TUGAS AKHIR

RANCANG BANGUN PLUGIN VS CODE EXTENSION UNTUK DOKUMENTASI OTOMATIS REST API DENGAN STANDAR OPENAPI BERBASIS TOOLS AI

Sebagai Memenuhi Persyaratan Mencapai Derajat Sarjana (S1)

DISUSUN OLEH:
MOHAMAD AENUR ROKHMAN
NIM.21106050081

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

LEMBAR PENGESAHAN

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

PENGESAHAN TUGAS AKHIR

Nomor : B-636/Un.02/DST/PP.00.9/04/2026

Tugas Akhir dengan judul : Rancang Bangun Plugin VS Code Extension Untuk Dokumentasi Otomatis Rest API dengan Standar Openapi Berbasis Tools AI

yang dipersiapkan dan disusun oleh:

Nama : MOHAMAD AENUR ROKHMAN
Nomor Induk Mahasiswa : 21106050081
Telah diujikan pada : Rabu, 11 Maret 2026
Nilai ujian Tugas Akhir : A

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

TIM UJIAN TUGAS AKHIR

Ketua Sidang

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

Valid ID: 69d8793eb0c48

Penguji I

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

Valid ID: 69d8723bd67c0

Penguji II

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

Valid ID: 69d86f2b1a543

Yogyakarta, 11 Maret 2026
UIN Sunan Kalijaga
Dekan Fakultas Sains dan Teknologi

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

Valid ID: 69d8a1c604e10

SURAT PERSETUJUAN

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

SURAT PERSETUJUAN SKRIPSI/TUGAS AKHIR

Hal : Persetujuan Skripsi / Tugas Akhir
Lamp : -

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

Assalammu'alaikum wr. wb.

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

Nama	:	Mohamad Aenur Rokhman
NIM	:	21106050081
Judul Skripsi	:	Rancang Bangun Plugin VS Code Extension Untuk Dokumentasi Otomatis Rest API dengan Standar Openapi Berbasis Tools AI.

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 dapat segera dimunaqasyahkan. Atas perhatiannya saya ucapkan terima kasih.

Wassalammu'alaikum wr. wb.

Yogyakarta, 27 Februari 2026
Pembimbing

Muhammad Didik Rohmad Wahyudi, S.T., MT
NIP. 19841115 201903 1 003

PERNYATAAN KEASLIAN

SURAT PERNYATAAN KEASLIAN/BEBAS PLAGIASI

Saya yang bertanda tangan dibawah ini:

Nama	: Mohamad Aenur Rokhman
NIM	: 21106050081
Program Studi	: Informatika
Fakultas	: Sains dan Teknologi

Dengan ini menyatakan bahwa isi tugas akhir saya yang berjudul "RANCANG BANGUN PLUGIN VS CODE EXTENSION UNTUK DOKUMENTASI OTOMATIS REST API DENGAN STANDAR OPENAPI BERBASIS TOOLS AI" merupakan hasil pemikiran penulis sendiri, tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan orang lain sepengetahuan penulis, kecuali yang tertulis dalam tugas akhir dan disebutkan dalam daftar pustaka.

Demikian surat pernyataan ini saya buat untuk dipergunakan sebagaimana mestinya.

Yogyakarta, 27 Februari 2026

Mohamad Aenur Rokhman

NIM.21106050081

ABSTRAK

Dokumentasi REST API merupakan komponen kritis dalam pengembangan perangkat lunak modern berbasis arsitektur mikrolayanan, namun proses pembuatannya masih bergantung pada anotasi manual yang rawan terhadap *documentation drift* dan *documentation smells*. Penelitian ini bertujuan merancang dan mengimplementasikan plugin Visual Studio Code Extension yang mampu menghasilkan dokumentasi REST API secara otomatis sesuai standar OpenAPI 3.1 dengan memanfaatkan kemampuan model kecerdasan buatan berbasis *Large Language Model* (LLM). Sistem yang dikembangkan mengintegrasikan model Gemma 3 12B-IT melalui platform inferensi OpenRouter AI, dengan mekanisme *prompt engineering* berbasis *few-shot learning* untuk menghasilkan output berformat YAML atau JSON yang valid secara sintaksis. Metodologi pengembangan menggunakan pendekatan Agile dengan kerangka kerja Kanban yang bersifat iteratif dan adaptif. Arsitektur sistem mengadopsi pola *client-server* di mana plugin VS Code berperan sebagai klien yang mengirimkan potongan kode hasil ekstraksi ke API OpenRouter, kemudian menerima respons dokumentasi terstruktur dalam format OpenAPI. Plugin dilengkapi dengan modul *RouteParser* untuk mengidentifikasi endpoint dan parameter secara otomatis, modul *OpenAPIGenerator* untuk menyusun spesifikasi dokumen, serta *SecureStorageService* untuk pengelolaan API key. Pengujian dilakukan meliputi pengujian fungsionalitas, validasi format OpenAPI, pengujian performa, dan pengujian akurasi dokumentasi dibandingkan dengan dokumentasi manual. Hasil penelitian menunjukkan bahwa plugin yang dikembangkan mampu mengotomatisasi proses dokumentasi REST API secara langsung di dalam lingkungan IDE VS Code, mengurangi beban kerja pengembang, dan menghasilkan dokumentasi yang sesuai standar OpenAPI 3.1 dengan tingkat akurasi dan konsistensi yang lebih baik dibandingkan pendekatan berbasis aturan konvensional.

Kata Kunci: VS Code Extension, REST API, Dokumentasi Otomatis, OpenAPI 3.1, Large Language Model, Gemma 3, OpenRouter AI, Prompt Engineering

ABSTRACT

REST API documentation is a critical component in modern software development based on microservices architecture; however, its creation still relies heavily on manual annotation, which is prone to documentation drift and documentation smells. This study aims to design and implement a Visual Studio Code Extension plugin capable of automatically generating REST API documentation in compliance with the OpenAPI 3.1 standard by leveraging the capabilities of AI-based Large Language Models (LLMs). The developed system integrates the Gemma 3 12B-IT model through the OpenRouter AI inference platform, employing a few-shot learning-based prompt engineering mechanism to produce syntactically valid YAML or JSON output. The development methodology adopts an Agile approach with an iterative and adaptive Kanban framework. The system architecture follows a client-server pattern in which the VS Code plugin acts as a client that sends extracted code snippets to the OpenRouter API and receives structured documentation responses in OpenAPI format. The plugin is equipped with a RouteParser module for automatic endpoint and parameter identification, an OpenAPIGenerator module for composing document specifications, and a SecureStorageService for API key management. Testing encompasses functional testing, OpenAPI format validation, performance testing, and documentation accuracy evaluation against manually written documentation. The results demonstrate that the developed plugin successfully automates the REST API documentation process directly within the VS Code IDE environment, reduces developer workload, and produces documentation conforming to the OpenAPI 3.1 standard with greater accuracy and consistency compared to conventional rule-based approaches.

Keywords: VS Code Extension, REST API, Automatic Documentation, OpenAPI 3.1, Large Language Model, Gemma 3, OpenRouter AI, Prompt Engineering

KATA PENGANTAR

Puja dan puji syukur kehadirat Allah SWT yang telah memberikan rahmat dan hidayah-Nya sehingga penulis dapat memulai mengerjakan sampai menyelesaikan tugas akhir yang berjudul "Rancang Bangun Plugin VS Code Extension Untuk Dokumentasi Otomatis Rest API dengan Standar Openapi Berbasis Tools AI" Sholawat serta salam senantiasa tercurahkan kepada baginda Nabi Muhammad SAW, yang telah memberikan syafaat serta teladan bagi umat islam dalam menuntut ilmu dan mengamalkannya.

Tersusunnya tugas akhir ini tidak lepas dari dukungan berbagai pihak yang telah memberikan semangat, bimbingan, dan kontribusi selama proses penyusunan. Oleh karena itu, dengan kerendahan hati, penulis menyampaikan ucapan terima kasih yang sebesar-besarnya kepada:

1. Bapak Prof. Noorhaidi Hasan, S.Ag., M.A., M.Phil., Ph.D., selaku Rektor Universitas Islam Negeri Sunan Kalijaga Yogyakarta.
2. Ibu Prof. Dr. Dra. Hj. Khurul Wardati, M.Si., selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Sunan Kalijaga Yogyakarta.
3. Bapak Muhammad Mustakim, S.T., M.T., selaku Ketua Program Studi Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Sunan Kalijaga Yogyakarta.
4. Bapak Ir. Muhammad Didik Rohmad Wahyudi, S.T., M.T., selaku Dosen Pembimbing Tugas Akhir yang telah meluangkan waktu untuk membantu dan mengarahkan selama proses penulisan tugas akhir.
5. Bapak Eko Hadi Gunawan, M.ENG., selaku Dosen Pembimbing Akademik yang telah membantu penulis selama perkuliahan. Seluruh Dosen dan Karyawan Program Studi Informatika UIN Sunan Kalijaga yang telah memberikan ilmu dan bantuan selama perkuliahan.
6. Kedua orang tua penulis Bapak Sunarto dan Ibu Tarisah yang telah memberikan dukungan materi dan moral kepada penulis hingga dapat

menempuh pendidikan tinggi agar menjadi pribadi yang berkontribusi pada negara.

7. Seluruh teman Informatika yang membantu penulis selama perkuliahan terutama teman Informatika angkatan 2021 yang telah menjadi saksi perjuangan penulis pada saat perkuliahan.

Tugas akhir ini tidak terlepas dari kesalahan dan kekurangan kualitas materi. Untuk itu, mohon kritik dan saran yang membangun agar dapat membawa manfaat bagi pembaca. Semoga tugas akhir ini dapat memberikan pengetahuan yang baru dan dapat dijadikan referensi pembaca untuk kedepannya.

Yogyakarta, 26 Februari 2026

Penulis

Mohamad Aenur Rokhman

NIM.21106050081

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I PENDAHULUAN

1.1 Latar Belakang

Dalam dua dekade terakhir, arsitektur perangkat lunak modern mengalami perubahan signifikan menuju paradigma mikrolayanan (*microservices*), yang mengedepankan modularitas, skalabilitas, dan fleksibilitas sistem. Setiap layanan dalam arsitektur ini berinteraksi melalui *Representational State Transfer* (REST) *Application Programming Interface* (API), sebuah antarmuka berbasis HTTP yang memungkinkan pertukaran data lintas sistem dengan format standar seperti JSON atau YAML [1]. Menurut Meshram [2], REST API telah menjadi tulang punggung komunikasi antar komponen sistem terdistribusi karena sifatnya yang ringan, independen terhadap platform, dan mudah diimplementasikan. Meskipun REST API telah menjadi standar de facto dalam rekayasa perangkat lunak modern, salah satu tantangan yang masih sering muncul adalah pemeliharaan dan pembaruan dokumentasi API.

Dokumentasi API berperan penting dalam menjembatani pemahaman antara pengembang backend dan konsumen layanan (frontend, mobile, atau pihak ketiga). Tanpa dokumentasi yang jelas, proses integrasi antar sistem menjadi lambat dan rawan kesalahan [3]. Penelitian oleh Bondel et al. [4] menemukan bahwa sekitar 40% penyedia API publik mengalami masalah keterlambatan pembaruan dokumentasi karena perubahan kode yang tidak diikuti oleh pembaruan deskripsi endpoint. Kondisi ini dikenal sebagai documentation drift, yaitu ketidaksesuaian antara dokumentasi dan implementasi aktual API. Masalah lain yang sering muncul adalah documentation smells, yaitu ketidakkonsistenan atau kesalahan semantik dalam dokumentasi API yang mengurangi keterbacaan dan keandalan dokumentasi. Khan et al. [5] menegaskan bahwa meskipun banyak alat bantu dokumentasi telah dikembangkan, seperti Swagger, Redocly, dan API Blueprint, permasalahan dokumentasi tidak selalu dapat diatasi sepenuhnya. Sebagian besar alat tersebut bergantung pada anotasi manual atau komentar dalam kode sumber, sehingga tetap

menuntut intervensi manusia dalam memperbarui informasi ketika struktur kode berubah.

Sejumlah pendekatan telah diajukan untuk mengotomatisasi proses dokumentasi API. Solusi konvensional meliputi code parser dan documentation generator yang mengekstraksi informasi dari struktur kode dan komentar. Namun, pendekatan ini masih bersifat rule-based dan bergantung pada format penulisan tertentu [6]. Menurut Dhyani [7], keterbatasan pendekatan berbasis aturan ini membuatnya sulit menangani variasi sintaks dan konteks logika yang kompleks dalam proyek nyata. Oleh karena itu, muncul kebutuhan akan sistem dokumentasi otomatis yang tidak hanya membaca kode secara statis, tetapi juga memahami konteks dan perilaku kode secara semantik. Perkembangan teknologi Artificial Intelligence (AI), khususnya Large Language Model (LLM), membuka peluang baru untuk menghadirkan sistem dokumentasi yang lebih adaptif dan kontekstual. LLM seperti GPT, LLaMA, dan Gemma memiliki kemampuan memahami struktur dan makna kode melalui contextual embeddings yang dilatih dari miliaran token kode dan dokumentasi perangkat lunak [8].

Menurut Wang et al. [9], integrasi LLM dalam proses dokumentasi memungkinkan pembangkitan deskripsi API secara otomatis dengan tingkat relevansi yang lebih tinggi dibandingkan metode tradisional. Penelitian gDoc oleh Wang et al. [10] menunjukkan kemampuan model bahasa besar dalam menghasilkan dokumentasi API yang terstruktur sesuai spesifikasi industri seperti OpenAPI 3.1, dengan tingkat kesesuaian sintaks mencapai lebih dari 92%. Selain itu, Lazar et al. [11] menekankan pentingnya standard compliance dalam konteks dokumentasi API modern. OpenAPI sebagai standar terbuka (disahkan oleh Linux Foundation) memungkinkan interoperabilitas antar sistem dengan format dokumentasi yang konsisten. Namun, banyak penelitian sebelumnya masih terfokus pada pembuatan standalone documentation generator, bukan pada integrasi langsung ke dalam Integrated Development Environment (IDE) seperti Visual Studio Code. Padahal, integrasi langsung ke IDE dapat mengurangi beban

kognitif pengembang dengan memungkinkan dokumentasi dihasilkan secara real-time saat menulis kode [12]. Visual Studio Code (VS Code) merupakan IDE yang populer dengan lebih dari 14 juta pengguna aktif bulanan [13]. Ekosistemnya yang berbasis extension memungkinkan pengembang menambahkan fungsionalitas baru secara modular. Dalam beberapa tahun terakhir, sejumlah ekstensi AI seperti GitHub Copilot dan Tabnine telah menunjukkan efektivitas AI dalam mendukung aktivitas pengembangan perangkat lunak, termasuk penulisan kode otomatis dan deteksi kesalahan sintaks [14]. Namun, sejauh ini belum ada ekstensi yang secara khusus mengotomatisasi dokumentasi REST API dengan memanfaatkan model AI berbasis konteks kode dan menghasilkan keluaran sesuai standar OpenAPI secara langsung di VS Code.

Dari sisi metodologi pengembangan, pendekatan Agile Software Development, khususnya kerangka kerja Scrum, menjadi pilihan yang ideal untuk pengembangan plugin semacam ini. Menurut Schwaber dan Sutherland [15], Agile berfokus pada iterasi cepat, umpan balik berkelanjutan, dan fleksibilitas dalam menghadapi perubahan kebutuhan karakteristik yang sangat sesuai untuk proyek berbasis integrasi AI, di mana eksperimen dan penyesuaian model merupakan bagian penting dari siklus pengembangan. Berdasarkan tinjauan tersebut, penelitian ini mengusulkan perancangan dan implementasi plugin Visual Studio Code Extension untuk dokumentasi otomatis REST API berbasis AI dengan standar OpenAPI 3.1. Sistem ini akan memanfaatkan kemampuan inferensi model Gemma 3 12B-IT, yaitu model instruction-tuned large language model (LLM) generasi terbaru dari Google DeepMind yang dirilis pada tahun 2025. Model ini bersifat open-weight dan dirancang untuk mendukung tugas pemahaman kode (code comprehension), penjelasan logika program, serta pembuatan dokumentasi teknis berbasis konteks [16]. Inferensi model dilakukan melalui OpenRouter AI, sebuah platform inferensi terbuka yang menyediakan akses terpadu ke berbagai model besar, termasuk Gemma 3, tanpa kebutuhan infrastruktur GPU lokal [17]. Pemilihan Gemma 3 12B-IT didasarkan pada kombinasi kemampuan semantik tingkat lanjut dan efisiensi latensi inferensi, yang menjadikannya ideal untuk

integrasi on-the-fly di lingkungan pengembangan terdistribusi seperti Visual Studio Code (VS Code). Secara konseptual, plugin yang dikembangkan dalam penelitian ini akan berfungsi sebagai agen AI yang secara otomatis memindai file proyek (khususnya berbasis Node.js/Express), mengidentifikasi endpoint REST, parameter, dan response, kemudian menghasilkan dokumentasi dalam format OpenAPI 3.1 yang tervalidasi otomatis.

Dengan memanfaatkan konektivitas inferensi berbasis OpenRouter, proses dokumentasi dapat diperbarui setiap kali kode mengalami perubahan, memastikan dokumentasi selalu selaras dengan implementasi aktual (synchronized documentation). Melalui pendekatan ini, penelitian diharapkan dapat menjawab tiga masalah utama dalam dokumentasi API modern: keterlambatan pembaruan, ketidaksesuaian sintaks dengan standar, dan minimnya integrasi antara dokumentasi dan lingkungan pengembangan. Dengan demikian, penelitian ini tidak hanya berkontribusi pada pengembangan perangkat lunak praktis berupa plugin VS Code, tetapi juga memberikan kontribusi ilmiah dalam bidang AI-assisted Software Documentation dan contextual code understanding. Secara akademis, hasil penelitian ini dapat menjadi acuan bagi pengembangan sistem dokumentasi cerdas di masa depan, serta memperluas penerapan LLM dalam domain software engineering automation.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan sebelumnya, maka penelitian ini merumuskan masalah: Bagaimana memanfaatkan model AI berbasis konteks kode untuk mengotomatisasi dokumentasi REST API yang sesuai standar OpenAPI 3.1 secara langsung di lingkungan pengembangan melalui plugin Visual Studio Code Extension?

1.3 Batasan Masalah

Agar penelitian ini dapat dilaksanakan secara terarah, fokus, dan sesuai dengan ruang lingkup yang realistis, maka diperlukan batasan masalah dari sistem yang dikembangkan. Pembatasan ini juga dimaksudkan untuk memastikan bahwa setiap tahapan dalam penelitian sesuai dengan tujuan utama. Adapun batasan-batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Penelitian ini dibatasi pada konteks perancangan dan implementasi plugin Visual Studio Code Extension yang berfungsi untuk menghasilkan dokumentasi REST API secara otomatis menggunakan model kecerdasan buatan (AI) Gemma 3 12B-IT melalui OpenRouter AI. Ruang lingkup sistem yang dikembangkan difokuskan pada integrasi antara model AI dan lingkungan pengembangan Integrated Development Environment (IDE) Visual Studio Code. Plugin ini dirancang untuk mendeteksi struktur endpoint dari proyek berbasis Node.js dan Express.js, lalu menghasilkan dokumentasi dalam format standar OpenAPI versi 3.1 sebagaimana ditetapkan oleh OpenAPI Initiative. Penelitian ini tidak mencakup integrasi untuk framework selain Express.js, serta tidak membahas proses fine-tuning model AI atau pengembangan model baru. Dengan demikian, sistem yang dirancang hanya berfokus pada proses inferensi berbasis prompt engineering untuk menghasilkan deskripsi dokumentasi dari konteks kode sumber.

2. Proses pengembangan dibatasi sampai tahap proof of concept (PoC) dan evaluasi sistem, tanpa mencakup proses distribusi plugin ke marketplace Visual Studio Code atau implementasi dalam skala produksi. Penelitian ini juga tidak membahas aspek bisnis, pemasaran, maupun manajemen proyek di luar konteks pengembangan teknis.
3. Pengujian dilakukan pada lingkungan pengembangan lokal dengan konfigurasi perangkat keras menengah, menggunakan sistem operasi Windows 10. Evaluasi sistem mencakup empat dimensi utama: fungsionalitas plugin, validasi dokumentasi terhadap format OpenAPI menggunakan Swagger Validator dan OpenAPI Linter Tools, pengukuran performa inferensi model melalui response time dan penggunaan sumber daya, serta pengujian akurasi hasil dokumentasi dibandingkan dokumentasi manual yang dibuat oleh pengembang. Namun, penelitian ini tidak mengevaluasi aspek keamanan model AI di sisi server, kinerja OpenRouter AI dalam skala masif, maupun keandalan inferensi pada jaringan tidak stabil.
4. Penelitian ini membatasi penggunaan model Gemma 3 12B-IT dalam mode inference only, tanpa melibatkan pelatihan ulang (fine-tuning) atau modifikasi arsitektur model. Model ini dipilih karena kemampuannya dalam memahami konteks kode secara semantik serta efisiensinya pada platform low-latency inference milik Openrouter AI. Proses inferensi dilakukan dengan mengirimkan potongan kode (snippet) yang relevan melalui API, kemudian hasil keluaran berupa deskripsi endpoint, parameter, dan respons API diterjemahkan ke format OpenAPI. Fokus penelitian terletak pada evaluasi kemampuan model dalam mengidentifikasi struktur logika kode dan menghasilkan dokumentasi yang sesuai standar, bukan pada eksplorasi arsitektur internal model atau teknik pembelajaran mesin yang mendasarinya.
5. Karena model AI dijalankan secara cloud-based, penelitian ini memastikan bahwa kode sumber yang dikirimkan untuk inferensi hanyalah bagian terbatas yang relevan, tanpa menyertakan data sensitif seperti API key,

kredensial, atau informasi rahasia milik pengguna. Aspek keamanan jaringan, autentikasi API, enkripsi data, serta kepatuhan terhadap regulasi keamanan dan privasi data tidak dibahas secara mendalam. Penelitian ini lebih berfokus pada bagaimana integrasi model AI dapat meningkatkan efisiensi dan kualitas dokumentasi secara fungsional, bukan pada sistem keamanan data.

1.4 Tujuan

Tujuan utama dari penelitian ini adalah untuk merancang dan mengimplementasikan plugin Visual Studio Code Extension yang mampu menghasilkan dokumentasi REST API secara otomatis berdasarkan kode sumber, dengan memanfaatkan kecerdasan buatan (AI) dan mengikuti standar OpenAPI 3.1. Sistem ini diharapkan dapat memberikan solusi terhadap permasalahan klasik dalam pengembangan perangkat lunak modern, yaitu keterlambatan dan ketidaksesuaian dokumentasi dengan implementasi aktual kode sumber. Dengan mengintegrasikan model AI langsung ke dalam lingkungan pengembangan Visual Studio Code, proses dokumentasi diharapkan dapat dilakukan secara real-time tanpa mengganggu alur kerja pengembang.

1.5 Manfaat

Penelitian ini diharapkan memberikan manfaat dalam tiga dimensi utama, yaitu manfaat akademis, praktis, dan teknis, baik bagi pengembangan ilmu pengetahuan maupun penerapannya di industri perangkat lunak modern.

1. Manfaat Akademis

   Penelitian ini berkontribusi dalam pengembangan ilmu di bidang rekayasa perangkat lunak berbasis kecerdasan buatan (AI-assisted software engineering). Integrasi Large Language Model (LLM) seperti Gemma 3 12B-IT dalam proses dokumentasi kode memberikan arah baru dalam studi

otomatisasi dokumentasi perangkat lunak, yang sebelumnya didominasi oleh pendekatan berbasis aturan (rule-based generators) atau parser statis.

2. Manfaat Praktis

   Penelitian ini menghadirkan solusi nyata bagi pengembang perangkat lunak yang menghadapi tantangan dalam membuat dokumentasi REST API yang selalu terkini dan sesuai standar. Plugin yang dikembangkan dalam penelitian ini memungkinkan dokumentasi dihasilkan secara otomatis langsung di lingkungan Visual Studio Code, sehingga pengembang tidak perlu bergantung pada alat eksternal seperti Swagger Editor atau Postman untuk memperbarui dokumentasi. Proses ini mendukung continuous documentation, di mana setiap perubahan pada kode dapat langsung diikuti dengan pembaruan dokumentasi. Dengan cara ini, sistem yang dikembangkan tidak hanya mempercepat proses kerja, tetapi juga meningkatkan kualitas dokumentasi dan mengurangi risiko kesalahan komunikasi antar tim pengembang.

3. Manfaat Teknis

   Penelitian ini menunjukkan penerapan konkret dari integrasi model kecerdasan buatan (AI) ke dalam ekosistem IDE modern melalui mekanisme plugin modular yang dapat digunakan ulang (reusable) dan diperluas (extensible), yang memungkinkan integrasi di masa depan dengan framework lain seperti Django REST Framework, Laravel API, atau FastAPI. Selain itu, hasil penelitian ini dapat menjadi blueprint bagi pengembang ekstensi IDE lainnya untuk mengintegrasikan layanan AI generatif ke dalam proses dokumentasi, pengujian, atau analisis kode.

BAB V PENUTUP

5.1 Kesimpulan

Implementasi sistem pendokumentasian REST API berbasis kecerdasan buatan dalam bentuk ekstensi Visual Studio Code telah berhasil dilakukan dan memenuhi seluruh spesifikasi fungsional yang direncanakan. Berdasarkan hasil pengujian yang dilakukan secara menyeluruh, sistem ini mampu melakukan pemindaian jalur komunikasi pada aplikasi Express.js dan mentransformasikannya menjadi dokumen teknis yang sepenuhnya mematuhi standar spesifikasi OpenAPI versi terbaru. Pengintegrasian model bahasa besar melalui layanan penghubung pihak ketiga terbukti memberikan hasil dokumentasi yang lebih deskriptif dan kontekstual dibandingkan metode konvensional yang mengandalkan templat statis. Seluruh komponen inti, mulai dari pemindai kode hingga generator dokumen, bekerja secara terintegrasi sehingga menghasilkan keluaran yang valid dan konsisten.

5.2 Saran untuk Pengembangan Selanjutnya

Penelitian ini masih memiliki beberapa keterbatasan yang dapat diperbaiki untuk meningkatkan kualitas layanan pada masa yang akan datang. Salah satu aspek utama yang perlu dikembangkan adalah perluasan dukungan terhadap kerangka kerja selain Express.js agar sistem ini dapat digunakan oleh populasi pengembang yang lebih luas. Modifikasi pada bagian pemindai rute perlu dilakukan agar sistem dapat mengenali pola deklarasi pada kerangka kerja lain seperti *Fastify* atau *NestJS*. Peningkatan ini akan memberikan fleksibilitas yang lebih besar bagi tim pengembang yang bekerja dengan berbagai lingkungan teknis yang berbeda di dalam satu organisasi.

DAFTAR PUSTAKA

[1] A. Ehsan, M. A. M. E. Abuhaliqa, C. Catal, and D. Mishra, "RESTful API Testing Methodologies: Rationale, Challenges, and Solution Directions," *Appl. Sci.*, vol. 12, no. 9, p. 4369, Apr. 2022, doi: 10.3390/app12094369.

[2] S. U. Meshram, "Evolution of Modern Web Services – REST API with its Architecture and Design," *Int J Res. Eng. Sci. Manag.*, vol. 4, pp. 83–86, 2021.

[3] P. Gowda and A. N. Gowda, "Best Practices in REST API Design for Enhanced Scalability and Security," *J. Artif. Intell. Mach. Learn. Data Sci.*, vol. 2, no. 1, pp. 827–830, Feb. 2024, doi: 10.51219/JAIMLD/priyanka-gowda/202.

[4] G. Bondel, A. Cerit, and F. Matthes, "Challenges of API Documentation from a Provider Perspective and Best Practices for Examples in Public Web API Documentation:," in *Proceedings of the 24th International Conference on Enterprise Information Systems*, Online Streaming, --- Select a Country ---: SCITEPRESS - Science and Technology Publications, 2022, pp. 268–279. doi: 10.5220/0011089700003179.

[5] J. Y. Khan, M. T. I. Khondaker, G. Uddin, and A. Iqbal, "Automatic Detection of Five API Documentation Smells: Practitioners' Perspectives," Feb. 16, 2021, *arXiv*: arXiv:2102.08486. doi: 10.48550/arXiv.2102.08486.

[6] Oracle, "AI Case Study: Auto-Generation of Swagger Documentation for Oracle API Gateway Cloud Service | 4i." Accessed: Dec. 09, 2025. [Online]. Available: https://www.4iapps.com/ai-case-study-auto-generation-of-swagger-documentation-for-oracle-api-gateway-cloud-service/

[7] P. Dhyani, S. Nautiyal, A. Negi, S. Dhyani, and P. Chaudhary, "Automated API Docs Generator using Generative AI," in *2024 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*, Bhopal, India: IEEE, Feb. 2024, pp. 1–6. doi: 10.1109/SCEECS61402.2024.10482119.

[8] "OpenAPI Specification - Version 3.1.0 | Swagger." Accessed: Dec. 09, 2025. [Online]. Available: https://swagger.io/specification/

[9] S. Wang, Y. Tian, and D. He, "Improving API Documentation Comprehensibility via Continuous Optimization and Multilingual SDK," Mar. 24, 2023, *arXiv*: arXiv:2303.13828. doi: 10.48550/arXiv.2303.13828.

[10] S. Wang, Y. Tian, and D. He, "gDoc: Automatic Generation of Structured API Documentation," in *Companion Proceedings of the ACM Web Conference 2023*, Apr. 2023, pp. 53–56. doi: 10.1145/3543873.3587310.

[11] K. Lazar *et al.*, "Generating OpenAPI Specifications from Online API Documentation with Large Language Models".

[12] Microsoft, "Web API Design Best Practices - Azure Architecture Center." Accessed: Dec. 09, 2025. [Online]. Available: https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design

[13] Strapi Team, "13 Essential VS Code Extensions for 2025." Accessed: Dec. 09, 2025. [Online]. Available: https://strapi.io/blog/vs-code-extensions

[14] S. Cavalcante, E. Ribeiro, and A. Oran, "The Impact of AI Tools on Software Development: A Case Study with GitHub Copilot and Other AI Assistants:," in *Proceedings of the 27th International Conference on Enterprise Information*

Systems, Porto, Portugal: SCITEPRESS - Science and Technology Publications, 2025, pp. 245–252. doi: 10.5220/0013294700003929.
[15] ..... K. Schwaber and J. Sutherland, *The Scrum Guide*. Scrum Alliance, 2020.
[16]G. Team *et al.*, "Gemma 3 Technical Report," Mar. 25, 2025, *arXiv*: arXiv:2503.19786. doi: 10.48550/arXiv.2503.19786.
[17]OpenRouter AI, "OpenRouter Quickstart Guide," OpenRouter Documentation. Accessed: Dec. 09, 2025. [Online]. Available: https://openrouter.ai/docs/quickstart
[18]H. Alaidaros, M. Omar, and R. Romli, "The state of the art of agile kanban method: challenges and opportunities," *Indep. J. Manag. Prod.*, vol. 12, no. 8, pp. 2535–2550, Dec. 2021, doi: 10.14807/ijmp.v12i8.1482.
[19]Y. Dong *et al.*, "A Survey on Code Generation with LLM-based Agents," Sep. 30, 2025, *arXiv*: arXiv:2508.00083. doi: 10.48550/arXiv.2508.00083.
[20]S. M. Abtahi and A. Azim, "Augmenting Large Language Models with Static Code Analysis for Automated Code Quality Improvements," Jun. 12, 2025, *arXiv*: arXiv:2506.10330. doi: 10.48550/arXiv.2506.10330.
[21]Z. Feng *et al.*, "CodeBERT: A Pre-Trained Model for Programming and Natural Languages," Sep. 18, 2020, *arXiv*: arXiv:2002.08155. doi: 10.48550/arXiv.2002.08155.
[22]Y. Wang, W. Wang, S. Joty, and S. C. H. Hoi, "CodeT5: Identifier-aware Unified Pre-trained Encoder-Decoder Models for Code Understanding and Generation," Sep. 02, 2021, *arXiv*: arXiv:2109.00859. doi: 10.48550/arXiv.2109.00859.
[23]W. Sun *et al.*, "Source Code Summarization in the Era of Large Language Models," Aug. 23, 2025, *arXiv*: arXiv:2407.07959. doi: 10.48550/arXiv.2407.07959.
[24]A. Svyatkovskiy, S. K. Deng, S. Fu, and N. Sundaresan, "IntelliCode Compose: Code Generation Using Transformer," Oct. 29, 2020, *arXiv*: arXiv:2005.08025. doi: 10.48550/arXiv.2005.08025.
[25]Q. Zhang *et al.*, "A Survey on Large Language Models for Software Engineering," Sep. 08, 2024, *arXiv*: arXiv:2312.15223. doi: 10.48550/arXiv.2312.15223.
[26]Y. Jorelle, "Generation of API Documentation using Large Language Models -- Towards Self-explaining APIs," *MS Thesis Sch. Sci. Aalto Univ*, 2024.
[27]"REST API Documentation Tool | Swagger UI." Accessed: Dec. 09, 2025. [Online]. Available: https://swagger.io/tools/swagger-ui/
[28]R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," *PhD Diss. Univ Calif. Irvine*, 2000.
[29]"VS Code API." Accessed: Dec. 09, 2025. [Online]. Available: https://code.visualstudio.com/api/references/vscode-api
[30]Y. Wu *et al.*, "LLM Prompt Duel Optimizer: Efficient Label-Free Prompt Optimization," Oct. 14, 2025, *arXiv*: arXiv:2510.13907. doi: 10.48550/arXiv.2510.13907.

LAMPIRAN

Github Projects:

https://github.com/users/hyea404/projects/2/views/1

Repository Proyek di Github :

https://github.com/hyea404/rest-api-doc-generator/commits/main/