RANCANG BANGUN APLIKASI GENERATOR DOKUMENTASI PERANGKAT LUNAK UNTUK PROYEK BERBAHASA JAVASCRIPT

TUGAS AKHIR

Untuk Memenuhi Sebagian Persyaratan Mencapai Derajat Sarjana S-1
Program Studi Informatika

Disusun Oleh:
Qonita Nadya Ramadhani
22106050004

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

LEMBAR PERNYATAAN KEASLIAN

Yang bertanda tangan di bawah ini:

Nama	: Qonita Nadya Ramadhani
NIM	: 22106050004
Program Studi	: Informatika
Fakultas	: Sains dan Teknologi

Menyatakan dengan sesungguhnya, bahwa tugas akhir saya yang berjudul **"Rancang Bangun Aplikasi Generator Dokumentasi Perangkat Lunak Untuk Proyek Berbahasa JavaScript"** merupakan hasil tugas akhir 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.

Yogyakarta, 17 Februari 2026

Penulis,

Qonita Nadya Ramadhani

NIM. 22106050004

LEMBAR PERSETUJUAN SKRIPSI/TUGAS AKHIR

Hal : Persetujuan Skripsi/Tugas Akhir
Lamp : -

Kepada
Yth.
Dekan Fakultas Sains dan Teknologi
Universitas Islam Negeri 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	:	Qonita Nadya Ramadhani
NIM	:	22106050004
Judul Skripsi	:	Rancang Bangun Aplikasi Generator Dokumentasi Perangkat Lunak Untuk Proyek Berbahasa JavaScript

Sudah dapat diajukan kepada Program Studi Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri 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, 17 Februari 2026
Pembimbing,

Dr. Agung Fatwanto, S.Si., M.Kom.
NIP. 19770103 200501 1 003

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-513/Un.02/DST/PP.00.9/03/2026

Tugas Akhir dengan judul : Rancang Bangun Aplikasi Generator Dokumentasi Perangkat Lunak Untuk Proyek Berbahasa JavaScript

yang dipersiapkan dan disusun oleh:

Nama : QONITA NADYA RAMADHANI
Nomor Induk Mahasiswa : 22106050004
Telah diujikan pada : Senin, 02 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

Dr. Agung Fatwanto, S.Si., M.Kom.
SIGNED

Valid ID: 69aa4a5c70cc8

Penguji I

Dr. Agus Mulyanto, S.Si., M.Kom., ASEAN Eng.
SIGNED

Valid ID: 69aa479748ae9

Penguji II

Muhammad Galih Wonoseto, M.T.
SIGNED

Valid ID: 69a8f2c48e574

Yogyakarta, 02 Maret 2026
UIN Sunan Kalijaga
Dekan Fakultas Sains dan Teknologi

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

Valid ID: 69aa584821d17

LEMBAR PEDOMAN PENGGUNAAN TUGAS AKHIR

Tugas Akhir ini tidak dipublikasikan, tetapi tersedia di perpustakaan dalam lingkungan Universitas Islam Negeri 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 Universitas Islam Negeri Sunan Kalijaga Yogyakarta

HALAMAN PERSEMBAHAN

Dengan penuh rasa syukur, tugas akhir ini penulis persembahkan kepada Allah Subhanahu Wata'ala atas segala rahmat, karunia, dan kemudahan yang telah diberikan kepada penulis sehingga dapat menyelesaikan tugas akhir ini dengan baik.

Tugas Akhir ini penulis persembahkan kepada kedua orang tua penulis, khususnya ibu, yang dengan kasih sayang, doa, dan pengorbanan tanpa henti telah membesarkan serta mendidik penulis sejak kecil, senantiasa mengupayakan yang terbaik, serta menjadi sumber kekuatan utama bagi penulis dalam menempuh pendidikan hingga ke jenjang perguruan tinggi.

Selanjutnya, penulis mempersembahkan tugas akhir ini kepada kakak dan adik penulis, yang dengan peran dan caranya masing-masing telah memberikan dukungan, motivasi, doa, serta kebersamaan yang menguatkan penulis dalam menghadapi setiap proses dan tantangan selama perkuliahan dan penyusunan tugas akhir ini.

Tugas Akhir ini juga penulis persembahkan kepada Adi Wibowo, yang telah setia menemani, memberikan dukungan, perhatian, serta menjadi pendengar yang baik dalam setiap dinamika yang penulis alami selama proses penyusunan tugas akhir ini.

Terakhir, tugas akhir ini penulis persembahkan kepada teman-teman Program Studi Informatika angkatan 2022, khususnya Alya Azzahra dan Yusrina Matura, sebagai rekan seperjuangan yang senantiasa saling mendukung, berbagi cerita, dan membersamai penulis sejak awal hingga akhir masa perkuliahan.

Terima kasih yang tak terhingga atas segala dukungan yang telah kalian berikan. Semoga Allah senantiasa melimpahkan rahmat dan berkah-Nya, serta memberikan balasan terbaik untuk semua kebaikan yang telah diberikan.

MOTTO

"Jika kamu berbuat baik, (sebenarnya) kebaikan itu untuk dirimu sendiri"
- QS. Al-Isra: 7 -

"*Jika ada yang harus diingatkan ya pasti hanyalah diriku, bahwa banyak hal yang harus dipikirkan selain cinta melulu. Berhenti mengira hanya aku yang paling pantas untuk mengeluh. Semua kepingan baik akan datang tapi mereka perlukan waktu. Sabar~*"
- Lomba Sihir -

"Ada waktu-waktu, hal buruk datang berturut-turut. Semua yang tinggal, juga yang hilang. Seberapa pun absurdnya pasti ada makna"
- Bernadya -

"So if you need a hero, just look in the mirror. No one's gonna save you now, so you better save yourself"
- Kali Uchis -

KATA PENGANTAR

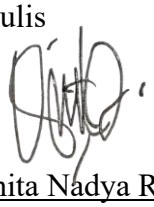
Alhamdulillahi Rabbil ‘Alamin, puji dan syukur penulis panjatkan ke hadirat Allah Subhanahu Wata’ala atas segala rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul “Rancang Bangun Aplikasi Generator Dokumentasi Perangkat Lunak Untuk Proyek Berbahasa JavaScript” dengan baik. Shalawat serta salam senantiasa tercurah kepada junjungan kita Nabi Muhammad Shallallahu ‘Alaihi Wasallam, yang telah membawa umat manusia dari zaman kegelapan menuju zaman yang penuh dengan ilmu pengetahuan dan cahaya keimanan.

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 tugas akhir ini, penulis menyadari bahwa keberhasilan penyelesaian karya ini tidak terlepas dari bantuan, bimbingan, serta dukungan dari berbagai pihak, baik secara langsung maupun tidak langsung. Oleh karena itu, pada kesempatan ini penulis menyampaikan rasa terima kasih yang sebesar-besarnya kepada:

1. Bapak Prof. Noorhaidi, 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 Universitas Islam Negeri Sunan Kalijaga Yogyakarta.
3. Bapak Dr. Muhammad Mustakim, S.T., M.T., selaku Ketua Program Studi Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Sunan Kalijaga Yogyakarta.
4. Bapak Dr. Agus Mulyanto, S.Si., M.Kom., selaku Dosen Penasihat Akademik yang telah memberikan arahan dan bimbingan kepada penulis selama menempuh perkuliahan.
5. Bapak Dr. Agung Fatwanto, S.Si., M.Kom., ASEAN Eng., selaku dosen pembimbing tugas akhir yang telah meluangkan waktu, memberikan masukan yang berharga, serta dengan sabar membimbing dan mengarahkan penulis dalam setiap tahapan penyusunan tugas akhir ini.
6. Seluruh dosen dan karyawan Program Studi Informatika Universitas Islam Negeri Sunan Kalijaga Yogyakarta yang telah memberikan ilmu, pengalaman, serta bantuan selama masa perkuliahan penulis.

Penulis menyadari bahwa tugas akhir ini masih memiliki keterbatasan dan kekurangan, baik dari segi penyajian maupun substansi pembahasan. Oleh karena itu, penulis sangat mengharapkan kritik dan saran yang bersifat membangun demi penyempurnaan karya ini di masa mendatang. Semoga tugas akhir ini dapat memberikan manfaat bagi pengembangan ilmu pengetahuan serta dapat dijadikan sebagai referensi untuk pengembangan selanjutnya.

Yogyakarta, 17 Februari 2026
Penulis



Qonita Nadya Ramadhani
NIM. 22106050004

INTISARI

Pembuatan dokumentasi kode program sering terabaikan dalam pengembangan proyek JavaScript karena proses manual yang tidak efisien. Penelitian ini bertujuan untuk menjawab bagaimana merancang dan membangun aplikasi generator dokumentasi otomatis berbasis *Command Line Interface* (CLI) untuk proyek JavaScript yang efektif. Fokus utama penelitian mencakup cara mendeteksi dan mengambil isi komentar JSDoc dari dalam kode program secara otomatis menggunakan analisis statis, serta bagaimana mengonversi data komentar tersebut menjadi file dokumen yang rapi dan terstruktur dalam format Markdown dan HTML tanpa ketergantungan pada antarmuka grafis yang berat.

Metode pengembangan sistem yang diterapkan dalam penelitian ini adalah *Prototyping*. Proses dimulai dengan tahapan *Requirements Gathering* untuk mengidentifikasi kebutuhan pengguna, dilanjutkan dengan *Quick Design* dan *Build Prototype* untuk menyusun kerangka dasar aplikasi. Prototipe yang dihasilkan kemudian melalui tahap *User Evaluation* untuk mendapatkan umpan balik, yang digunakan sebagai dasar dalam *Refining Prototype* guna menyempurnakan fitur dan penanganan kesalahan. Tahapan diakhiri dengan *Engineering Product* untuk mematangkan sistem menjadi perangkat lunak yang siap didistribusikan dan digunakan secara luas.

Hasil dari tugas akhir ini adalah aplikasi bernama my-docs-gen-nita berbasis Node.js yang telah didistribusikan melalui NPM Registry dan terbukti mampu berjalan lintas sistem operasi (Windows dan Linux) untuk pengelolaan dokumentasi via terminal. Penerapan metode *Abstract Syntax Tree* (AST) menggunakan pustaka *acorn* dan *estraverse* berhasil diimplementasikan untuk membaca struktur kode secara mendalam, mengekstraksi berbagai tag JSDoc (seperti *@param*, *@returns*, *@example*) secara dinamis, serta memiliki ketahanan (*robustness*) dalam menangani variasi penulisan komentar. Sistem berhasil menyajikan luaran berformat Markdown (*.md*) dan HTML (*.html*) secara terstruktur, didukung oleh hasil *User Acceptance Testing* (UAT) sebagai konfirmasi atas keberhasilan fungsionalitas serta kemudahan penggunaan perangkat lunak.

Kata kunci: Dokumentasi Otomatis, *Command Line Interface* (CLI), JavaScript, JSDoc, *Abstract Syntax Tree* (AST), *Prototyping*, my-docs-gen-nita.

ABSTRACT

The creation of program code documentation is often neglected in JavaScript project development due to inefficient manual processes. This research aims to address how to effectively design and build an automatic documentation generator application based on a Command Line Interface (CLI) for JavaScript projects. The main focus of this research includes how to automatically detect and extract JSDoc comment content from program code using static analysis, as well as how to convert said comment data into neat and structured document files in Markdown and HTML formats without relying on heavy graphical interfaces.

The system development method applied in this research is Prototyping. The process begins with the Requirements Gathering stage to identify user needs, followed by Quick Design and Build Prototype to structure the application framework. The resulting prototype then undergoes the User Evaluation stage to gather feedback, which is used as a basis for the Refining Prototype stage to improve features and error handling. The process concludes with the Engineering Product stage to mature the system into software ready for distribution and widespread use.

The result of this final project is a Node.js-based application named my-docs-gen-nita, which has been distributed via the NPM Registry and is proven capable of running across operating systems (Windows and Linux) for efficient documentation management via the terminal. The application of the Abstract Syntax Tree (AST) method using the acorn and estraverse libraries was successfully implemented to deeply read code structure, dynamically extracting various JSDoc tags (such as @param, @returns, and @example), and possessing robustness in handling variations in comment writing styles. The system successfully produces output in neat Markdown (.md) and HTML (.html) formats, with User Acceptance Testing (UAT) results showing an average usability score of 4.6 out of 5, confirming the functionality and ease of use of this software.

***Keywords**: Automatic Documentation, Command Line Interface (CLI), JavaScript, JSDoc, Abstract Syntax Tree (AST), Prototyping, my-docs-gen-nita.*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang Permasalahan

Dalam penerapan metode rekayasa perangkat lunak, fokus utama tidak hanya terletak pada pembuatan kode program, melainkan pada penciptaan sistem yang andal, terukur, dan berkualitas tinggi. Siklus hidup pengembangan perangkat lunak mencakup serangkaian proses kompleks mulai dari perencanaan, pengembangan, pengujian, operasi, hingga pemeliharaan yang berkelanjutan. Agar setiap tahapan tersebut dapat berjalan sistematis dan dievaluasi, diperlukan transfer pengetahuan yang efektif antar pengembang. Dalam konteks inilah, keberlangsungan dan kualitas proses pemeliharaan kode menjadi sangat bergantung pada keberadaan dokumentasi kode yang baik [1].

Dalam rekayasa perangkat lunak, dokumentasi kode merupakan elemen yang krusial karena berfungsi sebagai sumber pengetahuan utama mengenai desain, fungsi, dan penggunaan suatu sistem, sehingga mempengaruhi kualitas dan keberlangsungan perangkat lunak itu sendiri. Dokumentasi yang baik secara langsung mendukung kegiatan pemeliharaan (*maintenance*) dengan mengurangi waktu yang dibutuhkan untuk memahami bagian-bagian sistem yang perlu diperbaiki atau dikembangkan, sehingga menurunkan biaya dan risiko kesalahan saat melakukan perubahan [2]. Selain itu, dokumentasi berperan penting dalam proses onboarding anggota tim baru. Dokumentasi yang tersusun rapi dan mudah diakses mempercepat kurva pembelajaran pengembang baru sehingga mereka lebih cepat produktif dalam tim [3]. Dengan demikian, dokumentasi juga meningkatkan *readability* (keterbacaan) kode, baik melalui komentar terstruktur, konvensi penamaan, maupun panduan arsitektur yang selanjutnya mempermudah kolaborasi antar pengembang dan pemeliharaan jangka panjang [4].

Namun pada prakteknya, proses dokumentasi kode seringkali menjadi aspek yang terabaikan atau dilakukan secara tidak konsisten. Faktor teknis seperti fokus yang berlebih pada refaktorisasi kode dan fungsionalitas fitur sering kali tidak diimbangi dengan pembaruan dokumentasi yang memadai. Hal tersebut merupakan fenomena yang sering disebut sebagai *documentation debt* [5]. Kurangnya standar sistematis menyebabkan pengembang cenderung mendokumentasikan perubahan secara terbatas, atau bahkan membiarkan dokumentasi menjadi usang (*outdated*) dan tidak sinkron dengan versi kode terbaru. Studi menunjukkan bahwa perubahan kode tanpa pembaruan komentar yang sesuai adalah penyebab utama kebingungan dalam pemeliharaan perangkat lunak [6]. Hal ini menegaskan bahwa tanpa alat bantu yang tepat, dokumentasi sering dianggap sebagai beban administratif tambahan yang menghambat kecepatan alur pengembangan perangkat lunak.

Untuk mengatasi kesenjangan antara kode dan dokumentasi tersebut, diperlukan pendekatan otomatisasi yang dapat mengintegrasikan proses dokumentasi langsung ke dalam alur kerja pengembangan perangkat lunak. Pendekatan *automated code documentation* kini menjadi solusi yang banyak diteliti untuk menggantikan atau melengkapi upaya dokumentasi manual yang memakan waktu dan sumber daya [7]. Salah satu metode yang efisien adalah *Documentation as Code*, di mana dokumentasi

ditulis menyatu dengan kode sumber [8] menggunakan standar seperti JSDoc. Namun, agar komentar JSDoc tersebut dapat memberikan manfaat yang lebih luas dan mudah dibaca oleh seluruh tim, diperlukan alat bantu yang mampu mengekstrak informasi tersebut secara cerdas. Oleh karena itu, tugas akhir ini mengusulkan pengembangan aplikasi berbasis *Command Line Interface* (CLI) yang secara otomatis membaca dan memproses (*parsing*) struktur kode JavaScript dan komentar JSDoc, kemudian mengkonversinya menjadi format dokumentasi eksternal yang terstruktur seperti HTML atau Markdown.

1.2 Rumusan Masalah

Berdasarkan latar belakang masalah yang telah diuraikan, maka rumusan masalah dalam tugas akhir ini adalah:

1. Bagaimana merancang dan membangun aplikasi generator dokumentasi otomatis berbasis *Command Line Interface* (CLI) untuk proyek JavaScript?
2. Bagaimana cara mendeteksi dan mengambil isi komentar JSDoc dari dalam kode program secara otomatis?
3. Bagaimana mengkonversi data komentar tersebut menjadi file dokumen yang rapi dalam format Markdown dan HTML?

1.3 Batasan Masalah

Agar tugas akhir ini lebih terarah dan tidak menyimpang dari tujuan utama, maka penulis menetapkan batasan masalah sebagai berikut:

1. Aplikasi dibangun berbasis Node.js dan dijalankan melalui antarmuka baris perintah (*Command Line Interface* / CLI), tanpa antarmuka grafis (*Graphical User Interface* / GUI).
2. Aplikasi hanya mendukung pemrosesan kode sumber yang ditulis dalam bahasa pemrograman JavaScript (`.js`).
3. Mekanisme ekstraksi dokumentasi menggunakan pendekatan "*Node-Attachment*", di mana komentar JSDoc yang diproses adalah yang melekat langsung pada entitas kode (seperti *Class*, *Function*, *Method*, dan *Variable*).
4. Aplikasi dirancang dengan sistem parsing dinamis yang mampu mengekstrak berbagai jenis tag JSDoc secara fleksibel (seperti `@param`, `@returns`, `@example`, `@author`, `@deprecated`, maupun tag kustom lainnya) sesuai dengan apa yang ditulis pengembang, serta mendukung fitur pengecualian menggunakan tag `@ignore`.
5. Keluaran (*output*) dokumentasi yang dihasilkan berupa file statis dengan format Markdown (`.md`) dan HTML (`.html`).
6. Tugas Akhir ini tidak mencakup pembuatan konten dokumentasi secara otomatis (seperti menggunakan AI/LLM), melainkan hanya mengekstrak konten yang sudah ditulis oleh pengembang di dalam kode.
7. Fitur pencarian (*search*) yang diimplementasikan pada dokumen keluaran (HTML) bersifat *Client-Side* dan berjalan sepenuhnya di peramban web (*browser*) menggunakan manipulasi DOM (*Document Object Model*), tanpa melibatkan

penyimpanan basis data (*database*) maupun pemrosesan di sisi peladen (*server-side*).

1.4 Tujuan Tugas Akhir

Berdasarkan rumusan masalah yang telah diuraikan, tujuan dari tugas akhir ini adalah:

1. Merancang dan membangun aplikasi generator dokumentasi otomatis berbasis *Command Line Interface* (CLI) menggunakan lingkungan runtime Node.js untuk proyek perangkat lunak berbahasa JavaScript.
2. Mengimplementasikan mekanisme parsing kode yang mampu membaca struktur kode dan mengekstrak berbagai jenis tag komentar JSDoc secara dinamis, seperti `@param`, `@returns`, `@example`, dan tag kustom lainnya untuk mendukung fleksibilitas dokumentasi.
3. Menghasilkan output dokumentasi eksternal yang terstruktur, rapi, dan mudah dibaca dalam format Markdown dan HTML guna meningkatkan keterbacaan dan aksesibilitas informasi proyek.

1.5 Manfaat Tugas Akhir

Hasil dari pengembangan aplikasi generator dokumentasi otomatis ini diharapkan dapat memberikan dampak dan kegunaan bagi berbagai pihak.

1.5.1 Manfaat bagi Pengembang Perangkat Lunak

Aplikasi yang dikembangkan dalam tugas akhir ini diharapkan dapat memberikan solusi praktis untuk meningkatkan efisiensi kerja *developer*. Adapun manfaat spesifik yang dapat dirasakan oleh pengembang perangkat lunak adalah:

1. Efisiensi Waktu: Mengotomatisasi proses pembuatan dokumentasi yang sebelumnya dilakukan secara manual, sehingga pengembang dapat lebih fokus pada penulisan logika kode.
2. Konsistensi dan Akurasi: Memastikan dokumentasi selalu sinkron dengan kondisi kode terbaru, meminimalkan risiko terjadinya *outdated documentation* atau informasi yang menyesatkan.
3. Kemudahan Kolaborasi: Mempermudah proses onboarding anggota tim baru dan pemeliharaan sistem (*maintenance*) melalui penyediaan dokumentasi yang terstandarisasi dan mudah diakses.
4. Kemudahan Navigasi: Mempercepat penelusuran informasi fungsi atau variabel spesifik melalui fitur pencarian interaktif dan antarmuka yang responsif, sehingga pengembang tidak perlu menggulir (*scroll*) dokumen secara manual.

1.5.2 Manfaat bagi Penulis

Pelaksanaan tugas akhir ini menjadi sarana bagi penulis untuk mengimplementasikan ilmu yang telah diperoleh selama masa perkuliahan. Secara pribadi, manfaat yang didapatkan penulis dalam proses ini meliputi:

1. Menerapkan wawasan teoritis mengenai Teori Bahasa dan Otomata serta Rekayasa Perangkat Lunak ke dalam studi kasus nyata, khususnya dalam pemrosesan struktur pohon sintaksis abstrak (*Abstract Syntax Tree* / AST).
2. Mengasah kemampuan analisis dan pemecahan masalah (*problem solving)* dalam merancang algoritma parsing yang efisien serta penanganan kasus tepi (*edge cases*) pada kode sumber yang kompleks.
3. Memberikan pengalaman praktis dalam mengelola siklus hidup pengembangan perangkat lunak secara menyeluruh, mulai dari perancangan arsitektur, implementasi kode, pengujian (*testing*), hingga distribusi paket aplikasi secara publik.
4. Menghasilkan produk perangkat lunak *open-source* yang dapat menjadi portofolio kompetensi teknis dalam menghadapi kebutuhan industri pengembangan perangkat lunak modern.

1.5.3 Manfaat Akademis

Selain memberikan dampak praktis, tugas akhir ini juga diharapkan dapat memperkaya literatur di bidang Teknologi Informasi. Berikut adalah manfaat akademis yang ditawarkan:

1. Menambah literatur kepustakaan di bidang Rekayasa Perangkat Lunak, khususnya sebagai referensi teknis dalam perancangan dan implementasi alat bantu (*tools*) otomatisasi dokumentasi berbasis *Command Line Interface* (CLI).
2. Memberikan contoh implementasi nyata mengenai pemanfaatan metode *Abstract Syntax Tree* (AST) untuk mengekstraksi informasi dari kode sumber JavaScript tanpa melalui proses eksekusi program.
3. Menjadi bahan rujukan bagi mahasiswa atau pengembang lain mengenai pentingnya penerapan standar dokumentasi (seperti JSDoc) untuk menjaga keterbacaan dan kerapian kode dalam proyek pengembangan perangkat lunak.

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan hasil implementasi dan pengujian sistem yang telah diuraikan pada bab-bab sebelumnya, dapat ditarik kesimpulan sebagai berikut:

1. Tugas akhir ini telah berhasil merancang dan membangun aplikasi generator dokumentasi otomatis berbasis *Command Line Interface* (CLI) menggunakan lingkungan runtime Node.js. Aplikasi yang didistribusikan melalui NPM dengan nama paket `my-docs-gen-nita` ini terbukti mampu berjalan pada lintas sistem operasi (Windows dan Linux) dan memudahkan pengembang dalam mengelola dokumentasi proyek JavaScript secara efisien melalui terminal.
2. Penerapan metode *Abstract Syntax Tree* (AST) menggunakan pustaka `acorn` dan `estraverse` berhasil diimplementasikan untuk membaca struktur kode secara mendalam. Mekanisme ini terbukti mampu mengekstraksi berbagai jenis tag JSDoc secara dinamis, termasuk `@param`, `@returns`, dan `@example`, serta memiliki ketahanan (*robustness*) dalam menangani variasi gaya penulisan komentar melalui fitur normalisasi teks dan penanganan kesalahan (*error handling*) yang telah disempurnakan.
3. Sistem berhasil menghasilkan luaran dokumentasi dalam dua format standar, yaitu Markdown (`.md`) untuk kompatibilitas repositori dan HTML (`.html`) untuk tampilan web yang interaktif. Berdasarkan hasil pengujian pengguna (*User Acceptance Testing*), output yang dihasilkan dinilai terstruktur, rapi, dan mudah dibaca. Hal ini mengonfirmasi bahwa alat bantu ini efektif meningkatkan aksesibilitas informasi proyek perangkat lunak.

5.2 Saran

Berdasarkan hasil evaluasi dan umpan balik teknis yang diperoleh selama proses pengembangan serta pengujian perangkat lunak, penulis mengidentifikasi beberapa aspek yang dapat ditingkatkan untuk pengembangan selanjutnya:

1. Disarankan untuk mengembangkan algoritma pemindaian yang lebih cerdas dengan fitur pengecualian (*exclusion*) otomatis. Saat ini, jika pengguna menjalankan perintah berulang kali pada direktori yang sama tanpa membersihkan luaran sebelumnya, berkas dokumentasi lama berpotensi ikut terpindai sebagai input, yang menyebabkan redundansi data. Penambahan fitur `.gitignore` atau argumen `--ignore` pada CLI akan sangat membantu mengatasi isu rekursif ini.
2. Pengembangan selanjutnya perlu memprioritaskan penanganan ambiguitas penamaan, khususnya pada tag `@typedef` dan objek bersarang (*nested objects*). Parser saat ini cenderung mengambil nama variabel deklarasi dibandingkan nama definisi tipe di JSDoc, sehingga diperlukan logika ekstraksi yang lebih spesifik untuk menjamin konsistensi referensi tipe data pada dokumen akhir.
3. Mengingat tren ekosistem JavaScript modern yang mengarah pada penggunaan tipe statis, pengembangan selanjutnya sangat disarankan untuk memperluas

cakupan parser agar mendukung sintaks `TypeScript` (.ts). Hal ini akan meningkatkan relevansi dan adopsi alat oleh spektrum pengembang yang lebih luas di industri perangkat lunak.

4. Guna menurunkan kurva pembelajaran (*learning curve*) bagi pengguna baru, dokumentasi resmi pada repositori NPM perlu diperkaya dengan variasi contoh penggunaan (*use cases*) yang lebih lengkap. Panduan langkah demi langkah (*step-by-step*), termasuk instruksi penanganan izin eksekusi (`chmod`) pada sistem operasi berbasis seperti Linux, sangat diperlukan untuk memastikan pengalaman pengguna yang mulus lintas platform.

DAFTAR PUSTAKA

[1] H. Washizaki, eds., *Guide to the Software Engineering Body of Knowledge (SWEBOK Guide), Version 4.0*, IEEE Computer Society, 2024; www.swebok.org. [Accessed: Nov. 29, 2025].

[2] Alzahrani, A. A. H. (2024). Software Systems Documentation: A Systematic Review. In *IJACSA) International Journal of Advanced Computer Science and Applications* (Vol. 15, Issue 8). [Accessed: Nov. 29, 2025].

[3] Santos, I., Felizardo, K. R., Gerosa, M. A., & Steinmacher, I. (2024). *Software Solutions for Newcomers' Onboarding in Software Projects: A Systematic Literature Review*. http://arxiv.org/abs/2408.15989. [Accessed: Nov. 29, 2025].

[4] Oliveira, D., Santos, R., Madeiral, F., Masuhara, H., & Castor, F. (2023). A systematic literature review on the impact of formatting elements on code legibility. *Journal of Systems and Software*, *203*. https://doi.org/10.1016/j.jss.2023.111728. [Accessed: Nov. 29, 2025].

[5] Freire, S., Rios, N., Pérez, B., Castellanos, C., Correal, D., Ramač, R., Mandić, V., Taušan, N., López, G., Pacheco, A., Mendonça, M., Falessi, D., Izurieta, C., Seaman, C., & Spínola, R. (2024). Hearing the Voice of Software Practitioners on Technical Debt Monitoring: Understanding Monitoring Practices and the Practices' Avoidance Reasons. *Journal of Software Engineering Research and Development*, *12*(1). https://doi.org/10.5753/jserd.2024.4011. [Accessed: Dec. 01, 2025].

[6] Huang, Y., Chen, Y., Chen, X., & Zhou, X. (2024). Are your comments outdated? Towards automatically detecting code-comment consistency. *Journal of Software: Evolution and Process*, *37*(1). http://arxiv.org/abs/2403.00251. [Accessed: Dec. 01, 2025].

[7] Khan, J. Y., & Uddin, G. (2022). Automatic Code Documentation Generation Using GPT-3; Automatic Code Documentation Generation Using GPT-3. *Association for Computing Machinery*. https://doi.org/10.1145/3551349. [Accessed: Dec. 08, 2025].

[8] Pinho, G., Jeová Caçula, A., Costa, L., Wiese, I., & Allex Araújo, A. (n.d.). Challenges and Solutions of Free and Open Source Software Documentation: A Systematic Mapping Study. In *Proceedings of Brazilian Symposium on Software Engineering (SBES'24)* (Vol. 1). [Accessed: Dec. 08, 2025].

[9] Kabir, S., Udo-Imeh, D. N., Kou, B., & Zhang, T. (2024, May 11). Is Stack Overflow Obsolete? An Empirical Study of the Characteristics of ChatGPT Answers to Stack Overflow Questions. *Conference on Human Factors in Computing Systems - Proceedings*. https://doi.org/10.1145/3613904.3642596. [Accessed: Jan. 30, 2026].

[10] Yao, Y., Duan, J., Xu, K., Cai, Y., Sun, Z., & Zhang, Y. (2024). A Survey on Large Language Model (LLM) Security and Privacy: The Good, The Bad, and The Ugly. In *High-Confidence Computing* (Vol. 4, Issue 2). Shandong University. https://doi.org/10.1016/j.hcc.2024.100211. [Accessed: Jan. 30, 2026].

[11] Sheta, S. V. (2023). *The Importance of Software Documentation in the Development and Maintenance Phases* (Vol. 24, Issue 3). http://www.veterinaria.org. [Accessed: Dec. 10, 2025].

[12] Moran, K., Yachnes, A., Purnell, G., Mahmud, J., Tufano, M., Cardenas, C. B., Poshyvanyk, D., & H'Doubler, Z. (2022). An Empirical Investigation into the Use of Image Captioning for Automated Software Documentation. *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 514–525. https://doi.org/10.1109/SANER53432.2022.00069. [Accessed: Dec. 10, 2025].

[13] Tan, W. S., Wagner, M., & Treude, C. (2023). Wait, wasn't that code here before? Detecting Outdated Software Documentation. *2023 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 553–557. https://doi.org/10.1109/ICSME58846.2023.00071. [Accessed: Dec. 11, 2025].

[14] Cadavid, H., Andrikopoulos, V., & Avgeriou, P. (2022). *Documentation-as-code for Interface Control Document Management in Systems of Systems: a Technical Action Research Study*. http://arxiv.org/abs/2206.11668. [Accessed: Dec. 11, 2025].

[15] Basutakara Jayanthi P N, A. S. (2021). A Review of Static Code Analysis Methods for Detecting Security Flaws. *Journal of University of Shanghai for Science and Technology*, *23*, 647–653. [Accessed: Dec. 16, 2025].

[16] Ruiz, G. A., Robledo Giraldo, S., & Morales, H. H. (2023). Static Code Analysis: A Tree of Science Review. *Entre Ciencia e Ingeniería*, *17*(34), 9–14. https://doi.org/10.31908/19098367.2846. [Accessed: Dec. 16, 2025].

[17] Vadoce, T., Pritchard, J., & Fairbanks, C. (2024). *Enhancing JavaScript Source Code Understanding with Graph-Aligned Large Language Models*. Research Square Preprints. https://doi.org/10.21203/rs.3.rs-5106829/v1. [Accessed: Dec. 16, 2025].

[18] Brasoveanu, A., Moodie, M., & Agrawal, R. (2023). Textual evidence for the perfunctoriness of independent medical reviews. *CEUR Workshop Proceedings*, *2657*, 1–9. https://doi.org/https://doi.org/10.48550/arXiv.2312.00413. [Accessed: Dec. 16, 2025].

[19] Curtis, J. (2022). Student Research Abstract: On Language-Agnostic Abstract-Syntax Trees. *Proceedings of the ACM Symposium on Applied Computing*, 1619–1622. https://doi.org/10.1145/3477314.3506962. [Accessed: Dec. 16, 2025].

[20] Åström, D. (2021). Implementation and Evaluation of an Emulated Permission Sys-tem for VS Code Extensions using Abstract Syntax Trees Implementation och Utvärdering av ett Emulerat Be-hörighetssystem för Extensions i VS Code med hjälp av Abstrakta Syntaxträd. *Digitala Vetenskapliga Arkivet*. www.liu.se. [Accessed: Dec. 16, 2025].

[21] Putra, I. S., Rukmono, S. A., & Perdana, R. S. (2021). Abstract Syntax Tree (AST) and Control Flow Graph (CFG) Construction of Notasi Algoritmik. *2021 International Conference on Data and Software Engineering (ICoDSE)*, 1–6. https://doi.org/10.1109/ICoDSE53690.2021.9648437. [Accessed: Dec. 16, 2025].

[22] Spirin, E., Bogomolov, E., Kovalenko, V., & Bryksin, T. (2021). PSIMiner: A Tool for Mining Rich Abstract Syntax Trees from Code. *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*, 13–17. https://doi.org/10.1109/MSR52588.2021.00014. [Accessed: Dec. 16, 2025].

[23] Rozi, M. F., Ban, T., Ozawa, S., Yamada, A., Takahashi, T., Kim, S., & Inoue, D. (2023). Detecting Malicious JavaScript Using Structure-Based Analysis of Graph Representation. *IEEE Access*, *11*, 102727–102745. https://doi.org/10.1109/ACCESS.2023.3317266. [Accessed: Dec. 16, 2025].

[24] Basumatary, B., & Agnihotri, N. (2022). Benefits and Challenges of Using NodeJS. *International Journal of Innovative Research in Computer Science & Technology*, 67–70. https://doi.org/10.55524/ijircst.2022.10.3.13. [Accessed: Dec. 17, 2025].

[25] Ahmod, M. (2023). JAVASCRIPT RUNTIME PERFORMANCE ANALYSIS: NODE AND BUN. *Faculty of Information Technology and Communication Sciences (ITC) Master's Thesis*. [Accessed: Dec. 17, 2025].

[26] Rahmatulloh, A., Nugraha, F., Gunawan, R., & Darmawan, I. (2022). Event-Driven Architecture to Improve Performance and Scalability in Microservices-Based Systems. *Proceedings - International Conference Advancement in Data Science, E-Learning and Information Systems, ICADEIS 2022*. https://doi.org/10.1109/ICADEIS56544.2022.10037390. [Accessed: Dec. 17, 2025].

[27] Xie, H., Tan, Z., & Lv, X. (2024). Application of Artificial Intelligence in Financial Risk Management in a Company. *ACM International Conference Proceeding Series*, 349–354. https://doi.org/10.1145/1122445.1122456. [Accessed: Dec. 17, 2025].

[28] Turcotte, A., Arteca, E., Mishra, A., Alimadadi, S., & Tip, F. (2022). Stubbifier: debloating dynamic server-side JavaScript applications. *Empirical Software*

Engineering, *27*(7), 161. https://doi.org/10.1007/s10664-022-10195-6. [Accessed: Dec. 17, 2025].

[29] Sampath, H., Merrick, A., & MacVean, A. (2021, May 6). Accessibility of command line interfaces. *Conference on Human Factors in Computing Systems - Proceedings*. https://doi.org/10.1145/3411764.3445544. [Accessed: Dec. 17, 2025].

[30] Rastogi, R., Anand, T., Sharma, S., & Panwar, S. (2024). Applying command line-based chat tool: design and development. *International Journal of Student Project Reporting*, *2*(2). https://doi.org/https://doi.org/10.1504/IJSPR.2024.10064148. [Accessed: Dec. 17, 2025].

[31] Argesanu, A.-I., & Andreescu, G.-D. (2023). ACTA TECHNICA NAPOCENSIS Series: Applied Mathematics STREAMLINING MACHINE LEARNING WORKFLOWS IN INDUSTRIAL APPLICATIONS WITH CLI'S AND CI/CD PIPELINES. In *Mechanics, and Engineering* (Vol. 66). [Accessed: Dec. 17, 2025].

[32] Roach, M. J., Pierce-Ward, N. T., Suchecki, R., Mallawaarachchi, V., Papudeshi, B., Handley, S. A., Brown, C. T., Watson-Haigh, N. S., & Edwards, R. A. (2022). Ten simple rules and a template for creating workflows-as-applications. *PLoS Computational Biology*, *18*(12). https://doi.org/10.1371/journal.pcbi.1010705. [Accessed: Dec. 17, 2025].

[33] Wadhams, Z., Reinhold, A. M., & Izurieta, C. (2024). Automating Static Code Analysis Through CI/CD Pipeline Integration. *2024 IEEE International Conference on Software Analysis, Evolution and Reengineering - Companion (SANER-C)*, 119–125. https://doi.org/10.1109/SANER-C62648.2024.00021. [Accessed: Dec. 17, 2025].

[34] Rani, P., Birrer, M., Panichella, S., Ghafari, M., & Nierstrasz, O. (2021). What Do Developers Discuss about Code Comments? *2021 IEEE 21st International Working Conference on Source Code Analysis and Manipulation (SCAM)*, 153–164. https://doi.org/10.1109/SCAM52516.2021.00027. [Accessed: Dec. 18, 2025].

[35] Savidis, A., & Ntoulas, M. (2022). Improved Untyped IntelliSense for JavaScript with Type Carriers and Binders. *2022 13th International Conference on Information, Intelligence, Systems & Applications (IISA)*, 1–8. https://doi.org/10.1109/IISA56318.2022.9904413. [Accessed: Dec. 18, 2025].

[36] Esakkiraja, E., Akhiyarov, D., Shanmugham, A., & Ganapathy, C. (2025). *DeepCodeSeek: Real-Time API Retrieval for Context-Aware Code Generation*. https://doi.org/https://doi.org/10.48550/arXiv.2509.25716. [Accessed: Dec. 18, 2025].

[37] Mäkiranta, E. (2023). *PILOTING MARKDOWN-BASED DOCUMENTATION Using Markdown for SoC documentation*. [Accessed: Dec. 18, 2025].

[38] Söderberg, D. (2022). Automation of Non-code Documentation in a DevOps Environment. *Digitala Vetenskapliga Arkivet*. https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1670611. [Accessed: Dec. 18, 2025].

[39] Wang, A. Y., Wang, D., Drozdal, J., Liu, X., Park, S., Oney, S., & Brooks, C. (2021, May 8). What Makes aWell-Documented Notebook? A Case Study of Data Scientists' Documentation Practices in Kaggle. *Conference on Human Factors in Computing Systems - Proceedings*. https://doi.org/10.1145/3411763.3451617. [Accessed: Dec. 18, 2025].

[40] White, J. (2022). Using Markup Languages for Accessible Scientific, Technical, and Scholarly Document Creation. *Journal of Science Education for Students with Disabilities*, *25*(1), 1–22. https://doi.org/10.14448/jsesd.14.0005. [Accessed: Dec. 18, 2025].

[41] Nur Adiya, A. Z. D., Anggraeni, D. L., & Ilham Albana. (2024). Analisa Perbandingan Penggunaan Metodologi Pengembangan Perangkat Lunak (Waterfall, Prototype, Iterative, Spiral, Rapid Application Development (RAD)). *Merkurius : Jurnal Riset Sistem Informasi Dan Teknik Informatika*, *2*(4), 122–134. https://doi.org/10.61132/merkurius.v2i4.148. [Accessed: Dec. 19, 2025].

[42] Maulida, N. (2022). *STUDI LITERATUR PENERAPAN METODE PROTOTYPE DAN WATERFALL DALAM PEMBUATAN SEBUAH APLIKASI ATAU WEBSITE*. https://www.researchgate.net/publication/359814579_STUDI_LITERATUR_PENERAPAN_METODE_PROTOTYPE_DAN_WATERFALL_DALAM_PEMBUATAN_SEBUAH_APLIKASI_ATAU_WEBSITE. [Accessed: Dec. 19, 2025].

[43] Maryani, Prabowo, H., Gaol, F. L., & Hidayanto, A. N. (2022). Comparison of the System Development Life Cycle and Prototype Model for Software Engineering. *International Journal of Emerging Technology and Advanced Engineering*, *12*(4), 155–162. https://doi.org/10.46338/ijetae0422_19. [Accessed: Dec. 19, 2025].

[44] OpenJS Foundation, "Node.js v20.10.0 Documentation," Node.js, 2023. [Online]. Available: https://nodejs.org/en/docs/. [Accessed: Dec. 20, 2025].

[45] Microsoft, "Visual Studio Code - Code Editing. Redefined," Visual Studio Code, 2023. [Online]. Available: https://code.visualstudio.com/. [Accessed: Dec. 20, 2025].

[46] M. Marijn, "Acorn: A tiny, fast JavaScript parser," GitHub, 2023. [Online]. Available: https://github.com/acornjs/acorn. [Accessed: Dec. 20, 2025].

[47] Git Community, "Git - Fast Version Control System," Git SCM, 2023. [Online]. Available: https://git-scm.com/. [Accessed: Dec. 20, 2025].

[48] npm, Inc., "npm Documentation," npm Docs, 2023. [Online]. Available: https://docs.npmjs.com/. [Accessed: Dec. 20, 2025].

[49] JGraph, "draw.io - Free Online Diagram Software," diagrams.net, 2023. [Online]. Available: https://app.diagrams.net/. [Accessed: Dec. 20, 2025].

[50] K. Sjösten, "Mermaid: Generate diagrams from markdown-like text," Mermaid, 2023. [Online]. Available: https://mermaid.js.org/. [Accessed: Dec. 20, 2025].

[51] Figma, "Figma: The Collaborative Interface Design Tool," 2025. [Online]. Available: https://www.figma.com/. [Accessed: Dec. 20, 2025].

[52] Stack Overflow, "2024 Developer Survey," Stack Overflow, May 2024. [Online]. Available: https://survey.stackoverflow.co/2024/. [Accessed: Jan. 31, 2026].

[53] D. Flanagan, JavaScript: The Definitive Guide, 6th ed. Sebastopol, CA: O'Reilly Media, 2011. [Accessed: Jan. 31, 2026].

[54] ESTree Community, "ESTree Spec: The De-Facto Standard for JavaScript ASTs," GitHub, 2024. [Online]. Available: https://github.com/estree/estree. [Accessed: Jan. 31, 2026].

[55] Baskoro, F., Andrahsmara, R. A., Darnoto, B. R. P., & Tofan, Y. A. (2021). A Systematic Comparison of Software Requirements Classification. *IPTEK The Journal for Technology and Science*, *32*(3), 184. https://doi.org/10.12962/j20882033.v32i3.13005. [Accessed: Mar. 04, 2026].

[56] Pressman, R. S. (2010). *Software Engineering: A Practitioner's Approach* (7th ed.). https://share.google/dzOWW2rDilpbkohhb. [Accessed: Mar. 04, 2026].

[57] Hennink, M., & Kaiser, B. N. (2022). Sample sizes for saturation in qualitative research: A systematic review of empirical tests. *Social Science and Medicine*, *292*. https://doi.org/10.1016/j.socscimed.2021.114523 [Accessed: Mar. 03, 2026].

[58] Freire, S., Rios, N., Pérez, B., Castellanos, C., Correal, D., Ramač, R., Mandić, V., Taušan, N., López, G., Pacheco, A., Mendonça, M., Falessi, D., Izurieta, C., Seaman, C., & Spínola, R. (2024). Hearing the Voice of Software Practitioners on Technical Debt Monitoring: Understanding Monitoring Practices and the Practices' Avoidance Reasons. *Journal of Software Engineering Research and Development*, *12*(1). https://doi.org/10.5753/jserd.2024.4011. [Accessed: Mar. 04, 2026].