RANCANG BANGUN ALAT ANALISIS *MAINTAINABILITY INDEX* DAN DETEKSI *CODE SMELLS* PADA BAHASA PEMROGRAMAN PHP BERBASIS *SOFTWARE METRICS*

TUGAS AKHIR

DISUSUN OLEH :

ZILDAN PANDARU SIH MARGINATA

NIM. 21106050032

PROGRAM STUDI INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI SUNAN KALIJAGA

YOGYAKARTA

2025

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-1129/Un.02/DST/PP.00.9/06/2025

Tugas Akhir dengan judul : Rancang Bangun Alat Analisis Maintainability Index dan Deteksi Code Smells pada Bahasa Pemrograman PHP Berbasis Software Metrics

yang dipersiapkan dan disusun oleh:

Nama : ZILDAN PANDARU SIH MARGINATA
Nomor Induk Mahasiswa : 21106050032
Telah diujikan pada : Rabu, 11 Juni 2025
Nilai ujian Tugas Akhir : A

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

TIM UJIAN TUGAS AKHIR

Ketua Sidang

Dr. Siti Mutmainah, S.Kom., M.Cs.
SIGNED

Valid ID: 684ad9f047e71

Penguji I

Ir. Maria Ulfah Siregar, S.Kom., MIT., Ph.D.
SIGNED

Valid ID: 684f80e34e863

Penguji II

Dwi Otik Kurniawati, M.Eng.
SIGNED

Valid ID: 684f8f0115104

Yogyakarta, 11 Juni 2025
UIN Sunan Kalijaga
Dekan Fakultas Sains dan Teknologi

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

Valid ID: 684fb9a44c4d9

SURAT PERNYATAAN KEASLIAN

Yang bertanda tangan di bawah ini:

Nama	: Zildan Pandaru Sih Marginata
NIM	: 21106050032
Program Studi	: Informatika
Fakultas	: Sains dan Teknologi

Menyatakan bahwa Tugas Akhir saya yang berjudul "Rancang Bangun Alat Analisis *Maintainability Index* dan Deteksi *Code Smells* pada Bahasa Pemrograman PHP Berbasis *Software Metrics*" merupakan hasil karya saya sendiri, tidak terdapat pada karya yang pernah diajukan untuk memperoleh gelar akademik di suatu perguran tinggi, dan sepanjang pengetahuan saya, tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan orang lain, kecuali secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka. Berdasarkan pernyataan tersebut, Tugas Akhir ini saya ajukan untuk memenuhi salah satu syarat memperoleh gelar Sarjana pada Program Studi Informatika, Fakultas Sains dan Teknologi, Universitas Islam Negeri Sunan Kalijaga Yogyakarta.

Yogyakarta, 26 Mei 2025

Zildan Pandaru Sih Marginata

NIM. 21106050032

SURAT PERSETUJUAN TUGAS AKHIR

Hal : Surat Persetujuan 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, mengoreksi, serta mengadakan perbaikan seperlunya, maka kami selaku pembimbing berpendapat bahwa Tugas Akhir saudara:

Nama	:	Zildan Pandaru Sih Marginata
NIM	:	21106050032
Judul Tugas Akhir	:	Rancang Bangun Alat Analisis *Maintainability Index* dan Deteksi *Code Smells* pada Bahasa Pemrograman PHP Berbasis *Software Metrics*

sudah dapat diajukan kembali kepada Program Studi Informatika, Fakultas Sains dan Teknologi, Universitas Islam Negeri Sunan Kalijaga Yogyakarta, sebagai salah satu syarat untuk memperoleh gelar Sarjana Strata Satu (S1) dalam Program Studi Informatika.

Dengan ini kami berharap agar Tugas Akhir tersebut dapat segera di-*munaqasyah*-kan. Atas perhatiannya, kami ucapkan terima kasih.

Wassalamualaikum Wr. Wb.

Yogyakarta, 22 Mei 2025
Pembimbing,

Dr. Siti Mutmainah, S.Kom., M.Cs.
NIP. 19791204 200604 2 003

INTISARI

Pemeliharaan perangkat lunak merupakan aspek penting dalam pengembangan jangka panjang, di mana kualitas kode sangat mempengaruhi efisiensi dan keberhasilan proses. Kode yang tidak terstruktur atau sulit dipahami dapat menyulitkan proses refaktorisasi, meningkatkan risiko bug, dan memperlambat pengembangan fitur baru. Penelitian ini bertujuan untuk mengembangkan alat bantu analisis *Maintainability Index* dan deteksi *code smells* berbasis metrik perangkat lunak pada kode PHP yang menggunakan paradigma *Object Oriented Programming (OOP)*. Alat ini dirancang untuk membantu pengembang mengevaluasi dan meningkatkan kualitas kode melalui pendekatan kuantitatif yang terukur. Sistem dikembangkan menggunakan metode *waterfall*, dengan menerapkan teknik *code parsing* dan perhitungan metrik perangkat lunak. Pengujian dilakukan pada proyek sederhana untuk menguji akurasi sistem, dan juga diterapkan pada studi kasus Aplikasi Lapor Diri Prodi PPG UIN Sunan Kalijaga yang dikembangkan oleh Pusat Teknologi Informasi dan Pangkalan Data (PTIPD). Hasil menunjukkan bahwa alat mampu menghitung metrik dan mendeteksi *code smells* secara akurat, serta menemukan 35,5% file dengan *maintainability* rendah. Evaluasi *usability* menggunakan metode *System Usability Scale* (SUS) menghasilkan skor 80, yang menunjukkan sistem mudah digunakan dan dapat diterima oleh pengguna. Meski terdapat keterbatasan pada proyek berskala besar, alat ini menunjukkan potensi baik untuk mendukung peningkatan kualitas kode perangkat lunak.

Kata kunci: *Maintainability Index*, *Code Smells*, PHP.

ABSTRACT

Software maintenance is a crucial aspect of long-term development, where code quality significantly influences the efficiency and success of the process. Unstructured or hard-to-understand code can complicate refactoring, increase the risk of bugs, and slow down the development of new features. This study aims to develop a tool for analyzing the Maintainability Index and detecting code smells based on software metrics in PHP code using the Object-Oriented Programming (OOP) paradigm. The tool is designed to assist developers in evaluating and improving code quality through a measurable, quantitative approach. The system was developed using the waterfall methodology by applying code parsing techniques and calculating software metrics. Testing was conducted on a simple project to verify the accuracy of the system, and it was further applied to a case study involving the Self-Report Application of the PPG Study Program at UIN Sunan Kalijaga, developed by the Center for Information Technology and Databases (PTIPD). The results show that the tool can accurately calculate metrics and detect code smells, identifying 35.5% of files with low maintainability. A usability evaluation using the System Usability Scale (SUS) yielded a score of 80, indicating that the system is easy to use and well-received by users. Despite limitations in analyzing large-scale projects, the tool demonstrates strong potential to support software code quality improvement efforts.

Keywords: Maintainability Index, Code Smells, PHP.

MOTTO

“Bukan bakat yang membawa ke puncak, tapi tekad yang tak pernah padam”

HALAMAN PERSEMBAHAN

Dengan penuh rasa syukur dan hormat, tugas akhir ini kupersembahkan untuk kedua orang tuaku tercinta yang selalu menjadi sumber doa, semangat, dan cinta tanpa syarat dalam setiap langkah perjuanganku. Terima kasih atas segala pengorbanan, dukungan, dan kepercayaan yang tak pernah pudar. Tugas akhir ini juga kupersembahkan untuk diriku sendiri, sebagai bentuk apresiasi atas keteguhan hati, kerja keras, dan tekad yang tak pernah padam dalam menghadapi setiap rintangan hingga akhirnya mampu sampai di titik ini.

KATA PENGANTAR

Dengan menyebut nama Allah Subhanahu Wa Ta'ala, Yang Maha Pengasih lagi Maha Penyayang, penulis menyampaikan rasa syukur yang mendalam atas limpahan rahmat, taufik, dan hidayah-Nya, sehingga penulis dapat menyelesaikan tugas akhir dengan judul "Rancang Bangun Alat Analisis *Maintainability Index* dan Deteksi *Code Smells* pada Bahasa Pemrograman PHP Berbasis *Software Metrics*".

Shalawat serta salam semoga senantiasa tercurah kepada Nabi Muhammad SAW, suri teladan yang agung, yang telah membawa umat manusia dari masa kejahiliyahan menuju era ilmu pengetahuan dan peradaban. Semoga kita semua termasuk golongan yang mendapatkan syafaat beliau di hari akhir.

Tugas akhir ini merupakan hasil dari proses yang panjang dan menantang, yang menguji ketekunan, konsistensi, serta semangat belajar penulis. Segala pencapaian dalam proses ini tidak lepas dari bimbingan, doa, dan dukungan banyak pihak yang telah memberikan kontribusi berarti, baik secara langsung maupun tidak langsung. Untuk itu, penulis menyampaikan rasa terima kasih yang sebesar-besarnya kepada :

1. Prof. Noorhaidi, 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.
4. Bapak Dr. Ir. Aulia Faqih Rifa'i, M.Kom., selaku dosen penasihat akademik yang telah membimbing penulis selama masa studi.
5. Ibu Dr. Siti Mutmainah, S.Kom., M.Cs., selaku pembimbing tugas akhir yang dengan sabar memberikan arahan, masukan, dan dukungan dalam proses penyusunan tugas akhir ini.
6. Seluruh dosen dan staf Program Studi Informatika UIN Sunan Kalijaga yang telah memberikan ilmu dan bantuan selama masa perkuliahan.

7. Kedua orang tua tercinta, Ibu Sri Wahyuni dan Bapak Slamet Margiyono, atas segala doa, kasih sayang, serta dukungan yang tak pernah henti hingga penulis dapat menyelesaikan pendidikan tinggi ini.
8. Rekan-rekan seperjuangan Informatika angkatan 2021 yang telah menjadi bagian dari perjalanan akademik penulis, khususnya dalam memberikan semangat dan kebersamaan selama kuliah.

Penulis menyadari bahwa tugas akhir ini masih memiliki berbagai kekurangan, baik dari segi isi maupun penyajiannya. Oleh karena itu, penulis sangat mengharapkan kritik dan saran yang membangun guna perbaikan di masa yang akan datang. Semoga karya ini dapat memberikan manfaat dan menjadi dasar pengembangan lebih lanjut.

Yogyakarta, 21 Mei 2025

Penulis

Zildan Pandaru Sih Marginata

NIM. 21106050032

DAFTAR ISI

HALAMAN JUDUL i
LEMBAR PENGESAHAN ii
LEMBAR PERNYATAAN iii
LEMBAR PERSETUJUAN TUGAS AKHIR iv
INTISARI v
ABSTRACT ..... *vi*
MOTTO vii
HALAMAN PERSEMBAHAN viii
KATA PENGANTAR ix
DAFTAR ISI xi
DAFTAR TABEL xiv
DAFTAR GAMBAR xv
DAFTAR RUMUS xvii
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 3
1.3 Batasan Masalah 3
1.4 Tujuan 4
1.5 Manfaat 4
BAB II KAJIAN PUSTAKA 5
2.1 *Maintainability* 5
2.2 *Code Smells* 6
2.2.1 *Threshold* untuk Identifikasi *Code Smells* 8

2.3 Software Metrics.....8
2.3.1 *Maintainability Index*.....9
2.3.2 Chidamber dan Kemerer *Metrics*.....13
2.4 Penelitian Terdahulu.....15
BAB III METODE PENGEMBANGAN SISTEM.....20
3.1 Alat dan Bahan.....20
3.2 Waktu dan Tempat.....21
3.3 Metode Pengembangan.....21
3.3.1 Definisi.....21
3.3.2 Keunggulan.....21
3.3.3 Tahapan.....22
BAB IV PERANCANGAN DAN IMPLEMENTASI SISTEM.....28
4.1 Analisis Kebutuhan.....28
4.1.1 Hasil Wawancara Pengguna Potensial.....28
4.1.2 Kebutuhan Fungsional.....29
4.1.3 Kebutuhan Non-Fungsional.....31
4.2 Perancangan Sistem.....32
4.2.1 *Use Case Diagram*.....32
4.2.2 *Activity Diagram*.....33
4.2.3 *Sequence Diagram*.....36
4.2.4 Perancangan Antarmuka Pengguna.....39
4.3 Implementasi.....46
4.3.1 Hasil Implementasi.....47
4.4 Pengujian Sistem.....54
4.4.1 Pengujian *Black Box*.....54

4.4.2 Pengujian Akurasi 57
4.4.3 Pengujian Usabilitas 61
4.5 Hasil Pembahasan Studi Kasus Aplikasi Lapor Diri 63
4.5.1 Hasil *Maintainability Index* (MI) 63
4.5.2 Hasil Deteksi *Code Smells* 64
4.5.3 Pembahasan 65
4.6 Pemeliharaan 70
BAB V PENUTUP 71
5.1 Kesimpulan 71
5.2 Saran 72
DAFTAR PUSTAKA 74
LAMPIRAN 78

DAFTAR TABEL

Tabel 2.1 Penelitian terdahulu tentang *software metrics* untuk *maintainability* dan *code smells* 16
Tabel 4.1 Kebutuhan fungsional 29
Tabel 4.2 Kebutuhan non-fungsional 31
Tabel 4.3 *Use case* 33
Tabel 4.4 Pengujian *black box* 54
Tabel 4.5 Perbandingan pengujian akurasi MI 58
Tabel 4.6 Perbandingan pengujian akurasi WMC 58
Tabel 4.7 Perbandingan pengujian akurasi LCOM 59
Tabel 4.8 Perbandingan pengujian akurasi DIT 59
Tabel 4.9 Perbandingan pengujian akurasi NOC 59
Tabel 4.10 Akurasi deteksi code smells 59
Tabel 4.11 Pengujian usabilitas 62
Tabel 4.12 Hasil analisis MI 63

DAFTAR GAMBAR

Gambar 3.1 Tahapan metode *waterfall* 22
Gambar 4.1 Diagram *use case* 32
Gambar 4.2 *Activity diagram* untuk akses panduan.....34
Gambar 4.3 *Activity diagram* untuk akses tentang 35
Gambar 4.4 *Activity diagram* untuk proses analisis file 36
Gambar 4.5 *Sequence diagram* untuk akses panduan.....37
Gambar 4.6 *Sequence diagram* untuk akses tentang.....38
Gambar 4.7 *Sequence diagram* untuk proses analisis file 39
Gambar 4.8 Halaman utama.....40
Gambar 4.9 Tampilan saat tombol "*Analyze*" ditekan tanpa file 40
Gambar 4.10 Tampilan saat tombol "*Download* CSV" ditekan tanpa hasil analisis 41
Gambar 4.11 Tampilan setelah analisis file PHP berhasil (1/6).....42
Gambar 4.12 Tampilan setelah analisis file PHP berhasil (2/6).....42
Gambar 4.13 Tampilan setelah analisis file PHP berhasil (3/6).....43
Gambar 4.14 Tampilan setelah analisis file PHP berhasil (4/6).....43
Gambar 4.15 Tampilan setelah analisis file PHP berhasil (5/6).....43
Gambar 4.16 Tampilan setelah analisis file PHP berhasil (6/6).....44
Gambar 4.17 Tampilan modal panduan 45
Gambar 4.18 Tampilan modal tentang.....46
Gambar 4.19 Halaman utama.....47
Gambar 4.20 Tampilan modal panduan (1/2) 48
Gambar 4.21 Tampilan modal panduan (2/2) 48
Gambar 4.22 Tampilan modal tentang (1/2) 49
Gambar 4.23 Tampilan modal tentang (2/2) 49
Gambar 4.24 Tampilan grafik distribusi *Maintainability Index*.....50
Gambar 4. 25 Tampilan grafik *data class smell*.....50
Gambar 4.26 Tampilan grafik *parallel inheritance hierarchies smell*.....51
Gambar 4.27 Tampilan tabel analisis *Maintainability Index* (1/2) 52

Gambar 4.28 Tampilan tabel analisis *Maintainability Index* (2/2) 52
Gambar 4.29 Tampilan tabel deteksi *data class smell* 53
Gambar 4.30 Tampilan tabel deteksi *parallel inheritance hierarchies smell* 53
Gambar 4.31 Tampilan hasil distribusi MI aplikasi lapor diri 64
Gambar 4.32 Tampilan grafik data class aplikasi lapor diri.....65
Gambar 4.33 Tampilan grafik *parallel inheritance hierarchies* aplikasi lapor diri
.....65

DAFTAR RUMUS

Rumus (1).....9
Rumus (2).....11
Rumus (3).....11
Rumus (4).....15

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pada siklus pengembangan perangkat lunak, terdapat tahap pemeliharan. Tahap pemeliharan merupakan tahapan dimana sistem perangkat lunak dilakukan perbaikan agar dapat terhindar dari kesalahan yang belum terdeteksi pada tahap pengembangan dan pengujian sebelumnya. Terdapat beberapa perubahan yang dapat dilakukan pada perangkat lunak seperti perubahan sederhana untuk memperbaiki kesalahan pengkodean sistem, perubahan yang luas untuk memperbaiki kesalahan pada desain sistem, atau perubahan besar yang dilakukan untuk memperbaiki ketidaksesuaian dalam spesifikasi atau menyesuaikan dengan kebutuhan baru yang muncul [1].

Kemampuan sistem perangkat lunak untuk menerima dan mengatasi perubahan yang terjadi selama tahap pemeliharaan pada siklus pengembangan perangkat lunak disebut sebagai *maintainability*. Dalam proses pengembangan perangkat lunak, penting untuk menjaga kualitas kode supaya perangkat lunak dapat dengan mudah dipelihara, dimodifikasi, dan digunakan dalam jangka waktu yang lama. ISO/IEC 9126 menjelaskan bahwa *maintainability* merujuk pada sejauh mana kemampuan sistem perangkat lunak dapat diperbarui atau diubah. Selain itu, *maintainability* dapat juga didefinisikan sebagai kemudahan mengubah sistem atau komponen perangkat lunak untuk mengatasi kesalahan, meningkatkan kinerja atau fitur lainnya, dan menyesuaikan dengan perubahan lingkungan [2]. Tingkat *maintainability* yang semakin tinggi pada sistem perangkat lunak akan memudahkan pengembang dalam memodifikasi atau mengembangkan sistem, seperti menambahkan fitur tanpa menghadapi banyak kesulitan. Sedangkan, tingkat *maintainability* yang rendah dapat meningkatkan biaya pemeliharaan, meningkatkan risiko kesalahan, dan menurunkan efisiensi pengembangan perangkat lunak [3].

Pengembang perangkat lunak biasanya sering menghadapi tantangan dalam menjaga *maintainability*, terutama ketika ukuran kode program meningkat. *Code smells* seringkali menjadi penyebab menurunnya tingkat *maintainability* perangkat

lunak. Istilah ini bisa diartikan sebagai indikasi keberadaan masalah dalam struktur kode yang dapat memengaruhi kualitas dan menyulitkan proses pemeliharaan kode. Dalam praktik pengembangan perangkat lunak, *code smells* masih sering tidak diperhatikan karena dianggap bukan kesalahan yang memengaruhi fungsionalitas sistem [4].

Untuk mengatasi masalah terkait pemeliharaan, pendekatan yang dapat digunakan adalah penerapan metrik perangkat lunak untuk mengukur seberapa baik kualitas kode program secara kuantitatif, salah satunya adalah *Maintainability Index* (MI). MI merupakan metrik yang menggabungkan beberapa metrik kode sumber tradisional ke dalam satu nilai yang menunjukkan kemudahan pemeliharaan secara relative [5]. Selain itu, terdapat juga metrik seperti Chidamber dan Kemerer *metrics* yang banyak digunakan untuk mengidentifikasi dan mengenali adanya *code smells* di kode program sistem. CK *Metrics* merupakan sekumpulan metrik desain berorientasi objek yang diperkenalkan oleh Chidamber dan Kemerer untuk membantu memahami kompleksitas desain sistem berorientasi objek serta memprediksi kualitas eksternal perangkat lunak seperti cacat perangkat lunak, pengujian, dan upaya pemeliharaan [6]. Hal ini dapat memberikan gambaran lebih jelas terkait bagian kode program yang membutuhkan lebih banyak perhatian.

Saat ini, aplikasi yang mengintegrasikan analisis nilai *Maintainability Index* (MI) dan deteksi *code smells* masih jarang ditemukan di pasaran, padahal ini dapat memberikan keunggulan dalam proses pengembangan sistem perangkat lunak. Adanya alat yang mengintegrasikan kedua hal ini akan memberikan wawasan yang lebih jelas kepada pengembang perangkat lunak tentang kualitas kode program mereka dan mempercepat mereka dalam menganalisis setiap kode program dengan mempertimbangkan nilai *Maintainability Index*, serta mengidentifikasi bagian-bagian kode yang memiliki masalah atau *code smells*. Dengan demikian, proses *refactoring* kode bisa menjadi lebih efisien dan kualitas perangkat lunak pun dapat meningkat dengan lebih efektif.

Tugas akhir ini bertujuan untuk merancang dan mengembangkan alat yang mampu menghitung nilai *Maintainability Index* (MI) dan mendeteksi *code smells* seperti *data class* dan *parallel inheritance hierarchies* dalam kode PHP berbasis

OOP (*Object Oriented Programming*). Alat ini akan menggunakan berbagai metrik perangkat lunak untuk menghasilkan analisis yang komprehensif terkait kualitas kode program PHP, hal ini dapat memberikan gambaran yang lebih jelas kepada pengembang tentang bagian-bagian kode yang memerlukan perhatian lebih. Alat ini diharapkan dapat menjadi solusi praktis dan efektif untuk membantu pengembang meningkatkan produktivitas dalam menjaga kualitas perangkat lunak.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, rumusan masalah dalam penelitian ini adalah: "Bagaimana merancang dan membangun sebuah alat yang dapat menghitung *Maintainability Index* (MI) serta mendeteksi *code smells* dalam kode PHP guna membantu pengembang dalam meningkatkan *maintainability* perangkat lunak?"

1.3 Batasan Masalah

Agar penelitian ini lebih terfokus dan dapat diselesaikan dengan optimal sesuai dengan waktu dan sumber daya yang tersedia, maka ditetapkan beberapa batasan masalah sebagai berikut:

1. Ruang lingkup analisis kode yang dilakukan sistem hanya berlaku pada kode program PHP yang ditulis dengan paradigma *Object Oriented Programming* (OOP). Sistem tidak dirancang untuk menganalisis kode PHP yang seluruhnya menggunakan pendekatan prosedural atau fungsional.
2. Jenis *code smells* yang dianalisis dalam penelitian ini terbatas pada dua jenis, yaitu *Data Class* dan *Parallel Inheritance Hierarchy*. Pemilihan kedua jenis tersebut didasarkan pada kesesuaian dengan metrik CK (Chidamber & Kemerer) yang digunakan dalam sistem, serta mempertimbangkan keterbatasan waktu dan ruang lingkup pengembangan. Kedua *code smells* ini dipilih karena relatif sering ditemukan dalam pengembangan aplikasi berbasis PHP, khususnya yang menggunakan paradigma OOP, dan telah dianggap cukup representatif untuk memberikan gambaran terhadap *maintainability* suatu kode program.
3. Aplikasi dibangun dalam bentuk web berbasis React.js dan Node.js, dan telah dideploy menggunakan platform Vercel. Namun, sistem yang telah dideploy

secara publik memiliki keterbatasan dalam hal ukuran file unggahan, yaitu maksimal 5 MB, sesuai dengan kebijakan platform Vercel. Oleh karena itu, analisis terhadap proyek berskala besar dilakukan secara lokal di lingkungan pengembangan.

4. File yang dapat dianalisis oleh sistem harus dikompresi dalam format .zip dan hanya berisi file sumber kode dengan ekstensi .php. Sistem tidak melakukan analisis terhadap file non-PHP yang mungkin disertakan dalam proyek.

1.4 Tujuan

Berdasarkan rumusan masalah yang telah disebutkan, penelitian ini bertujuan untuk merancang dan membangun alat yang dapat menghitung *Maintainability Index* (MI) serta mendeteksi *code smells* seperti *data class* dan *parallel inheritance hierarchies* dalam kode PHP untuk membantu pengembang meningkatkan *maintainability* perangkat lunak.

1.5 Manfaat

Penelitian dan pengembangan alat ini memberikan berbagai manfaat bagi pengembang perangkat lunak, terutama dalam meningkatkan kualitas dan *maintainability* kode. Berikut manfaat utama yang diharapkan dari alat ini meliputi:

1. Dapat mengukur nilai *Maintainability Index (MI)* dan mendeteksi code smells secara otomatis sehingga diharapkan dapat meningkatkan *maintainability* perangkat lunak.
2. Mengurangi biaya dan waktu pemeliharaan dengan mengidentifikasi potensi masalah dalam kode lebih cepat.
3. Meningkatkan efisiensi pengembangan dengan laporan yang jelas dan mudah dimengerti.
4. Menjaga kualitas perangkat lunak dalam jangka waktu yang lama melalui deteksi masalah struktur kode lebih awal.

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan hasil implementasi dan pengujian sistem, alat analisis *Maintainability Index* (MI) dan deteksi *code smells* berbasis CK *Metrics* yang dikembangkan menggunakan metode *Waterfall* menunjukkan kinerja yang sangat baik, khususnya pada file PHP dengan struktur sederhana. Akurasi perhitungan metrik seperti MI, WMC, LCOM, DIT, dan NOC berada dalam tingkat kesalahan yang sangat kecil. Sistem juga mampu mendeteksi *code smells* seperti *Data Class* dan *Parallel Inheritance Hierarchies* secara tepat, sehingga dapat memberikan informasi yang relevan bagi pengembang dalam upaya peningkatan kualitas kode.

Sistem kemudian diuji dari sisi *usability* menggunakan metode *System Usability Scale* (SUS) yang melibatkan 26 responden dengan tingkat pengalaman berbeda dalam pemrograman PHP. Hasil pengujian menunjukkan skor SUS sebesar 80, yang berada dalam kategori "*Good*" dan melampaui ambang batas minimum sebesar 68. Hal ini menandakan bahwa sistem dinilai mudah digunakan dan nyaman, serta antarmukanya dirancang dengan baik sehingga dapat dioperasikan dengan lancar oleh pengguna dari berbagai latar belakang keahlian.

Pengujian lebih lanjut dilakukan menggunakan studi kasus Aplikasi Lapor Diri Prodi PPG yang memiliki struktur kode kompleks dan berskala besar. Hasil menunjukkan bahwa sistem berhasil memberikan gambaran kualitas kode secara kuantitatif, dengan temuan bahwa 35,5% file memiliki MI rendah, serta terdeteksi adanya *code smells* pada sejumlah file, khususnya *Parallel Inheritance Hierarchies* (227 file) dan *Data Class* (38 file). Meski sebagian besar kode masih tergolong *maintainable*, hasil ini mengindikasikan adanya bagian-bagian yang memerlukan *refactoring* atau perbaikan desain. Studi kasus ini membuktikan bahwa sistem yang dikembangkan mampu digunakan dalam konteks nyata dan kompleks, serta memberikan kontribusi nyata dalam proses evaluasi dan peningkatan kualitas kode perangkat lunak.

5.2 Saran

Meskipun alat ini sudah berfungsi dengan baik, beberapa saran untuk pengembangan lebih lanjut adalah sebagai berikut:

1. Peningkatan responsif untuk perangkat *mobile*

   Saat ini, sistem hanya dioptimalkan untuk perangkat komputer (PC/laptop). Sebaiknya sistem dikembangkan agar dapat diakses dengan baik melalui perangkat mobile, untuk meningkatkan kenyamanan pengguna yang lebih sering menggunakan perangkat tersebut.

2. Penambahan fitur deteksi *code smells* lainnya

   Meskipun alat ini sudah mampu mendeteksi *data class* dan *parallel inheritance hierarchies*, menambahkan deteksi *code smells* lainnya, seperti *god class* atau *feature envy*, dapat meningkatkan kegunaan alat ini dalam berbagai konteks pengembangan perangkat lunak.

3. Penggunaan *library parsing* selain php-parser di node.js

   Library *php-parser* yang digunakan saat ini memiliki keterbatasan dari sisi performa, khususnya saat menangani proyek PHP yang kompleks atau berukuran besar. Oleh karena itu, disarankan untuk mempertimbangkan penggunaan *library parsing* PHP alternatif yang mampu memberikan kinerja parsing yang lebih efisien dan akurat.

4. Optimalisasi waktu proses untuk file lebih besar

   Pengolahan file dengan ukuran lebih besar perlu dioptimalkan lebih lanjut agar waktu respons tetap efisien.

5. Restrukturisasi kode pada aplikasi lapor diri

   Meskipun struktur kode di aplikasi lapor diri sudah cukup baik, dalam satu file terdapat banyak sekali fungsi dan baris kode yang dapat menyulitkan pengelolaan seiring bertambahnya fungsionalitas aplikasi. Disarankan untuk menerapkan struktur *Controller-Service-Repository* guna meningkatkan *maintainability*. Dengan memisahkan tanggung jawab antara *Controller* (untuk menangani *request*), *Service* (untuk logika bisnis), dan *Repository* (untuk interaksi dengan API atau *database*), aplikasi akan menjadi lebih terstruktur, mudah dipelihara, diuji, serta dikembangkan di masa depan. Struktur ini juga

memungkinkan kode menjadi lebih modular, mengurangi duplikasi, dan meningkatkan keterbacaan serta fleksibilitas.

DAFTAR PUSTAKA

[1] I. Sommerville, *Software Engineering*, 9th ed. Boston, MA: Pearson Education, Inc., 2011.

[2] B. Seref and O. Tanriover, “Software Code Maintainability: A Literature Review,” *International Journal of Software Engineering & Applications (IJSEA)*, vol. 7, no. 3, pp. 69-87, May 2016

[3] R. Andriani, P. N. Sabrina, and H. Ashaury, “Metode Refactoring Untuk Meningkatkan Kualitas Maintainability Pada Sistem Informasi Puskesmas,” *Jurnal TEKNO KOMPAK*, vol. 19, no. 1, pp. 227-239, Oct. 2024.

[4] H. K. Putra, B. Priyambadha, dan D. Pramono, "Pengembangan Sistem Aplikasi Pendeteksi Long Method Smell Berdasarkan Refactoring Filtering Metrics," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, vol. 3, no. 7, pp. 7143–7149, Jul. 2019. [Online]. Available: http://j-ptiik.ub.ac.id

[5] N. M. A. M. Najm, "Measuring Maintainability Index of a Software Depending on Line of Code Only," *IOSR Journal of Computer Engineering (IOSR-JCE)*, vol. 16, no. 2, pp. 64–69, Mar.-Apr. 2014.

[6] P. S. Sandhu and H. Singh, “A critical suggestive evaluation of CK metric,” presented at the Pacific Asia Conf. on Inf. Syst. (PACIS 2005), Bangkok, Thailand, Jul. 7–10, 2005.

[7] D. Coleman, B. Lowther, and P. Oman, “The Application of Software Maintainability Models in Industrial Software Systems,” *Journal of Systems and Software*, vol. 29, pp. 3-16, Apr. 1995.

[8] R. Malhotra and A. Chug, “Software Maintainability: Systematic Literature Review and Current Trends,” *International Journal of Software Engineering and Knowledge Engineering*, vol. 26, no. 8, pp. 1221–1253, 2016.

[9] I. Heitlager, T. Kuipers, and J. Visser, “A Practical Model for Measuring Maintainability,” in *Proc. 6th Int. Conf. Quality of Information and Communications Technology (QUATIC)*, Lisbon, Portugal, Sep. 2007, doi: 10.1109/QUATIC.2007.8.

[10] S. F. Sujadi, “Evaluasi deteksi smell code dan anti pattern pada aplikasi berbasis Java,” *Jurnal Teknik Informatika dan Sistem Informasi*, vol. 5, no. 3, pp. 370-385, Dec. 2019.

[11] M. Agnihotri and A. Chug, “A Systematic Literature Survey of Software Metrics, Code Smells and Refactoring Techniques,” *Journal of Information Processing Systems*, vol. 16, no. 4, pp. 915–934, Aug. 2020.

[12] M. Ilyas, S. Razzaq, F. Maqbool, W. Ahmed, and S. M. Adnan, “Code Smell Detection and Refactoring Using AST Visitor,” *Technical Journal, University of Engineering and Technology (UET) Taxila, Pakistan*, vol. 25, no. 1, pp. 59-65, 2020.

[13] M. Alharthi and W. Aljedaibi, "Theoretical Validation of Inheritance Metric in QMOOD against Weyuker’s Properties," *IJCSNS Int. J. Comput. Sci. Netw. Secur.*, vol. 21, no. 7, pp. 284–296, Jul. 2021.

[14] P. Danphitsanuphan and T. Suwantada, “Code Smell Detecting Tool and Code Smell-Structure Bug Relationship,” in *Proc. SCET Conf.*, Bangkok, Thailand, May 2012, doi: 10.1109/SCET.2012.6342082.

[15] R. G. Atmaja, B. Priyambadha, and F. Pradana, “Pembangunan kakas bantu untuk mengukur maintainability index pada perangkat lunak berdasarkan nilai Halstead metrics dan McCabe's cyclomatic complexity,” *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, vol. 3, no. 3, pp. 2167–2172, Mar. 2019.

[16] T. Hariprasad, K. Seenu, G. Vidhyagaran, and C. Thirumalai, “Software Complexity Analysis Using Halstead Metrics,” in *Proceedings of the International Conference on Trends in Electronics and Informatics (ICEI 2017)*, Tirunelveli, India, May 2017, pp. 1109–1112, doi: 10.1109/ICOEI.2017.8300883.

[17] D. R. Wijendra and K. P. Hewagamage, “Analysis of Cognitive Complexity with Cyclomatic Complexity Metric of Software,” *International Journal of Computer Applications*, vol. 174, no. 19, pp. 14–19, Feb. 2021.

[18] T. Hericko and B. Sumak, “Exploring Maintainability Index Variants for Software Maintainability Measurement in Object-Oriented Systems,” *Appl. Sci.*, pp. 1–38, Feb. 2023, doi: 10.3390/app13052972.

[19] S. Prykhodko, N. Prykhodko, and T. Smykodub, “A Statistical Evaluation of the Depth of Inheritance Tree Metric for Open-Source Applications Developed in Java,” *Foundations of Computing and Decisions Sciences*, vol. 46, no. 2, pp. 160–172, 2021.

[20] S. R. Chidamber and C. F. Kemerer, “A Metric Suite for Object Oriented Design*,” IEEE Transactions on Software Engineering*, vol. 20, pp. 476–493, 1994.

[21] B. M. Goel and P. K. Bhatia, “Analysis of Reusability of Object-Oriented System using CK Metrics,” *International Journal of Computer Applications*, vol. 60, no. 10, pp. 32-36, Dec. 2012.

[22] A. Ganpati, A. Kalia, and H. Singh, “Maintainability Index Over Multiple Releases: A Case Study PHP Open Source Software,” *International Journal of Engineering Research & Technology (IJERT)*, vol. 1, no. 6, pp. 1–3, Aug. 2012.

[23] Z. Tóth, "Applying and Evaluating Halstead’s Complexity Metrics and Maintainability Index for RPG," in *Lecture Notes in Computer Science*, pp. 1–16, Jul. 2017.

[24] R. Subramanyam and M. S. Krishnan, “Empirical Analysis of CK Metrics for Object-Oriented Design Complexity: Implications for Software Defects,” *IEEE Transactions on Software Engineering*, vol. 29, no. 4, pp. 297–309, Apr. 2003.

[25] S. Arshad and C. Tjortjis, “Clustering Software Metric Values Extracted from C# Code for Maintainability Assessment,” in *Proceedings of the 9th Hellenic Conference on Computer and Machine Engineering*, May 2016, doi: 10.1145/2903220.2903252.

[26] N. W. J. K. Dewi, I. G. M. Y. Antara, and D. Sucahyono, “Application of The Waterfall Method to the Website-Based JM Leather & Shoes Point of Sales

Information System," *TIERS Information Technology Journal*, vol. 5, no. 1, pp. 1–12, Jun. 2024.

[27] B. A. Andrei, A. C. Casu-Pop, S. C. Gheorghe, and C. A. Boiangiu, "A Study on Using Waterfall and Agile Methods in Software Project Management," *Journal of Information Systems & Operations Management*, pp. 125–135, May 2019.

[28] A. A. Wahid, "Analisis Metode Waterfall untuk Pengembangan Sistem Informasi," *Jurnal Ilmu-ilmu Informatika dan Manajemen STMIK*, Oct. 2020.