "Developer Quickstart," OpenAI API *Documentation*. [Online]. Available: https://developers.openai.com/api/docs/quickstart

[28] K. Beck and C. Andres, *Extreme Programming Explained: Embrace Change*, 2nd ed. Boston, MA, USA: Addison-Wesley Professional, 2004. [Online]. Available: https://www.informit.com/store/extreme-programming-explained-embrace-change-9780321278654

[29] A. Shrivastava, I. Jaggi, N. Katoch, D. Gupta, and S. Gupta, "A systematic review on extreme programming," *Journal of Physics: Conference Series*, vol. 1969, no. 1, Art. no. 012046, 2021, doi: 10.1088/1742-6596/1969/1/012046. [Online]. Available: https://doi.org/10.1088/1742-6596/1969/1/012046

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

[30] Object Management Group, *OMG Unified Modeling Language (OMG UML), Version 2.5.1*, Dec. 2017. [Online]. Available: https://www.omg.org/spec/UML/2.5.1

[31] P. Ammann and J. Offutt, *Introduction to Software Testing*, 2nd ed. Cambridge, U.K.: Cambridge University Press, 2016. [Online]. Available: https://www.cambridge.org/highereducation/books/introduction-to-software-testing/95E57CCADEA697EC8594F03729F47311

[32] B. R. Siqueira, F. C. Ferrari, K. E. Souza, V. V. Camargo, and R. de Lemos, "Testing of adaptive and context-aware systems: Approaches and challenges," *Software Testing, Verification and Reliability*, vol. 31, no. 7, Art. no. e1772, 2021, doi: 10.1002/stvr.1772. [Online]. Available: https://doi.org/10.1002/stvr.1772

[33] F. D. Davis, "Perceived usefulness, perceived ease of use, and user acceptance of information technology," *MIS Quarterly*, vol. 13, no. 3, pp. 319–340, 1989, doi: 10.2307/249008. [Online]. Available: https://doi.org/10.2307/249008

[34] V. Venkatesh and F. D. Davis, "A theoretical extension of the Technology Acceptance Model: Four longitudinal field studies," *Management Science*, vol. 46, no. 2, pp. 186–204, 2000, doi: 10.1287/mnsc.46.2.186. [Online]. Available: https://doi.org/10.1287/mnsc.46.2.186

[35] R. H. Mustofa, T. G. Kuncoro, D. Atmono, H. D. Hermawan, and Sukirman, "Extending the Technology Acceptance Model: The role of subjective norms, ethics, and trust in AI tool adoption among students," *Computers and Education: Artificial Intelligence*, vol. 8, Art. no. 100379, 2025, doi: 10.1016/j.caeai.2025.100379. [Online]. Available: https://doi.org/10.1016/j.caeai.2025.100379

[36] M. L. B. M. Khidir, S. N. B. Sa'ari, and N. I. @ N. B. Ayob, "Evaluating an AI-Integrated E-Learning Portal in Malaysian TVET: A DeLone-McLean and TAM Perspective on SmartGradeAi," EPRA International Journal of Multidisciplinary Research, vol. 12, no. 5, May 2026, doi: 10.36713/epra27505. [Online]. Available: https://doi.org/10.36713/epra27505

[37] IONOS, "UML Component Diagram: Explanation, Drawing, and Example," IONOS Digital Guide. [Online]. Available: https://www.ionos.ca/digitalguide/websites/web-development/uml-component-diagram/

[38] A. Mufid, "Use Case Diagram Adalah: Definisi, Simbol dan Cara Membuat," Rumahweb, Aug. 4, 2023. [Online]. Available: https://blog.rumahweb.com/use-case-diagram-adalah

STATE ISLAMIC UNIVERSITY SUNAN KALIJAGA YOGYAKARTA

[39] A. R. Ibrahim and M. A. Nemt-allah, "Development and psychometric validation of the AI chatbots acceptance and perception scale for higher education students," *Scientific Reports*, vol. 16, no. 1, Art. no. 22941, 2026, doi: 10.1038/s41598-026-62798-4. [Online]. Available: https://doi.org/10.1038/s41598-026-62798-4

[40] P. M. Podsakoff, S. B. MacKenzie, and N. P. Podsakoff, "Sources of method bias in social science research and recommendations on how to control it," *Annual Review of Psychology*, vol. 63, pp. 539–569, 2012, doi: 10.1146/annurev-psych-120710-100452. [Online]. Available: https://doi.org/10.1146/annurev-psych-120710-100452

[41] A. J. Nederhof, "Methods of coping with social desirability bias: A review," *European Journal of Social Psychology*, vol. 15, no. 3, pp. 263–280, 1985, doi: 10.1002/ejsp.2420150303. [Online]. Available: https://doi.org/10.1002/ejsp.2420150303

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA