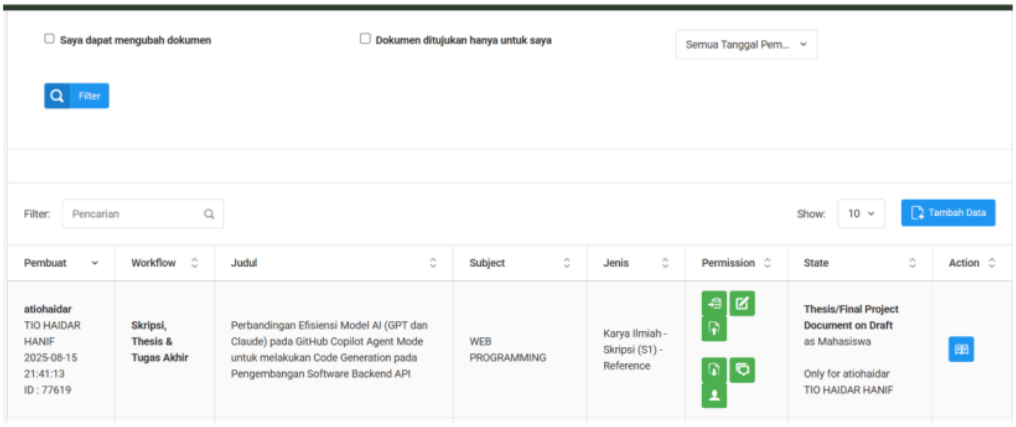
1302210057

By Tio Haidar Hanif

WORD COUNT	19701	TIME SUBMITTED	20-AUG-2025 01:54PM
		PAPER ID	117811297

01. SS Unggah Open Library



2 a. Buku TA lengkap

Perbandingan Efisiensi Model AI (GPT dan Claude) pada GitHub Copilot Agent Mode untuk melakukan Code Generation pada Pengembangan Software Backend API

Tugas Akhir

diajukan untuk memenuhi salah satu syarat memperoleh gelar sarjana

dari Program Studi S1 Rekayasa Perangkat Lunak

Fakultas Informatika

Universitas Telkom

1302210057

Tio Haidar Hanif

Program Studi Sarjana S1 Rekayasa Perangkat Lunak

Fakultas Informatika

Universitas Telkom

Bandung

2025

LEMBAR PENGESAHAN

Perbandingan Efisiensi Model AI (GPT dan Claude) pada GitHub Copilot Agent Mode untuk melakukan Code Generation pada Pengembangan Software Backend API

Comparison of Efficiency of AI Models (GPT and Claude) on GitHub Copilot Agent Mode for Backend API Software Development

1302210057

Tio Haidar Hanif

Tugas akhir ini telah diterima dan disahkan untuk memenuhu sebagai syarat memperoleh gelar pada Program Studi Sarjana <Nama Prodi>

Fakultas Informatika

Universitas Telkom

Bandung, 15 Agustus 2025

Menyetujui

Pembimbing I,	Pembimbing II,
Dana Sulistiyo Kusumo, S.T., M.T., PhD	Nungki Selviandro, S.Kom., M.Kom., Ph.D.
NIP: 02780011	NIP: 14880076

Ketua Program Studi
Sarjana S1 Rekayasa Perangkat Lunak,

15/08/2025

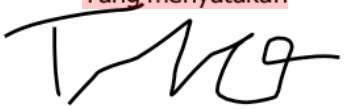
Dr. Eng. Jati Hiliamsyah Husen, S.T., M.Eng.
NIP: 20920040

LEMBAR ORISINALITAS

Dengan ini saya, Tio Haidar Hanif menyatakan sesungguhnya bahwa Tugas Akhir saya dengan judul "Perbandingan Efisiensi Model AI (GPT dan Claude) pada GitHub Copilot Agent Mode untuk melakukan Code Generation pada Pengembangan Software Backend API" berserta dengan seluruh isinya merupakan hasil karya saya sendiri, dengan tidak melakukan penjiplakan yang tidak sesuai dengan etika keilmuan yang berlaku dengan masyarakat keilmuan, serta produk dari tugas akhir ini bukan merupakan hasil dari *Generative AI*. Saya siap menanggung risiko/sanksi yang diberikan jika di kemudian hari ditemukan pelanggaran terhadap etika keilmuan dalam Laporan Tugas Akhir, atau jika ada klaim dari pihak lain terhadap keaslian karya.

Bandung, 15 Agustus 2025

Yang menyatakan



Tio Haidar Hanif

NIM 1302210057

ABSTRAK

Penggunaan AI sudah sangat marak digunakan, dan salah satu yang terdampak adalah pada ranah pengembangan aplikasi. Github Copilot yang merupakan salah satu *tools* berbasis AI yang dapat mempermudah pengembangan aplikasi serta meningkatkan efisiensi dalam pengembangan aplikasi. Namun, tingkat efisiensi pada Github Copilot juga bergantung dengan model AI yang digunakannya. Penelitian ini bertujuan untuk membandingkan secara kuantitatif efisiensi dari model AI GPT 4.1 dan Claude 3.7 Sonnet menggunakan Github Copilot dengan Mode Agent dengan studi kasus untuk melakukan pengembangan aplikasi *backend* REST API. Penelitian ini menggunakan pendekatan studi komparatif dengan melakukan eksperimen membuat beberapa fitur yang dikembangkan secara pararel pada masing-masing model. Pengukuran yang digunakan pada penelitian ini adalah menggunakan metode Goal Question Metric (GQM) dengan Metrik yang diukur adalah jumlah *prompt* dan durasi yang dibutuhkan oleh setiap *prompt* untuk menyelesaikan tiap tugas. Kesimpulan dari penelitian ini memperlihatkan adanya pengaruh antara kecepatan dengan model yang digunakan untuk *code generation*. GPT 4.1 menunjukkan keunggulan dari model ini yang dilihat dari durasi yang dibutuhkan pada model tersebut yang lebih cepat serta jumlah *prompt* uang dilakukan yang lebih sedikit. Akan tetapi, Claude 3.7 lebih efektif dalam menangani permintaan pembuatan kode yang lebih besar pada satu kali *prompt.* Dengan catatan, durasi *prompt* yang dilakukan cenderung lebih lama.

Kata Kunci:Generative AI, LLM4SE, Github Copilot, Rest API, Backend

ABSTRACT

The use of AI has become widespread, and one area that has been significantly impacted is application development. Github Copilot is one of the AI-based tools that can simplify application development and improve efficiency in the development process. However, the efficiency of Github Copilot also depends on the AI model it uses. This study aims to quantitatively compare the efficiency of the GPT 4.1 and Claude 3.7 Sonnet AI models using Github Copilot in Agent Mode, with a case study focused on developing a backend REST API application. The study employs a comparative approach by conducting experiments to develop several features in parallel across each model. The measurement used in this study is the Goal Question Metric (GQM) method, with the metrics measured being the number of prompts and the duration required by each prompt to complete each task. The conclusion of this study shows a correlation between speed and the model used for code generation. GPT 4.1 demonstrates the advantages of this model, as seen in the shorter duration required by the model and the fewer prompts needed. However, Claude 3.7 is more effective in handling larger code generation requests in a single prompt.

Keywords: Generative AI, LLM4SE, Github Copilot, Rest API, Backend

KATA PENGANTAR

Alhamdulillah, puji syukur atas kehadirat Allah yang telah memberikan segala nikmat karunia sehingga penulis bisa menyelesaikan Tugas Akhir dengan judul "Perbandingan Efisiensi Model AI (GPT dan Claude) pada GitHub Copilot Agent Mode untuk melakukan Code Generation pada Pengembangan Software Backend API". Penulisan TA ini merupakan syarat wajib untuk memperoleh gelar sarjana para program studi S1 Rekayasa Perangkat Lunak Universitas Telkom. Penulis mengambil tema ini karena penulis memiliki rasa keingintahuan yang tinggi terhadap teknologi AI. Untuk ke depannya penulis rasa AI akan menjadi bagian dari berbagai lini kehidupan, yang salah satunya menyangkut pada program studi ini. Proses penyusunan TA ini merupakan pembelajaran yang berharga untuk diri saya sendiri. Penulis ingin menyampaikan rasa Terima Kasih sebesar besarnya pada pihak yang berkontribusi baik secara langsung maupun tidak langsung pada proses penyusunan dokumen ini. Penulis juga menyadari bahwa Tugas Akhir ini masih sangat banyak kekurangan, baik dari segi penulisan atau konten dari Tugas Akhir ini. Untuk kritik yang membangun dari pembaca sangat kami harapkan agar perbaikan di masa mendatang. Semoga TA ini bisa memberikan manfaat dan bisa memberikan kontribusi positif dalam perkembangan ilmu pengetahuan.

UCAPAN TERIMA KASIH

Dengan segenap kerendahan hati, penulis haturkan puji dan syukur yang tak terhingga ke hadirat Allah Subhanahu Wa Ta'ala. Atas segala rahmat, karunia, dan nikmat-Nya yang tak terhingga—yang tak mungkin tertuliskan seluruhnya dalam dokumen ini—penulis akhirnya dapat menyelesaikan skripsi sebagai tahap akhir dari perjalanan studi ini.

Oleh karena itu, penulis ingin menyampaikan rasa terima kasih kepada:

1. Bapak Ode dan Mamah Ratna Astuti, kedua orang tua tercinta yang selalu menjadi sumber kekuatan. Terima kasih atas setiap doa, dukungan moril dan materiel
2. Bapak Dana Sulistiyo, S.T., M.T., PhD dan Bapak Nungki, S.Kom., M.Kom.,Ph.D selaku dosen pembimbing. Terima kasih atas kesabaran, waktu, ilmu, dan arahan yang sangat berharga dalam membimbing penulis dari awal hingga akhir pengerjaan skripsi ini.
3. Rekan-rekan satu tim TA Capstone (Sul, Nabiel, Ghaza, Burhan, Rizki). Terima kasih atas kerja sama, solidaritas, dan perjuangan bersama yang telah kita lalui sejak penyusunan proposal hingga saat ini.
4. Keluarga besar LDK Al-Fath, DKM Syamsul Ulum, dan Badan Mentoring. Terima kasih telah menjadi lingkungan yang positif, penuh inspirasi, dan selalu mengingatkan penulis pada tujuan yang lebih besar.
5. Seluruh pihak lain yang tidak dapat penulis sebutkan satu per satu, terima kasih atas segala bentuk bantuan dan dukungan yang telah diberikan.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1 PENDAHULUAN

1.1. Latar Belakang

Dalam pengembangan perangkat lunak, efisiensi merupakan aspek krusial yang mencakup waktu pengerjaan dan biaya pengerjaan. Di era teknologi dengan kompleksitas yang sangat tinggi, industri perangkat lunak memiliki tekanan untuk meningkatkan produktivitas serta mengurangi biaya yang berkaitan dengan pembuatan dan pemeliharaan perangkat lunak [1]. Akan tetapi, sering kali aspek ini kurang diperhatikan pada beberapa proyek. Berdasarkan laporan pada tahun 2025 dari gitnux.org[1], sekitar 55% proyek pengembangan perangkat lunak mengalami keterlambatan dan kelebihan biaya. Selain itu berdasarkan penelitian yang dilakukan oleh Standish Group's CHAOS Study[2], hanya 31% *project* IT yang benar-benar memenuhi target (termasuk ketepatan waktu dan anggaran). Oleh karena itu, efisiensi menjadi hal yang penting dalam pengembangan proyek aplikasi.

Pada era AI (*Artificial Intelligent*) seperti saat ini, solusi dari permasalahan dapat direalisasikan dengan lebih cepat dan lebih murah dengan bantuan AI [2]. Inovasi AI yang berupa Large Language Model (LLM) dan *chatbot* sangat berpengaruh dan mengubah cara menyelesaikan masalah serta proses pengembangan perangkat lunak [3]. Kemampuan AI ini bukan hanya sekedar memahami dan menghasilkan bahasa alami, tetapi juga dapat meningkatkan produktivitas dalam melakukan pengkodean program [3].

---

[1] Jannik Linder , *Software Development Industry Statistics* (https://gitnux.org/software-development-industry-statistics/, diakses pada 29 Juni 2025)
[2] Dillon Courts, *Software Project Failures: Why 70% Miss the Mark* (https://www.callibrity.com/articles/why-software-projects-miss-the-mark, diakses pada 22 Mei 2025, 2025)

Ada banyak *tools* berbasis AI yang dapat melakukan *code generation[4].* Salah satu dari *tools* tersebut yang cukup populer adalah Github Copilot[3]. Github Copilot merupakan *tools* untuk *pair programing* berbasis AI yang dapat terintegrasi langsung dengan *code editor [5]*. Keunggulan Github Copilot dalam membantu berbagai pekerjaan *programmer* telah dibuktikan melalui berbagai penelitian, baik yang dilakukan oleh Github langsung[4] maupun peneliti lainnya[6].

Berdasarkan penelitian yang dilakukan oleh Pandey et al. [6], menggunakan Github Copilot dapat menghemat waktu hingga 50% dalam dokumentasi kode, serta efisiensi 30% sampai 40% untuk tugas pengkodean berulang, pembuatan *unit test*, *debuging* dan *pair programming*. Serta mengurangi waktu pengkodean sebanyak 35% dari *baseline [C]*. Salah satu fitur baru dari Github Copilot adalah Agent Mode, yang menawarkan kemudahan dalam pengembangan aplikasi karena dengan fitur ini AI dapat melakukan operasi *file* pada proyek program, menjalankan perintah di terminal, merespons kesalahan dan masih banyak lagi[5]. Pada Github Copilot, pengguna diberikan kebebasan untuk memilih model AI yang digunakan[6].

Salah satu faktor yang menentukan kualitas dari hasil keluaran AI adalah model yang digunakannya [7]. Sehingga dapat diasumsikan tingkat efisiensi dari prosesnya juga tergantung pada model yang digunakan. Perbedaan tingkat efisiensi yang kecil pada proses pengembangan perangkat lunak dapat berpengaruh jika digunakan pada skala proyek besar. Sebagai contoh jika

---

[3] Ishnu Vasudevan, *Github Copilot Adoption Trends: Insights from Real Data* (https://www.opsera.io/blog/github-copilot-adoption-trends-insights-from-real-data, diakses pada 6 Juni 2025)

[4] Ya Gao C GitHub Customer Research, *Research: Quantifying GitHub Copilot's impact in the enterprise with Accenture* (https://github.blog/news-insights/research/research-quantifying-github-copilots-impact-in-the-enterprise-with-accenture/, diakses pada 6 Juni 2025)

[5] Isidor Nikolic, *Introducing GitHub Copilot agent mode (preview).* (https://code.visualstudio.com/blogs/2025/02/24/introducing-copilot-agent-mode, diakses oada 8 Juni 2025)

[6] Isidor Nikolic, *Introducing GitHub Copilot agent mode (preview).* (https://code.visualstudio.com/blogs/2025/02/24/introducing-copilot-agent-mode, diakses oada 8 Juni 2025)

pada model A, proses pembuatan satu fitur adalah 5 menit, dan model B bisa sampai 6 menit, maka jika dihitung kasar pada proyek perangkat lunak dengan 100 fitur, model A dapat menghemat waktu sebanyak 100 menit. Bahkan, pada pengembangan perangkat lunak besar dapat sampai 1195 fitur yang dibuat [8]. Oleh karena itu perbandingan model AI diperlukan untuk memperkirakan tingkat efisiensi selama proses pengembangan perangkat lunak yang memanfaatkan AI.

Namun, dari penelitian yang sudah ditemukan, belum banyak membahas tentang perbandingan efisiensi pada proses penggunaan AI pada pengembangan perangkat lunak pada proyek nyata menggunakan Github Copilot Agent Mode. Banyak penelitian sebelumnya yang menggunakan dataset yang ada untuk melakukan pengujiannya, bukan berdasarkan kasus proyek nyata [3] [9] [10] [11]. Kemudian penelitian dengan kasus pengujian berdasarkan kasus nyata hanya membandingkan tingkat akurasinya[12] ataupun tingkat *correctness [13]*. Akurasi dan *correctness* dari model AI dalam *code generation* adalah hal yang penting, akan tetapi efisiensi dari proses juga penting.

Oleh karena itu, pada penelitian kali ini, penulis mengusulkan melakukan perbandingan antar model AI menggunakan Github Copilot Agent Mode untuk menguji efisiensi masing-masing model dalam melakukan pengembangan sistem aplikasi. Penelitian ini akan melakukan studi komparatif untuk membandingkan antara kedua model yaitu GPT dan Claude pada Github Agent Mode. Dengan adanya penelitian ini, diharapkan dapat menjadi wawasan kepada tim *developer* untuk memilih model AI yang paling sesuai dengan kebutuhan dan alur kerja.

1.2. Rumusan Masalah

Berdasarkan latar belakang yang sudah dituliskan sebelumnya, rumusan masalah dari penelitian ini adalah sebagai berikut

- Bagaimana perbandingan jumlah *prompt* serta durasi setiap prompt yang dibutuhkan pada setiap model AI (GPT dan Claude) untuk menyelesaikan serangkaian tugas yang sama dalam pengembangan perangkat lunak sistem *backend* REST API menggunakan Github Copilot Agent?
- Berdasarkan analisis data yang dilakukan serta pengalaman selama pengembangan, model AI mana yang memiliki tingkat efisiensi terbaik untuk melakukan tugas pengembangan perangkat lunak sistem *backend* REST API menggunakan Github Copilot Agent?

1.3. Tujuan dan Manfaat

Berdasarkan latar belakang yang ada, tujuan penelitian ini dapat dibagi dua yaitu

- Mengukur serta membandingkan tingkat efisiensi dari dua model AI untuk menghasilkan kode dalam pengembangan aplikasi menggunakan Github Copilot Agent Mode
- Memberikan data empiris sebagai dasar evaluasi terkait penggunaan Generative AI dalam mengembangkan aplikasi pada proyek nyata.

Adapun manfaat dari penelitian ini di antaranya adalah

- Untuk membantu pengguna atau *developer* dalam memilih model AI yang paling sesuai untuk membangun aplikasi dengan menggunakan Github Copilot Agent Mode
- Bagi peneliti, membantu memberikan referensi terkait efisiensi penggunaan AI pada lingkup *software development* terutama pada *code generation*

1.4. Batasan Masalah

Batasan masalah yang ada pada penelitian ini ada beberapa di antaranya adalah:

- Studi kasus pada penelitian ini dilaksanakan secara berkelompok sebagai bagian dari TA *Capstone,* dengan peneliti berperan sebagai *Backend Developer.* Fokus pengembangan aplikasinya dibuat hanya dari sisi *backend* dengan *output* berupa kode program menggunakan laravel 12 dan *API Documentation* yang di dihasilkan oleh AI. Sedangkan untuk rancangan kebutuhan aplikasinya ditentukan oleh *system analist* dan mengacu juga pada MVP yang dibuat *hustler* sampai tanggal 23 Juni 2025.
- Perbandingan yang dilakukan hanya dari sisi yang dirasakan oleh *developer* saja, tanpa melibatkan hal yang berkaitan langsung dengan *machine learning* seperti *fine tuning* dan lain sebagainya
- Akun yang digunakan pada Github Copilot menggunakan versi *student*. Sehingga ada kemungkinan perbedaan durasi setiap *prompt* antara Versi Pro dengan versi *student*.
- Kecepatan dari durasi *prompt* dapat tergantung dengan koneksi internet yang digunakan oleh peneliti.
- Penelitian dilaksanakan pada rentang waktu Mei 2025 sampai Juni 2025, sehingga mungkin di luar waktu tersebut ada perubahan dari Github Copilot yang dapat mengakibatkan hasil yang diberikan tidak sama persis dengan penelitian saat ini karena mungkin akan perubahan seperti peningkatan kualitas, baik dari model maupun dari *tools* itu sendiri.
- Model yang diujikan hanya 2 saja yaitu GPT 4.1 dan Claude 3.7 Sonnet yang ada pada Github Copilot.

1.5. Metode Penelitian

Penelitian ini menggunakan pendekatan studi komparatif dengan metode eksperimen untuk melakukan perbandingan efisiensi antara dua model AI yaitu GPT 4.1 dan Claude 3.7. pada *platform* Github Copilot Agent Mode. Pendekatan eksperimen ini digunakan untuk melakukan simulasi

pengembangan aplikasi pada dunia nyata, sehingga menggunakan *benchmark* standar yang ada.

Efisiensi diukur menggunakan metrik yang diadopsi dari *framework* yang bernama *Goal Question Metric* dengan metrik utamanya adalah jumlah *prompt* dan durasi setiap *prompt* yang dibutuhkan untuk menyelesaikan serangkaian tugas. Data kuantitatif akan dianalisis menggunakan statistika deskriptif sederhana untuk membandingkan kedua performa dari model tersebut.

BAB 2 TINJAUAN PUSTAKA

Bab ini menjelaskan tentang landasan teori dari penelitian ini serta penelitian terdahulu.

2.1. Landasan Teori

2.1.1. AI dan LLM

AI (Artificial Intelligent) merupakan teknologi meniru kecerdasan yang selayaknya manusia berfikir, mengambil keputusan, bahkan melakukan tindakan[7]. AI bekerja berdasarkan data yang dianalisis dan dijadikan sebuah pola untuk pengambilan keputusan. LLM (Large Language Model) merupakan salah satu penerapan dari Artificial Intelegece (AI) –Lebih tepatnya *Generative AI*[8]– yang dapat meniru dan memahami bahasa manusia dengan memanfaatkan *machine learning [14]*. LLM bekerja berdasarkan konsep *neuron network* dengan cara mempelajari pola dan probabilitas statistik dari sejumlah kata yang dilakukan anilisis yang memungkinkan LLM untuk memprediksi kata berikutnya, menjawab pertanyaan serta menghasilkan teks layaknya manusia[9].

LLM memahami dan melakukan tugas serta memberikan keluaran dengan menggunakan bahasa manusia sebagai perintah yang nantinya perintah tersebut akan disebut dengan *prompt [15].* LLM dapat melakukan pekerjaan manusia seperti *text generation, language translation* bahkan dapat melakukan pekerjaan yang lebih spesifik pada *software engineering* seperti *Code Generation, code summarization* dan masih banyak lagi [1]. Berdasarkan salah satu *Systematic Literatur Review* yang dipublikasikan pada

---

[7] *Artificial Intelligence: Pengertian, Cara Kerja dan Manfaatnya* (https://binus.ac.id/bandung/2024/02/artificial-intelligence-pengertian-cara-kerja-dan-manfaatnya/, diakses pada 06 Juni 2025)

[8] Lucas Mearian, *What are LLMs, and how are they used in generative AI?* (https://www.computerworld.com/article/1627101/what-are-large-language-models-and-how-are-they-used-in-generative-ai.html, diakses pada 6 Juni 2025)

[9] *Model Bahasa Besar* (https://www.solix.com/ms/kb/large-language-models/, diakses pada 06 Juni 2025)

Desember 2024 [16], terdapat lebih dari 300 penelitian yang menjelaskan tentang penerapan LLM pada *Software Engineering*. Mulai dari *Requirement Engineering, Software Design, Software Development, Software Management* sampai *Software Maintance*.

2.1.2. *Code Generation*

Code Generation merupakan proses untuk membuat kode secara otomatis berdasarkan spesifikasi atau kebutuhan yang ditentukan oleh *developer* [17]. *Code Generation* merupakan salah satu bagian dari penerapan LLM pada *Software Engineering* di bidang *Software Development* [16]. Bahkan dari temuan tersebut, lebih dari 115 penelitian yang membahas tentang *Code Generation.* Dengan *code generation, developer* dapat mengurangi *effort* dalam melakukan pengkodean serta memungkinkan untuk fokus pada tugas dan penyelesaian masalah yang lebih kompleks [17]. Hal ini menunjukan betapa banyak minat penelitian terkait bidang tersebut. *Code Generation* dengan menggunakan LLM dilakukan dengan cara memberikan perintah (*prompt*) dengan bahasa manusia ke dalam *tools* untuk *code generation* seperti Github Copilot.

2.1.3. Github Copilot

Github Copilot merupakan salah satu *coding assistant* yang dibuat oleh OpenAI dan Microsoft yang saat ini banyak diteliti terkait efektifitas dan keandalannya dalam melakukan *code generation*[6]. Github Copilot menggunakan teknik *machine learning* dan *codebase* yang luas sehingga memungkinkan *developer* untuk meningkatkan produktivitas serta mempercepat proses pengembangan perangkat lunak[18].

Github Copilot Agent Mode merupakan salah satu fitur dari Github Copilot yang dapat memberikan pengalaman yang berbeda dalam melakukan

pengkodean karena dapat memahami permintaan yang lebih kompleks[10]. Mode Agent menawarkan kemudahan dalam pengembangan aplikasi karena dengan fitur ini AI dapat melakukan operasi *file* pada proyek program, menjalankan perintah di terminal, merespons kesalahan dan masih banyak lagi[11].

2.1.4. Model: Claude G GPT

Salah satu faktor yang menentukan kualitas dari hasil AI merupakan model [7]. Model merupakan program yang sebelumnya sudah dilatih oleh sekumpulan data untuk mengenali pola tertentu atau membuat keputusan berdasarkan data tanpa intervensi dari manusia[12]. Ada berbagai macam model AI, model ini dibagi menjadi 3 jenis berdasarkan arsitekturnya seperti *Encoder Only*, *Encoder-Decoder*, dan *Decoder Only[1C]*. Akan tetapi, model yang umum digunakan untuk *code generation* adalah *Decoder Only[1C]* . Ada banyak macam model yang termasuk kategori *Decoder Only*, di antaranya ada Claude dari Antropic, Gemini dari Google, GPT dari OpenAI dan masih banyak lagi[16].

Dari sekian banyak model yang tersedia, penulis akan berfokus pada dua model yaitu Claude dan GPT. Kedua model tersebut dipilih karena memiliki keunggulan di antara model yang lain seperti pada penelitian tentang LLM4DS yang membandingkan beberapa model lain [19]. Pada penelitian berbeda – persisnya tentang perbandingan model AI untuk pembuatan kode untuk HRI– Claude memiliki performansi paling tinggi yaitu ada di angka 95% untuk ketepatan dalam pembuatan kode [12]. Kemudian pada penelitian [19], ChatGPT merupakan model yang memiliki keunggulan ketika menangani masalah yang sulit. Selain itu, ChatGPT-4 secara spesifik memiliki keunggulan

---

[10] Brylie Christopher Oxley, *GitHub Copilot Agent Mode: Enhancing Developer Workffows* (https://dev.to/brylie/github-copilot-agent-mode-enhancing-developer-workflows-2ae0, diakses pada 8 Juni 2925)

[11] Isidor Nikolic, *Introducing GitHub Copilot agent mode (preview).* (https://code.visualstudio.com/blogs/2025/02/24/introducing-copilot-agent-mode, diakses pada 8 Juni 2025)

[12] IBM, *What is an AI model?* (https://www.ibm.com/think/topics/ai-model, . 4 Mei 2025)

dibanding Gemini dalam mendeteksi dan memperbaiki *bug* pada kode program [1].

Pada penelitian kali ini peneliti berfokus pada dua model yaitu GPT dan Claude. Versi yang akan digunakan adalah GPT 4.1 dan Claude 3,7 Sonnet. Versi tersebut dipilih karena versi tersebut merupakan versi tertinggi (sebelum tanggal 22 Mei 2025) yang ada di Github Copilot ketika penelitian ini dibuat. GPT 4.1 rilis pada tanggal 14 April 2025[13] sedangkan Claude 3.7 Sonnet rilis pada tanggal 25 Februari 2025[14]. Sehingga dari tanggal rilis yang masih dalam tahun yang sama masih dianggap relevan untuk dilakukan pengujian.

Terdapat beberapa perbedaan kedua model yang digunakan[15] yang relevan pada penelitian ini terkait. GPT 4.1 memberikan fleksibilitas yang lebih besar karena memiliki dukungan 50 bahasa dan *context windows* yang banyak yaitu sebanyak satu juta *token*. Sementara Claude 3.7, sangat baik saat menggunakan bahasa inggris, serta menyediakan *context windows* sebanyak 200 ribu token. *Knowledge cutoff* pada GPT 4.1 ada pada Juni 2024, sedangkan Claude ada pada Maret 2024.

2.1.5. Rest API

Rest API yang merupakan singkatan dari *Represetational State Transsfer API* merupakan gaya arsitektur perangkat lunak yang dikemukakan oleh Roy Thomas Fleding pada tahun 2000 [20]. Rest API merupakan gaya arsitektur yang melibatkan jaringan untuk komunikasinya. Rest memisahkan antara *server* dan *client* sebagai penerapan prinsip *sepearation of concen*.

---

[13] Tempo, *OpenAI Buka Akses GPT-4.1 untuk Pelanggan Plus, Pro, dan Team* (https://www.tempo.co/sains/openai-buka-akses-gpt-4-1-untuk-pelanggan-plus-pro-dan-team--1443992, diakses pada 6 Jun 2025

[14] Tempo, *Anthropic Luncurkan Model AI Claude 3.7 Sonnet, Ungguli Model OpenAI dan DeepSeek?* (https://www.tempo.co/digital/anthropic-luncurkan-model-ai-claude-3-7-sonnet-unggul-model-openai-dan-deepseek--1212031, diakses pada 9 Juni 2025)

[15]OpenAI GPT 4.1 vs Claude 3.7 vs Gemini 2.5: Which Is Best AI?(https://yourgpt.ai/blog/updates/openai-gpt-4-1-vs-claude-3-7-vs-gemini-2-5, diakses pada 21 Juli 2024)

Penggunaan Rest API digunakan untuk melakukan komunikasi antara *frontend* dengan *backend* pada proyek penelitian ini.

2.1.6. Efisiensi

Berdasarkan Kamus besar bahasa Indonesia, Efisiensi berarti 'Ketepatan waktu cara (usaha, kerja) dalam menjalankan sesuatu (dengan tidak membuang waktu, tenaga biaya)[16]. Dalam pengembangan aplikasi, efisiensi dapat diukur dengan menghitung waktu yang dibutuhkan untuk menyelesaikan suatu proses, serta jumlah sumber daya yang digunakan pada setiap proses [21]. Pada penelitian ini, efisien dilihat dari berapa banyak *prompt* yang digunakan, dan waktu yang dibutuhkan dalam melakukan *prompting*.

2.1.7. Goal Question Matrix (GQM)

Goal Question Matrix merupakan salah satu paradigma yang diteliti kembali oleh Basili dan Rombach [21] untuk melakukan pengukuran *software* dan prosesnya. Seperti namanya, paradigma ini terdiri dari tiga bagian yaitu Goals, Question dan Matrix.

Goals berarti tujuan yang ingin dicapai, Question berarti pertanyaan yang ingin dijawab dari tujuan tersebut, dan Metric berarti aspek pengukuran yang perlu dikumpulkan untuk menjawab pertanyaan yang diajukan[17]. Referensi penelitian yang menggunakan metode ini adalah penelitian untuk melakukan *code generation* untuk masalah *data science* [19].

2.2. Penelitian Terdahulu

Dari berbagai model AI yang tersedia pada Github Copilot, penelitian ini akan berfokus dalam menggunakan model GPT dari Open AI dan Claude sebagai objek penelitian. Pemilihan kedua model ini dipilih karena keunggulannya dibanding dengan model lain pada penelitian sebelumnya.

[16] KBBI, (https://kbbi.kemdikbud.go.id/entri/efisiensi, diakses pada 8 Juni 2025)
[17] Goal-Question-Metrics (GQMs), (https://www.productplan.com/glossary/goal-question-metrics/, diakses pada 8 Juni 2025)

Sebagai contoh studi komparatif yang dilakukan oleh Nascimento et al. [19] yang melakukan pengujian pada konteks pembuatan kode untuk keperluan *data science*. Dari berbagai model yang diuji seperti Microsoft Copilot (GPT-4 Turbo), ChatGPT (o1-preview), Claude (3.5 Sonnet), and Perplexity Labs (Llama-3.1-70b-instruct), ditemukan bahwa hanya GPT dan Claude yang memiliki tingkat keberhasilan diatas 60%. Pada penelitian tersebut juga ChatGPT menunjukkan konsistensi dari performanya pada setiap tingkatan tugas. Kemudian penelitian lain yang membahas studi komparatif pada pembuatan kode untuk Human-Robot Interaction (HRI), Claude berhasil unggul dan menunjukkan performa akurasi tertinggi hingga 95% [12].

Kebanyakan penelitian terkait perbandingan model AI menggunakan *dataset* seperti humaneval, leetcode, pass@k dan lain sebagainya[16] [3] [9] [10] [11]. Salah satu kekurangan menggunakan *dataset* adalah ada kemungkinan hasil perbandingannya tidak mewakili kompleksitas yang ada pada dunia nyata[18]. Oleh karena itu perlu juga dilakukan evaluasi berdasaran pengalaman saat proses pengembangan *software* yang harapannya dapat digunakan untuk keputusan praktis di industri.

Kemudian, perbandingan model AI tanpa menggunakan *dataset* adalah penelitian yang dilakukan oleh Sobo et al. [12]. Penelitian ini melakukan studi komparatif terkait efektivitas antara model GPT, Claude, dan Gemini dalam pembuatan kode untuk HRI. Penelitian ini menggunakan 20 prompt yang di desain untuk *generate task* pada *robotic task* seperti *basic movement* dan lain sebagainya Proses ini melakukan *prompt* pada tiap LLM yang diuji, lalu mengintegrasikannya dengan lingkungan uji yang sudah dibuat oleh peneliti. Setiap *logic* dari hasil LLM tidak ada yang diubah untuk memastikan integritas penelitian. Akan tetapi kesalahan minor secara *syntax* seperti kehilangan titik koma masih ditoleransi. Penelitian tersebut relevan dengan penelitian yang dilakukan saat ini. Akan tetapi penelitian ini bukan untuk mengukur efisiensi

[18] *What is HumanEval?* (https://www.deepchecks.com?m/glossary/humaneval/, Diakses pada 12 Mei 2025)

dari masing-masing model AI serta studi kasus yang digunakan bukan untuk membuat aplikasi *backend*.

Untuk penelitian terhadap proyek nyata, Pandey et al. [6] melakukan evaluasi terhadap penggunaan Github Copilot pada proyek *Software Engineering* di dunia nyata. efisiensi dan tantangan dalam penggunaannya adalah aspek yang dievaluasi pada penelitian tersebut dalam menggunakan Github Copilot. Penelitian ini memberi kesimpulan bahwa Github Copilot secara signifikan meningkatkan efisiensi dari developer. Github Copilot baik dalam mengurangi waktu developer pada tugas yang repetitif.

Terkait perhitungan efisiensi, penelitian yang dilakukan Wang et al [22] mengukur waktu lama latensi dalam *generate code* untuk tiap fungsi. Penelitian ini menghasilkan angka 0.63 sampai 2.34 menit dalam melakukan proses pembuatan kode pada setiap fungsi. Lalu pada penelitian oleh Vasiliniuc et al. [23] menghitung efisiensi dari tugas yang diberikan kepada peserta penelitian dan menghitung waktu yang dibutuhkan untuk mengerjakan tugasnya. Penelitian ini membandingkan antara yang menggunakan AI dan yang tidak. Tetapi kedua penelitian tersebut belum mencoba untuk mengukur waktu efisiensi dari durasi setiap *prompt* dan jumlah *prompt* untuk menyelesaikan fitur atau tugas.

BAB 3 PERANCANGAN SISTEM

Bab ini akan memaparkan gambaran umum terkait perancangan penelitian yang akan dilakukan. Bab ini meliputi desain penelitian seperti alur penelitian dan metrik yang digunakan, serta desain perancangan *backend* yang akan dikembangkan.

3.1. Desain G Alur Penelitian

Fokus utama penelitian ini adalah untuk membandingkan efisiensi pada dua model AI yang tersedia pada Github Copilot Agent Mode. Secara spesifik model yang akan diujikan adalah GPT 4.1 dan Claude 3.7 Sonnet. Kedua model ini akan diujikan dalam konteks pengembangan sistem Backend Rest API. Perlu ditekankan penelitian ini **bukan** menguji masing-masing model pada masing-masing *website* seperti chatgpt.com dan claude.ai, tetapi menguji masing-masing model pada satu *platform* yang sama yaitu Github Copilot dengan mode Agent.

Studi komparatif merupakan jenis penelitian deskriptif yang bertujuan menjawab sebab akibat dari suatu fenomena dengan menganalisis faktor yang menjadi penyebab terjadinya atau munculnya suatu fenomena. Pendekatan studi komparatif dipilih karena relevan untuk membandingkan dua atau lebih kelompok terhadap variabel tertentu [24]. Metode Penelitian ini dilandaskan pada penelitian sebelumnya yang menggunakan komparasi model AI untuk melakukan untuk *Human Robot Interaction [12]*.

Penelitian ini menggunakan studi komparatif sehingga perlu menetapkan beberapa variabel[19]. Pada penelitian ini, variabel bebasnya adalah model yang digunakan. Lalu variabel tergantungnya adalah durasi setiap *prompt* dan jumlah *prompt* (sesuai dengan *Goal Question Matrix*). Variabel pengganggu dari penelitian ini, di antaranya adalah koneksi internet,

[19] Gramedia Blog, Studi Komparatif: Pengertian, Manfaat, Variabel, Macam dan Contoh (https://www.gramedia.com/literasi/studi-komparatif/?srsltid=AfmBOopUMK7wCRPHaPevDzWJZCw3039wf_qJPa5sgqaoDiDtWBej071J, diakses pada 21 Juli 2025)

keterhubungan dengan konteks sebelumnya dan potensi adanya *error* internal proyek atau *tools* yang digunakan. Untuk itu, variabel kontrol dari penelitian ini adalah penggunaan *prompt* awal yang sama persis, serta menggunakan *environment* yang sama (seperti versi VSCode yang sama, *device* yang sama, versi Github Copilot yang sama dan lain sebagainya). Variabel mediator pada penelitian ini adalah variasi fitur yang akan dibuat. Untuk menunjang hasil yang diinginkan, variabel mediasi dari penelitian ini adalah hasil kode yang dibuat, serta *log* awal dan *log* akhir pada setiap *prompt* yang dilakukan.

Pengumpulan data dilakukan melalui serangkaian eksperimen. Data yang dicatat meliputi jumlah *prompt* yang dibutuhkan untuk melakukan setiap tugas serta mengukur durasi pada setiap melakukan *prompting*. Pendekatan ini dilakukan untuk menguji performa model AI dalam skenario proyek pengembangan aplikasi di dunia nyata. Jumlah *prompt* dan durasi setiap *prompt* adalah kedua hal yang memang dirasakan langsung oleh *programmer* dalam menggunakan AI dalam *code generation*. Peneliti tidak menggunakan *dataset benchmark* yang sudah tersedia dan biasa digunakan untuk pengujian model AI seperti Leetcode dan HumanEval[20] karena dengan menggunakan *dataset* seperti itu tidak sepenuhnya merepresentasikan kompleksitas pengembangan aplikasi di dunia nyata [1]. Terlebih lagi penelitian ini akan berfokus pada pengembangan backend API, sehingga diperlukan metode yang dapat melakukan evaluasi terhadap spesifik domain tertentu[19]. Untuk memandu pengumpulan dan analisis data, peneliti menggunakan Goal Question Matrix [21] yang diadaptasi sebagai berikut:

[20] *What is HumanEval?* (https://www.deepchecks.com/glossary/humaneval/, Diakses pada 12 Mei 2025)

Untuk memandu pengumpulan dan analisis data, peneliti menggunakan Goal Question Matrix [21] yang diadaptasi sebagai berikut:

Tabel 1. Goal Question Metric

Goal	Question	Metrix
Membandingkan efisiensi model AI dalam proses *Code Generation* dari sudut pandang *programmer*	Berapa banyak *prompt* yang dibutuhkan untuk menghasilkan satu fitur ?	1. Jumlah *prompt* per fitur hingga fiturnya sesuai dengan kebutuhan 2. Jumlah *prompt* untuk menyelesaikan satu proyek
	Berapa lama durasi yang dibutuhkan AI untuk merespons *prompt* dan menyelesaikan satu fitur?	1. durasi setiap satu kali *prompt* 2. total durasi *prompt* yang dibutuhkan untuk menyelesaikan satu proyek

Teknik analisis data menggunakan statistika deskriptif. Angka yang dihasilkan dari tiap metrik akan dianalisis untuk mendapatkan nilai rata-rata, median, modus, standar deviasi, nilai minimum dan nilai maksimum. Statistika deskriptif dianggap cukup untuk membandingkan tingkat efisiensi dari kedua model AI yang diuji.

3.2. Desain Perancangan Sistem

Pelaksanaan percobaan melibatkan pengembangan sebuah aplikasi *backend* REST API dengan daftar fitur yang sebelumnya sudah ditentukan oleh peran *hustler* dan *system analyst* pada pengerjaan TA Capstone ini. Aplikasi ini akan dibuat menggunakan *framework* PHP yaitu Laravel versi 12. Laravel dipilih didasarkan pada popularitas penggunaan *framework* yang luas sampai saat ini[21]. Laravel dapat digunakan untuk membuat website salah satunya membuat Rest API. Pengembangan sistem ini dibantu sepenuhnya oleh Github Copilot Agent Mode.

3.2.1. Teknik *Prompting*

[21] Aljaz Mughal, *Laravel Usage Statistics 2024* (https://aundigital.ae/blog/laravel-usage-statistics/, Diakses pada 22 Mei 2025, 2024)

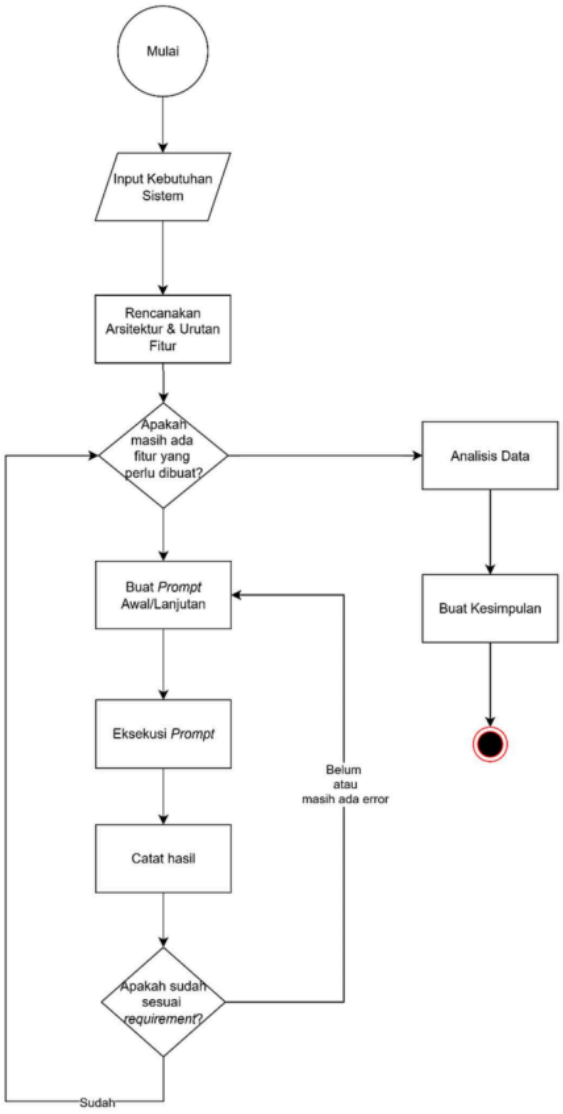
Prompt yang digunakan dibagi menjadi dua yaitu *prompt* utama untuk membuat fitur, dan *prompt* tambahan untuk *follow* up jika ada kesalahan atau ada kekurangan pada pelaksanaan tugas. Penyempurnaan *prompt* diperlukan untuk mendapatkan hasil yang lebih maksimal[25]. Strategi *prompt* utama pada penelitian ini mayoritas mengandalkan pembuatan *prompt* dengan bantuan dari AI juga. Penulis memberikan konteks permasalahan atau fitur yang ingin dibuat, kemudian AI akan membuat *prompt* versi lebih lengkap dan komprehensif untuk menghasilkan keluaran yang lebih efektif. Struktur *prompt* utama meliputi konteks proyek, spesifikasi fitur, langkah-langkah implementasi, batasan teknis dan pencatatan. Setiap model akan memiliki *prompt* utama yang sama, tetapi untuk *prompt* tambahan akan menyesuaikan dengan kondisi kode yang telah dibuat.

3.2.2. Alur Proses Pelaksanaan

Proses eksperimen akan dimulai dengan menentukan fitur yang akan dikembangkan terlebih dahulu. Setelah itu, dilakukan proses *prompting* untuk menghasilkan kode program serta *unit test* yang sesuai dengan fitur tersebut.

Jika proses hasil kodenya belum sesuai, maka dilakukan *prompting* kembali. Jika kebutuhan pada fitur tersebut sudah terpenuhi, maka di lanjut ke fitur berikutnya yang belum dibuat. Kebutuhan fitur dikatakan terpenuhi jika *unit/feature test* yang dibuat telah berhasil dijalankan. Pengujian *unit/feature* test ini juga untuk mendukung *correctness* dari pembuatan kode.

Pembuatan kode dibuat perfitur agar lebih mudah untuk di kelola. Setiap proses *prompting* yang dilakukan akan dicatat waktu proses tunggu hasilnya dan jumlah *prompt* yang dilakukan tiap tugas. Setelah semua berhasil, dilakukan analisis data dan pembuatan kesimpulan. Berikut gambaran *ffowchart*nya



Gambar 1. Alur Perancangan

Setiap tugas nantinya akan dihitung jumlah *prompt* yang dibutuhkan serta durasi pada tiap *prompt*. Setiap *prompt* dapat memiliki perintah yang bervariasi. Kebanyakan *prompt* dilakukan untuk melanjutkan proses *code generation* yang belum dan juga melakukan *prompting* untuk perbaikan kode yang belum sempurna.

Data waktu diambil dari *log* pemrosesan pada Github Copilot Chat dengan cara menghitung selisih antara waktu *log* pertama dan terakhir yang paling masuk akal pada setiap *prompt.*

Penting untuk diperhatikan bahwa pengukuran waktu ini tidak sepenuhnya merepresentasikan waktu pemrosesan AI secara keseluruhan. Hal ini disebabkan karena pada Github Copilot Agent mode, terdapat proses yang memerlukan konfirmasi dari pengguna untuk eksekusi suatu perintah seperti pembuatan *folder* atau pengeksekusian kode pada *command line*. Sehingga waktu yang digunakan pada *command line* akan terhitung juga pada total durasi pemrosesan *code generation*. Meskipun demikian, peneliti berupaya melakukan perhitungan seobjektif mungkin berdasarkan kemunculan *log* pertama dan *log* terakhir yang paling masuk akal.

3.2.3. *Fairness* dalam eksperimen

Proses penelitian ini dibuat seadil mungkin dengan beberapa kontrol yang ditetapkan.

- Perangkat lunak dan perangkat keras yang digunakan pada tiap percobaan yang sama akan menggunakan perangkat yang sama (seperti komputer, internet) dan versi perangkat lunak yang sama (seperti versi VSCode, Github Copilot, Codespace) Keadilan dalam penelitian ini didasarkan oleh penelitian yang dilakukan oleh Sobo dan lainnya [12]
- Untuk menghindari terbawanya konteks dari interaksi sebelumnya, setiap tugas baru (seperti membuat fitur baru) akan memulai *session chat* yang baru (*new chat*) pada masing-masing model AI.
- *Prompt* utama (*prompt* awal tiap tugas) akan dibuat sama persis untuk kedua model yang akan diuji.
- Sebagai upaya untuk meminimalisir bias urutan (*sequence bias*), maka diterapkan strategi proses *prompting* dengan urutan yang bergantian. Kumpulan fitur awal (manajemen pengguna sampai fitur kirim email),

pengembangan di mulai dari Claude 3.7 kemudian baru menggunakan GPT 4.1. Selain fitur yang telah disebutkan sebelumnya, proses *code generation* dimulai dari GPT 4.1 terlebih dahulu. Pendekatan ini diterapkan untuk meminimalisir salah satu model mendapat keuntungan dari konteks dan solusi yang mungkin dihasilkan oleh model sebelumnya.

BAB 4 HASIL PERCOBAAN DAN ANALISIS

Bab ini memaparkan hasil percobaan dan analisis dari eksperimen yang dikerjakan. Bab ini akan menjelaskan tentang proses pengerjaan dan hasil yang di dapat selama penelitian.

4.1. Skenario Percobaan

Eksperimen dilakukan dengan tahapan yang sudah di paparkan pada bab 3. Adapun contoh dari *prompt* utama pada setiap fitur sebagai berikut:

Tabel 2. ***Sample*** **Rencana** ***Prompt***

Fitur	Prompt
Manajemen Pengguna	Saya ingin mengembangkan fitur "Manajemen User" untuk sistem Helpdesk Ticketing. Berdasarkan #file:progress.md fitur manajemen user yang lengkap belum diimplementasikan. Fitur ini membutuhkan: Controller yang menangani operasi CRUD untuk User (UserController) Implementasi middleware role untuk memastikan hanya admin yang dapat mengakses fitur ini Endpoint API untuk mengelola user dan melihat statistik user Sesuai dengan #file:api.md endpoint yang perlu dibuat adalah: GET /users untuk mendapatkan semua user (Admin only) GET /users/{id} untuk mendapatkan detail user (Admin only) PATCH /users/{id} untuk memperbarui informasi user (Admin only) PATCH /users/{id}/role untuk memperbarui role user (Admin only) DELETE /users/{id} untuk menghapus user (Admin only) GET /users/statistics untuk mendapatkan statistik user (Admin only) Mohon bantu saya untuk: Membuat implementasi backend untuk fitur manajemen user dengan memastikan hanya admin yang bisa mengakses

	Membuat unit test dan feature test untuk memastikan fitur berfungsi Mendokumentasikan perubahan di #file:progress.md kententuan ada pada #file:copilot-instruction.md lifecycle laravel 12 #fetch https://laravel.com/docs/12.x/lifecycle Lakukan langkah demi langkah
Tambah Statistik Ticket pada user	agent Saya ingin mengedit fitur manajemen user agar menambahkan statistik dan daftar ticket dari setiap user ketika get all user dan get user by id. Berdasarkan progress.md, fitur manajemen user sudah diimplementasikan tetapi belum memiliki statistik detail dan daftar ticket per user. Fitur yang perlu ditambahkan: - Statistik jumlah ticket yang telah dibuat oleh user - Daftar ticket (id, judul, status) yang dibuat oleh user tersebut - Informasi ini harus ditampilkan pada endpoint GET /users dan GET /users/{id} Sesuai dengan api.md, endpoint yang perlu dimodifikasi adalah: - GET /users untuk mendapatkan semua user dengan tambahan statistik per user - GET /users/{id} untuk mendapatkan detail user dengan tambahan statistik dan daftar ticket dari user tersebut Mohon bantu saya untuk: 1. Memodifikasi UserController untuk menambahkan statistik ticket dan daftar ticket pada response 2. Menambahkan relasi antara User model dan Ticket model jika belum ada 3. Mengoptimalkan query database untuk menghindari N+1 problem saat mengambil data ticket 4. Membuat unit test dan feature test untuk memastikan fitur berfungsi dengan benar 5. Mendokumentasikan perubahan di progress.md Lakukan langkah demi langkah dengan mempertimbangkan lifecycle Laravel 12 untuk memastikan implementasi yang efisien dan sesuai dengan best practice.

Manajemen Chat	agent Saya ingin mengembangkan fitur ""Manajemen Chat"" untuk sistem Helpdesk Ticketing. Berdasarkan progress.md, fitur chat belum diimplementasikan. Fitur ini membutuhkan: - Menggantikan sistem ""feedback"" yang saat ini ada pada setiap ticket - Model ChatMessage untuk menyimpan pesan chat - Model ChatAttachment untuk menangani lampiran pada chat - Controller untuk operasi CRUD pada pesan chat - Implementasi real-time chat (opsional, jika memungkinkan) Sesuai dengan api.md, endpoint yang perlu dibuat adalah: - GET /tickets/{id}/chat untuk mendapatkan semua pesan chat pada ticket tertentu - POST /tickets/{id}/chat untuk menambahkan pesan chat baru pada ticket - POST /tickets/{id}/chat/attachment untuk mengunggah lampiran pada chat - DELETE /tickets/{id}/chat/{message_id} untuk menghapus pesan chat (oleh penulis pesan saja) - GET /tickets/{id}/chat/attachments untuk mendapatkan semua lampiran chat Mohon bantu saya untuk: 1. Membuat migrasi database untuk tabel chat_messages dan chat_attachments 2. Membuat model ChatMessage dan ChatAttachment dengan relasi yang sesuai 3. Membuat controller untuk menangani operasi chat 4. Implementasi otorisasi agar pengguna hanya bisa mengakses chat ticket yang relevan dengan mereka 5. Integrasi dengan sistem notifikasi yang sudah ada agar pengguna mendapat notifikasi saat ada pesan chat baru 6. Membuat unit test dan feature test untuk memastikan fitur berfungsi dengan benar 7. Mendokumentasikan perubahan di progress.md Lakukan langkah demi langkah dengan mempertimbangkan lifecycle Laravel 12 untuk memastikan implementasi yang efisien dan sesuai dengan best practice.
Manajemen FAQ	agent

	Saya ingin mengembangkan fitur ""Manajemen FAQ"" untuk sistem Helpdesk Ticketing. Berdasarkan progress.md, fitur FAQ belum diimplementasikan. Fitur ini membutuhkan: - Model FAQ untuk menyimpan pertanyaan yang sering diajukan dan jawabannya - Controller untuk operasi CRUD pada FAQ - Kemampuan untuk mengkonversi ticket yang sudah ada menjadi FAQ - Implementasi otorisasi agar hanya admin yang dapat mengelola FAQ - Endpoint API untuk mengakses FAQ secara publik (tanpa autentikasi) Sesuai dengan best practice API, endpoint yang perlu dibuat adalah: - GET /faqs untuk mendapatkan semua FAQ (publik) - GET /faqs/{id} untuk mendapatkan detail FAQ (publik) - POST /faqs untuk membuat FAQ baru (admin only) - POST /tickets/{id}/convert-to-faq untuk mengkonversi ticket menjadi FAQ (admin only) - PATCH /faqs/{id} untuk mengupdate FAQ (admin only) - DELETE /faqs/{id} untuk menghapus FAQ (admin only) - GET /faqs/categories untuk mendapatkan kategori FAQ (publik) Mohon bantu saya untuk: 1. Membuat migrasi database untuk tabel faqs dengan kolom-kolom yang diperlukan 2. Membuat model FAQ dengan relasi ke Category dan User 3. Membuat controller FAQController untuk menangani operasi CRUD 4. Mengimplementasikan endpoint untuk konversi ticket menjadi FAQ 5. Membuat middleware otorisasi agar hanya admin yang bisa mengelola FAQ 6. Membuat unit test dan feature test untuk memastikan fitur berfungsi dengan benar 7. Mendokumentasikan perubahan di progress.md Lakukan langkah demi langkah dengan mempertimbangkan lifecycle Laravel 12 untuk memastikan implementasi yang efisien dan sesuai dengan best practice.

Sebagai catatan, fitur autentikasi dijadikan kondisi awal proyek sebelum pengujian. Sehingga pada saat eksperimen proses *prompting*, fitur yang sudah ada sejak awal adalah autentikasi. Artinya, fitur yang akan di uji adalah fitur manajemen pengguna, tiket, notifikasi dan lain sebagainya.

4.2. Hasil Pengumpulan data

Proses pengumpulan data tidak sepenuhnya berjalan dengan lancar. Beberapa kali sering terjadi masalah yang tidak berkaitan dengan model AI seperti stabilitas jaringan, masalah pada *code editor* bahkan isu teknis yang ada pada lingkungan pengembangan. Oleh karena itu, Data yang disajikan pada bagian ini merupakan data yang berhasil mengatasi kendala-kendala tersebut.

Pengumpulan data efisiensi diambil berdasarkan metrix yang telah ditetapkan pada Goal Question Metrix pada bab 3

Tabel 3. Hasil Percobaan

GPT 4.1				
Kode	**Fitur**	**Jumlah *Prompt***	**Rata-rata waktu (Menit)**	**Total Waktu (Menit)**
F1	Manajemen Pengguna	2	00:54.8	01:49.6
F2	Manajemen Kategori & Subkategori	4	01:27.3	05:49.2
F3	Manajemen Tiket	6	01:18.0	07:47.9
F4	Manajemen Notifikasi	3	01:28.5	04:25.6
F5	Tambah Statistik Ticket pada user	3	01:08.6	03:25.9
F6	Manajemen Chat	2	01:40.3	03:20.5
F7	Manajemen FAQ	5	01:39.3	08:16.4
F8	Kirim email	4	01:30.8	06:03.1
F9	Tambah Token Ticket	2	02:56.2	05:52.3
F10	Tambah informasi chat	2	01:46.0	03:32.0
F11	Tambah Prioritas	3	01:29.0	04:27.1
F12	Log Aktivitas	2	01:58.6	03:57.1
	Keseluruhan	38	1:32.808	58:46.694
	Min	2	00:54.8	01:49.6
	Max	6	02:56.2	08:16.4
	Median	3	01:29.9	04:26.4
	Standar Deviasi	1.280190958	0.000334784	0.001269856

Claude 3.7				
Kode	Fitur	Jumlah *Prompt*	Rata-rata waktu (Menit)	Total Waktu (Menit)
F1	Manajemen Pengguna	4	03:43.6	14:54.3
F2	Manajemen Kategori & Subkategori	4	02:47.0	11:07.9
F3	Manajemen Tiket	4	03:06.8	12:27.1
F4	Manajemen Notifikasi	6	03:23.0	20:17.9
F5	Tambah Statistik Ticket pada user	3	01:34.6	04:43.8
F6	Manajemen Chat	5	03:54.0	19:30.0
F7	Manajemen FAQ	3	03:18.1	09:54.2
F8	Kirim email	3	01:50.4	05:31.2
F9	Tambah Token Ticket	3	02:44.8	08:14.3
F10	Tambah informasi chat	4	02:44.5	10:58.1
F11	Tambah Prioritas	3	04:25.6	13:16.9
F12	Log Aktivitas	2	03:17.8	06:35.7
	Keseluruhan	44	3:07.530	137:31.313
	Min	2	01:34.6	04:43.8
	Max	6	04:25.6	20:17.9
	Median	3.5	03:12.3	11:03.0
	Standar Deviasi	1.027402334	0.000538021	0.003332316

4.3. Pembahasan Hasil

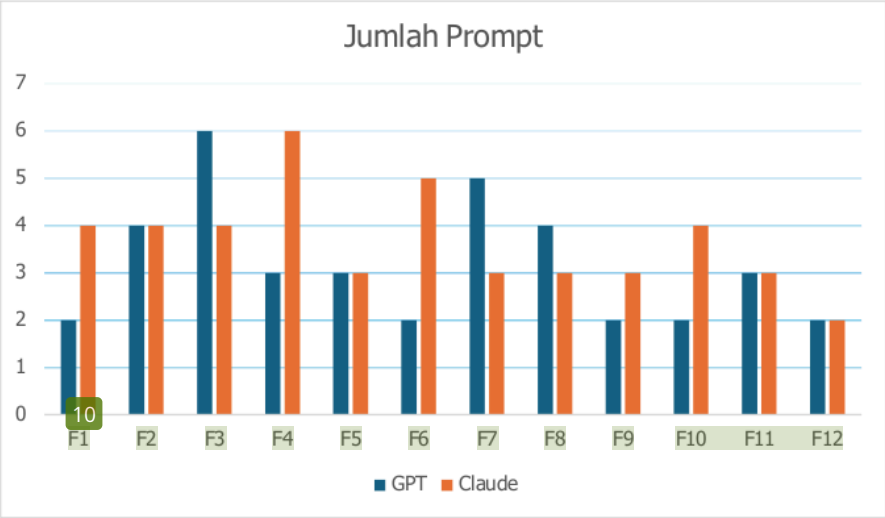
Hasil dari kode yang dibuat pada percobaan ini telah dilakukan dengan mempertimbangkan *correctness*, sehingga seluruh fitur yang diujikan pada penleitian ini sudah lolos unit/feature test yang dibuat saat proses *code generation* yang dilakukan oleh LLM. Bagian ini akan lebih lanjut membahas tentang hasil temuan yang didapatkan selama proses eksperimen.

4.3.1. Analisis Metrik Efisiensi Kuantitatif

Berikut analisis dari hasil percobaan terhadap 12 tugas yang dikembangkan. Analisis ini meliputi jumlah *prompt* serta waktu yang dibutuhkan pada proses pembuatan kode.

Analisis Jumlah *Prompt*

Dari proses *prompting* yang dilakukan, didapat grafik yang menunjukkan jumlah *prompt* yang diperlukan masing-masing model pada pengerjaan setiap tugas.



Gambar 2. Diagram Batang Jumlah *Prompt*

Grafik tersebut memperlihatkan bahwa secara umum, model GPT 4.1 memiliki jumlah *prompt* yang lebih sedikit dibandingkan model Claude 3.7. Pada GPT 4.1, secara keseluruhan jumlah *prompt* yang dibutuhkan adalah 38 kali, sedangkan pada Claude membutuhkan 44 kali *prompt* untuk menyelesaikan semua tugas yang diberikan peda percobaan ini.

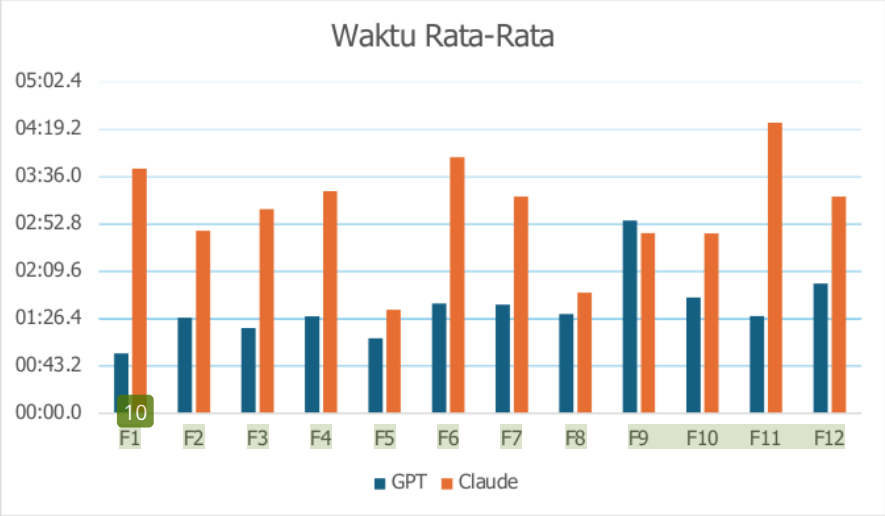
Berdasarkan percobaan yang telah dilakukan, penulis menyimpulkan bahwa kompleksitas fitur, konteks proyek, serta solusi yang ditawarkan oleh AI sangat berpengaruh pada jumlah *prompt* yang dibutuhkan. Hal ini juga sesuai dengan literatur yang digunakan oleh Sobo et al yang menjelaskan kalau kompleksitas tugas berpengaruh pada kinerja LLM [12]. Ada beberapa fitur spesifik yang mengakibatkan GPT lebih banyak melakukan *prompt* daripada Claude seperti manajemen tiket, FAQ dan kirim email. Fitur manajemen tiket dan FAQ pada model GPT membutuhkan jumlah *prompt* lebih banyak karena pada kedua fitur tersebut di model GPT, kode yang dihasilkan tidak langsung terbuat semua. Sehingga peneliti harus melakukan *prompt* tambahan untuk menyelesaikan semua bagian yang belum dikerjakan. Sementara pada fitur kirim email di model GPT, terdapat *error* pada

pembuatan kode yang menyebabkan AI berulang kali mencari solusi dari permasalahan tersebut.

Secara umum jumlah *prompt* terbanyak secara akumulatif ada pada Claude. Karena pada Claude, keluaran yang dihasilkan lebih komprehensif dan lebih lengkap daripada menggunakan GPT. Meskipun demikian, dari data yang telah dipaparkan menunjukkan bahwa performa dari masing-masing model pada jumlah *prompt* yang dilakukan tidak bersifat mutlak. tetapi juga sangat bergantung pada jenis tugas yang dikerjakan.

Analisis Waktu Pengerjaan

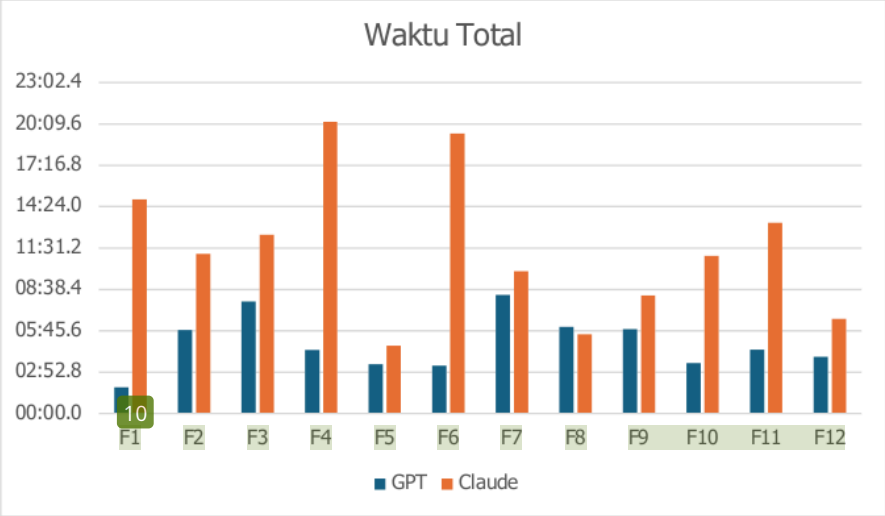
Berdasarkan waktu yang dihasilkan oleh masing-masing model melalui log (penjelasan cara mendapatkan data ada di bab 3 bagian Alur Proses Pelaksanaan), dapat dilihat bahwa GPT memiliki waktu eksekusi *prompt* yang lebih konsisten daripada Claude. Berikut merupakan grafik yang merepresentasikan waktu yang dibutuhkan tiap model proses *code generation* pada setiap tugas.



Gambar 3. Diagram Batang Waktu Rata-rata Per Fitur

Berdasarkan data tersebut, dapat disimpulkan bahwa waktu yang dibutuhkan oleh model Claude 3.7 lebih banyak daripada GPT 4.1. Claude 3.7 mendapati

rata-ratanya adalah 3 menit, sedangkan GPT 4.1 memiliki rata-rata selama sekitar 1,5 menit saja.



Gambar 4. Diagram Batang Waktu Total Tiap Fitur

Jika ditotal secara keseluruhan, waktu yang dibutuhkan Claude lebih banyak dibandingkan GPT. Waktu yang dibutuhkan Claude untuk menyelesaikan seluruh tugas ini adalah 137 menit, sedangkan pada GPT hanya butuh waktu 58 menit. Meskipun GPT memiliki durasi setiap *prompt* yang lebih cepat dalam berbagai macam fitur, ada banyak faktor yang di luar kendali dari model tersebut yang mempengaruhi waktu tunggunya.

Ada beberapa faktor eksternal yang mempengaruhi durasi waktu untuk menyelesaikan suatu *prompt*. Di antara faktornya sebagai berikut:

a. Waktu Eksekusi pada Terminal

Beberapa kasus, AI memilih untuk melakukan eksekusi beberapa hal pada terminal, seperti operasi *file* atau bahkan melakukan pengujian (*testing*) yang dapat menyebabkan waktu tunggunya menjadi lebih lama.

b. Waktu tunggu *trigger* dari pengguna

Pada Github Copilot Agent Mode, terkadang terdapat eksekusi *prompt* yang memerlukan persetujuan dari pengguna sebelum melakukan eksekusinya. seperti pembuatan folder atau lainnya yang dapat mempengaruhi waktu yang diperlukan untuk melakukan *prompting*

c. Kompleksitas konteks proyek dan fitur yang akan dibuat

Hal ini sangat berpengaruh pada durasi *prompt*. semakin kompleks fitur yang dibuat atau semakin kompleks *project* yang sudah ada, maka waktu untuk eksekusi *prompt*-nya semakin lama karena AI harus membaca konteks dari proyek tersebut, memikirkan solusi yang harus dilakukan pada proyek tersebut, dan membuat kode sesuai dengan kebutuhan.

d. Batas maksimum durasi setiap *prompt* pada model

Pada GPT, durasi paling lama dalam satu *prompt* itu ada di waktu sekitar dua setengah menit untuk satu kali *prompt*. Jika lebih dari 2 setengah menit atau sampai kurang dari tiga menit, AI akan meminta untuk melakukan *prompting* lagi (*Continue Iteration*). Sedangkan pada Claude paling lama 6 menit lebih untuk satu kali *prompt* sebelum akhirnya meminta untuk *prompt* berikutnya.

4.3.2. Analisis Kualitatif

Secara umum, peneliti mendapatkan hasil kode yang dibuat oleh masing-masing model dapat berbeda. Sebagai contoh, fitur manajemen tiket pada kedua model menghasilkan kode yang berbeda dan pendekatan yang berbeda, walaupun tetap sesuai dengan kebutuhan fitur. Kemudian pada fitur notifikasi, meskipun *prompt* awal yang dibuat sama, tetapi konteks proyek yang mulai berbeda menyebabkan kedua model tersebut tidak menghasilkan metode yang sama dalam membuat kode manajemen notifikasi. Pada Claude, fitur notifikasi menggunakan metode *service* biasa, sedangkan pada GPT menggunakan metode *events & listener.* Meskipun metode yang digunakan berbeda, setidaknya hasilnya kurang lebih sesuai dengan kebutuhan.

Peneliti juga menemukan beberapa kesimpulan secara kualitatif pada masing-masing model sebagai berikut:

GPT 4.1

- Efektivitas lebih tinggi untuk mengerjakan fitur yang tidak begitu kompleks. Kode yang dihasilkan terkadang ringkas dan langsung dapat menjawab kebutuhan.
- Akurasi dalam membuat kodenya konsisten untuk berbagai kesulitan dan juga di dukung dengan penelitian sebelumnya *pada Generate Code* untuk LLM4DS pada ChatGPT [19].
- Terkadang terdapat kode yang belum lengkap dibuat, jadi peneliti harus perlu melakukan *prompting* ulang untuk melengkapi kekurangan kode tersebut
- Kesulitan melakukan *debuging* kompleks, terutama jika dihadapkan pada *bug* yang terjadi pada proyek tersebut. Terkadang model ini melakukan solusi yang berulang ulang dan tidak efektif

Claude 3.7 Sonnet

- Bagus untuk fitur yang lebih kompleks, biasanya menghasilkan kode yang lebih lengkap daripada GPT.
- Pembuatan kode terkadang dilakukan secara berlebihan. Sebagai contoh pada fitur Manajemen kategori, yang peneliti butuhkan hanya fungsi *get* dan *post* saja, tetapi oleh model ini justru membuat lengkap CRUD-nya
- Anomali saat melakukan *unit test*. Model ini dapat menghasilkan test case yang cukup banyak. bahkan setelah semua fungsionalitas sudah terbukti benar dengan *unit test* yang dibuat sebelumnya, model ini tiba-tiba membuat *test case* yang lain yang mengakibatkan kode menjadi salah sehingga merusak *project* yang sudah benar. Penulis bahkan

harus menghentikan paksa proses *prompting* agar *error*-nya tidak semakin melebar.

Pada penelitian ini juga, pada *functional correctness* dilakukan validasi dengan menggunakan *unit/feature test* yang juga dibuat oleh AI. Fitur atau *task* yang diberikan dianggap selesai hanya ketika semua *test* yang sudah dibuat berhasil dijalankan dengan baik. Sehingga dari 12 *task* pada penelitian ini berhasil memenuhi kriteria dari *test* yang sudah ditentukan.

Secara umum, model GPT sepertinya sudah didukung dengan baik oleh Github Copilot. Hal itu disebabkan karena ketika *prompting* dilakukan, Github Copilot dapat menampilkan proses pengkodean dari *file* yang diubah. Sedangkan pada Claude tidak secara langsung menampilkan perubahannya, tetapi perubahan tersebut muncul ketika satu file tersebut sudah sesleai melakukan *code generation.*

BAB 5 KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan hasil eksperimen pengembangan *backend* Rest API menggunakan Github Copilot dengan mamastikan *correctness* dari kode yang dibuat bedasarkan *unit/feature test*, kesimpulan utama untuk menjawab rumusan masalah yaitu sebagai berikut:

a. Perbandingan jumlah *prompt*

Berdasarkan data yang didapat, model Claude 3.7 memerlukan proses *prompt* yang lebih banyak dibandingkan dengan GPT 4.1 –Claude 3.7 lebih banyak 6 *prompt* daripada GPT– sehingga dapat disimpulkan bahwa GPT 4.1 lebih efisien dibanding model Claude 3.7 Sonnet. Akan tetapi, Claude 3.7 menyelesaikan tugas lebih andal dibandingkan GPT 4.1, terutama untuk fitur yang lebih kompleks.

b. Perbandingan durasi setiap *prompt*

Pada durasi setiap *prompt,* GPT secara signifikan lebih cepat dibandingkan Claude 3.7. Angka tersebut dapat dilihat dari rata-rata maupun akumulasi keseluruhan, yaitu perbedaannya adalah 78 menit untuk waktu total dan 1,5 menit untuk rata-rata. Hal ini disebabkan karena durasi maksimal yang dibutuhkan GPT 4.1 pada satu kali *prompt* lebih singkat dibanding Claude 3.7

c. Kesimpulan model yang menangani efisiensi terbaik

Pada penelitian ini, GPT 4.1 terbukti lebih efisien untuk melakukan tugas yang ringkas dan tidak begitu besar. Keunggulan ini dilihat dari jumlah *prompt* yang umumnya lebih sedikit serta durasi setiap *prompt* yang lebih singkat.

Meskipun durasi *prompt* yang dibutuhkan lebih banyak, Claude 3.7 lebih efektif untuk melakukan pembuatan kode yang berskala lebih besar pada satu kali *prompt*. Perlu dicatat bahwa efisiensi antara model AI bergantung juga pada tingkat kompleksitas fitur AI konteks yang diberikan pada AI.

5.2. Batasan Penelitian

Dari percobaan yang dilakukan, peneliti menyadari bahwa penelitian ini memiliki beberapa keterbatasan yang perlu dipertimbangkan. Keterbatasan tersebut di antaranya:

a. Lingkup Proyek yang terbatas

Penelitian ini hanya berfokus pada lingkup pengembangan backend Rest API menggunakan Laravel dengan tugas yang spesifik. Oleh karena itu, untuk generalisasi hasil penelitian ke bahasa pemrograman, *framework* atau pengembangan yang lain perlu memerlukan validasi lebih lanjut.

b. Potensi Subjektivitas Peneliti

Proses evaluasi dan perhitungan jumlah *prompt* yang dilakukan oleh peneliti besar kemungkinan mengandung hasil yang bersifat subjektif. Meskipun demikian, peneliti berusaha sebisa mungkin mengupayakan proses penelitian dengan objektif.

c. Metrik yang terbatas

Pengukuran efisiensi berfokus pada jumlah *prompt* dan durasi setiap *prompt.* Penelitian ini belum menganalisis secara mendalam aspek lainnya seperti kualitas dari kode, *maintability*, keamanan, performansi dan lain sebagainya.

d. Perkembangan Teknologi

Mengingat kecepatan evolusi AI yang sangat pesat, hasil penelitian ini merefleksikan performa model pada versi dan waktu tertentu. Relevansi dari performa model AI dapat berubah pada waktu dan versi tertentu, sehingga relevansi dari penelitian ini dapat berkurang seiring berjalannya waktu.

e. Kesalahan minor

Pada proses *code generation* ini tidak luput pada kesalahan minor sehingga beberapa bagian yang tidak sepenuhnya sama persis dengan *requirement* awal. Oleh karena itu, kriteria keberhasilan pada pengujiannya ditentukan

berdasarkan keberhasilan dari skenario pengujian unit test yang dihasilkan oleh AI.

5.3. Saran

Setiap model memiliki kekurangan dan kelebihannya masing-masing. Alangkah lebih baik lagi jika di kombinasikan penggunaannya berdasarkan tingkat kompleksitas dari fitur yang dibuat dengan karakteristik dari masing-masing model AI dibanding memilih satu model AI untuk mengerjakan semua proyek.

Untuk memperdalam penelitian tentang bidang ini, berikut beberapa arah penelitian yang dapat dieksplorasi ke depannya:

a. Evaluasi pada model AI terbaru

Mengingat perkembangan AI yang sangat pesat, penelitian selanjutnya sangat dianjurkan untuk melakukan evaluasi ulang AI pada model terbaru untuk memastikan relevansi temuan.

b. Percobaan dengan konteks berbeda

Untuk menguji generalisasi temuan, penelitian ini dapat di replikasi dengan *framework*, bahasa pemrograman, atau konteks proyek yang berbeda yang berbeda. Pengujian ini dilakukan untuk mengetahui konsistensi dari tingkat efisiensinya pada model yang diujikan.

c. Perluas Metrik Evaluasi

Penelitian ke depan disarankan untuk memperluas metrik pengujian di luar efisiensi yang sudah ditentukan sebelumnya. Metrik yang dimaksud diharapkan dapat mencakup analisis terhadap kualitas kode yang dihasilkan seperti performansi kode, kompleksitas, kemudahan, keamanan dan lain sebagainya.

d. Penerapan MCP pada pengujian model

Akhir-akhir ini muncul teknologi yang bernama Model Context Protocol pada ranah *code generation* yang dapat meningkatkan efektivitas dan efisiensi proses *code generation* menggunakan AI. Penelitian selanjutnya dapat mengukur dampak dari penerapan MCP pada proses *Code Generation*.

e. Penggunaan bahasa inggris untuk melakukan *prompting*

Penelitian yang dilakukan saat ini masih menggunakan bahasa Indonesia sebagai bahasa yang digunakan pada proses *prompting*. Untuk penelitian berikutnya dapat dicoba menggunakan bahasa lain pada penggunaan LLM.

f. Validasi dengan lebih dari satu developer

Salah satu keterbatasan penelitian ini adalah dilakukan oleh satu orang peneliti. Sedangkan pada *prompt* lanjutan -- untuk *follow up* jika kode yang dibuat belum memenuhi *requirement*-- sangat bergantung pada *developer* yang melakukannya. Oleh karena itu perlu melibatkan lebih dari seatu *developer* untuk menjaga konsistensi dan reabilitas temuan.

DAFTAR PUSTAKA

[1] Y. Grace, E. Wen, and H. Sun, "Comparative Analysis of ChatGPT-4 and Gemini Advanced in Erroneous Code Detection and Correction," 2024.

[2] R. Dwi Natasya, "IMPLEMENTASI ARTIFICIAL INTELLIGENCE (AI) DALAM TEKNOLOGI MODERN," *Jurnal Komputer dan Teknologi Sains (KOMTEKS)*, vol. 2, no. 1, pp. 22–24, Dec. 2023, [Online]. Available: https://ojs.unm.ac.id/pengabdi/article/view/46

[3] M. K. Siam, H. Gu, and J. Q. Cheng, "Programming with AI: Evaluating ChatGPT, Gemini, AlphaCode, and GitHub Copilot for Programmers," *IEEE International Conference on Program Comprehension*, vol. 2022-March, pp. 36–47, Nov. 2024, Accessed: Apr. 30, 2025. [Online]. Available: http://arxiv.org/abs/2411.09224

[4] J. Jiang, F. Wang, J. Shen, S. Kim, and S. Kim, "A Survey on Large Language Models for Code Generation," Jun. 2024, [Online]. Available: http://arxiv.org/abs/2406.00515

[5] B. Yetiştiren, I. Özsoy, M. Ayerdem, and E. Tüzün, "Evaluating the Code Quality of AI-Assisted Code Generation Tools: An Empirical Study on GitHub Copilot, Amazon CodeWhisperer, and ChatGPT," Apr. 2023, [Online]. Available: http://arxiv.org/abs/2304.10778

[6] R. Pandey, P. Singh, R. Wei, and S. Shankar, "Transforming Software Development: Evaluating the Efficiency and Challenges of GitHub Copilot in Real-World Projects," Jun. 2024, [Online]. Available: http://arxiv.org/abs/2406.17910

[7] K. Mohamed, M. Yousef, W. Medhat, E. H. Mohamed, G. Khoriba, and T. Arafa, "Hands-on analysis of using large language models for the auto evaluation of programming assignments," Feb. 01, 2025, *Elsevier Ltd*. doi: 10.1016/j.is.2024.102473.

[8] M. Cataldo and J. D. Herbsleb, "Factors leading to integration failures in global feature-oriented development: an empirical analysis," in *2011 33rd International Conference on Software Engineering (ICSE)*, 2011, pp. 161–170. doi: 10.1145/1985793.1985816.

[9] M. Chen *et al.*, "Evaluating Large Language Models Trained on Code," Jul. 2021, [Online]. Available: http://arxiv.org/abs/2107.03374

[10] D. Huang, Y. Qing, W. Shang, H. Cui, and J. M. Zhang, "EffiBench: Benchmarking the Efficiency of Automatically Generated Code," May 2025, [Online]. Available: http://arxiv.org/abs/2402.02037

[11] L. B. Heitz, J. Chamas, and C. Scherb, "Evaluation of the Programming Skills of Large Language Models," May 2024, [Online]. Available: http://arxiv.org/abs/2405.14388

[12] A. Sobo, A. Mubarak, A. Baimagambetov, and N. Polatidis, "Evaluating LLMs for Code Generation in HRI: A Comparative Study of ChatGPT, Gemini, and Claude," *Applied Artificial Intelligence*, vol. 39, no. 1, 2025, doi: 10.1080/08839514.2024.2439610.

[13] N. Nguyen and S. Nadi, "An Empirical Evaluation of GitHub Copilot's Code Suggestions," in *Proceedings - 2022 Mining Software Repositories Conference, MSR 2022*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 1–5. doi: 10.1145/3524842.3528470.

[14] C. Sriwilailak, Y. Higo, P. Lapvikai, C. Ragkhitwetsagul, and M. Choetkiertikul, "Autorepairability of ChatGPT and Gemini: A Comparative Study."

[15] S. Schulhoff *et al.*, "The Prompt Report: A Systematic Survey of Prompt Engineering Techniques," Jun. 2024, [Online]. Available: http://arxiv.org/abs/2406.06608

[16] H. Xu *et al.*, "Large Language Models for Software Engineering: A Systematic Literature Review," *ACM Transactions on Software Engineering and Methodology*, Dec. 2024, doi: 10.1145/3695988.

[17] F. Liu *et al.*, "Exploring and Evaluating Hallucinations in LLM-Powered Code Generation," Apr. 2024.

[18] Z. C. Ani, Z. A. Hamid, and N. N. Zhamri, "The Recent Trends of Research on GitHub Copilot: A Systematic Review," 2024, pp. 355–366. doi: 10.1007/978-981-99-9589-9_27.

[19] N. Nascimento, E. Guimaraes, S. S. Chintakunta, and S. A. Boominathan, "LLM4DS: Evaluating Large Language Models for Data Science Code Generation," Nov. 2024, [Online]. Available: http://arxiv.org/abs/2411.11908

[20] R. T. Fielding and R. N. Taylor, "Architectural styles and the design of network-based software architectures," 2000.

[21] Ian. Sommerville, *Software engineering*. Pearson, 2011.

[22] C. Wang *et al.*, "Teaching Code LLMs to Use Autocompletion Tools in Repository-Level Code Generation," Jul. 2024, [Online]. Available: http://arxiv.org/abs/2401.06391

[23] M.-S. Vasiliniuc and A. Groza, "Case Study: Using AI-Assisted Code Generation In Mobile Teams," Sep. 2023, [Online]. Available: http://arxiv.org/abs/2308.04736

[24] W. P. Zayu, H. Herman, and G. Vitri, "Studi Komparatif Pelaksanaan Tugas Besar Perencanaan Geometrik Jalan Secara Daring Dan Luring," *Jurnal Penelitian Dan Pengkajian Ilmiah Eksakta*, vol. 2, no. 1, pp. 92–96, Feb. 2023, doi: 10.47233/jppie.v2i1.762.

[25] J. D. Velásquez-Henao, C. J. Franco-Cardona, and L. Cadavid-Higuita, "Prompt Engineering: a methodology for optimizing interactions with AI-Language Models in the field of engineering," *Dyna (Medellin)*, vol. 90, no. 230, pp. 9–17, Nov. 2023, doi: 10.15446/dyna.v90n230.111700.

LAMPIRAN

Lampiran 01. Tabel perhitungan

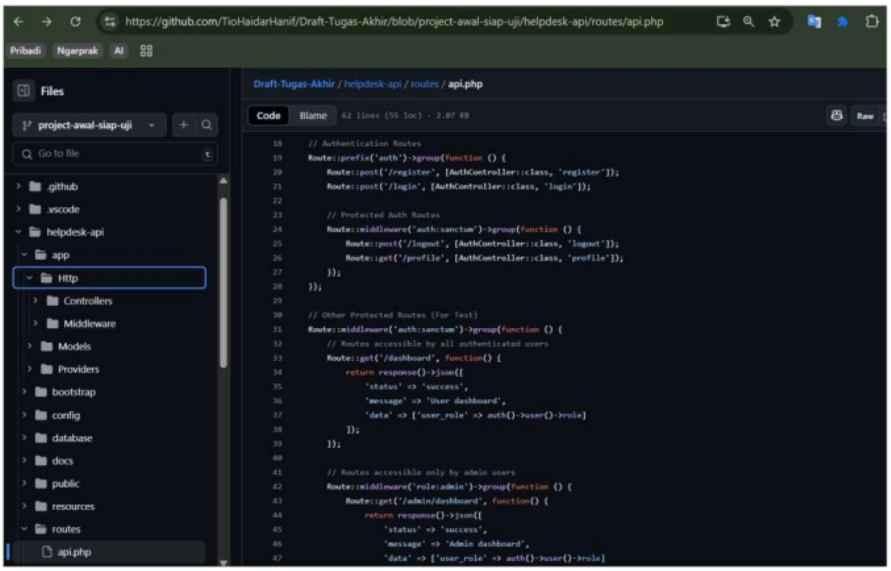
GPT 4.1							
Fitur	**Prompt**	**Log**	**Waktu (MS)**	**Readable Time**	**Keterangan**	**Hasil**	**Readable**
Manajemen Pengguna	Saya ingin mengembangkan fitur "Manajem	2025-05-23 0	96656	1:36.656	Jumlah Iterasi	2	
	Continue Iteration	2025-05-23 0	12894	0:12.894	Waktu Total	109550	1:49.550
					Rata rata	54775	0:54.775
Manajemen Kategori dan Sub	Saya ingin mengembangkan fitur "Manajem	2025-05-24 0	60107	1:00.107	Jumlah Iterasi	4	
	Continue Iteration	2025-05-24 0	142173	2:22.173	Waktu Total	349199	5:49.199
	Continue Iteration	2025-05-24 0	104827	1:44.827	Rata rata	87300	1:27.300
	pada fitur manajemen kategori, tolong ses	2025-05-24 0	42092	0:42.092			
Manajemen Ticket	agentSaga ingin mengembangkan fitur "Ma	2025-05-24 0	71459	1:11.459	Jumlah Iterasi	6	
	Continue Iteration	2025-05-24 0	92672	1:32.672	Waktu Total	467889	7:47.889
	lakukan implementasi pada #file:TicketCo	2025-05-24 0	153655	2:33.655	Rata rata	77982	1:17.982
	Continue Iteration	2025-05-24 0	60675	1:00.675			
	Belum semua API pada #file:api.php untuk	2025-05-24 0	49996	0:49.996			
	tolong perbaiki agar attachment dari ticket	2025-05-24 0	39432	0:39.432			
Manajemen Notifikasi	Saya ingin mengembangkan fitur "Manajem	2025-05-26 0	66047	1:06.047	Jumlah Iterasi	3	
	Continue Iteration	2025-05-26 0	158076	2:38.076	Waktu Total	265585	4:25.585
	Continue Iteration	2025-05-26 0	41462	0:41.462	Rata rata	88528	1:28.528
Tambah Statistik Ticket pada	agentSaga ingin mengedit fitur manajemen	2025-05-29 0	70847	1:10.847	Jumlah Iterasi	3	
TioHaidarHanif: agentSaga ingin mer	Continue Iteration	2025-05-29 0	106113	1:46.113	Waktu Total	205881	3:25.881
	pada endpoint user/{id}, di setiap ticket yan	2025-05-29 0	28921	0:28.921	Rata rata	68627	1:08.627
Manajemen Chat	agentSaga ingin mengembangkan fitur ""M	2025-05-30 13	168976	2:48.976	Jumlah Iterasi	2	
TioHaidarHanif: agentSaga ingin mer	Continue Iteration	2025-05-30 13	31553	0:31.553	Waktu Total	200529	3:20.529
					Rata rata	100265	1:40.265
Manajemen FAQ	agentSaga ingin mengembangkan fitur ""M	2025-05-30 14	112670	1:52.670	Jumlah Iterasi	5	
	lanjutkan ke langkah selanjutnya	2025-05-30 14	135588	2:15.588	Waktu Total	496404	8:16.404
	Continue Iteration	2025-05-30 14	72971	1:12.971	Rata rata	99281	1:39.281
	lanjutkan untuk membuat test pada setiap	2025-05-30 15	123723	2:03.723			
	ubah urutan posisi Route::get('/faqs/categ	2025-05-30 15	51452	0:51.452			
Kirim email	agentSaga ingin mengembangkan fitur "Kir	2025-06-08 14	76506	1:16.506	Jumlah Iterasi	4	
TioHaidarHanif: agentSaga ingin mer	Continue Iteration	2025-06-08 14	94985	1:34.985	Waktu Total	363143	6:03.143
	Continue Iteration	2025-06-08 14	169718	2:49.718	Rata rata	90786	1:30.786
	file #file:ManualMailTest.php dihapus saja,	2025-06-08 14	21934	0:21.934			
Tambah Token Ticket	Saya ingin mengembangkan fitur "Token Rahasia	[illegible]	156876	2:36.876	Jumlah Iterasi	2	
	Continue Iteration	[illegible]	195429	3:15.429	Waktu Total	352305	5:52.305
					Rata rata	176153	2:56.153
Tambah informasi chat	Saya ingin mengembangkan fitur "Informasi Jumla	[illegible]	141156	2:21.156	Jumlah Iterasi	2	
TioHaidarHanif: Saya ingin mengembangka	Continue Iteration	[illegible]	70816	1:10.816	Waktu Total	211972	3:31.972
					Rata rata	105986	1:45.986
Tambah Prioritas	Saya ingin mengembangkan fitur "Prioritas Ticket	[illegible]	84810	1:24.810	Jumlah Iterasi	3	
TioHaidarHanif: Saya ingin mengembangka	Continue Iteration	[illegible]	163595	2:43.595	Waktu Total	267129	4:27.129
	FAILED Tests\Feature\TicketPriorityTest > cre	[illegible]	18724	0:18.724	Rata rata	89043	1:29.043
Log Aktivitas	Saya ingin mengembangkan fitur "Log Aktivitas A	[illegible]	137385	2:17.385	Jumlah Iterasi	2	
TioHaidarHanif: Saya ingin mengembangka	masih ada error pada : FAILED Tests\Feature\U	[illegible]	99723	1:39.723	Waktu Total	237108	3:57.108
					Rata rata	118554	1:58.554
Update Batasan Lampiran	Saya ingin memperbaiki fitur upload file pada tick	[illegible]	67339	1:07.339			
Berikut hasil identifikasi dan rencana perb							

Claude 3.7 Sonnet							
Fitur	**Prompt**	**Log**	**Waktu (MS)**	**Readable Time**	**Keterangan**	**Hasil**	**Readable**
Manajemen Pengguna	Saya ingin meng	2025-05-23 02	156114	2:36.114	Jumlah iterasi	4	
	Continue Iterati	2025-05-23 03	292194	4:52.194	Waktu total	894289	14:54.289
	Continue Iterati	2025-05-23 03	278959	4:38.959	Rata-rata	223572	3:43.572
	Tolong buatkan	2025-05-23 03	167022	2:47.022			
Manajemen Kategori dan Subk	Saya ingin meng	2025-05-23 10	177474	2:57.474	Jumlah iterasi	4	
	Continue Iterati	2025-05-23 10	243611	4:03.611	Waktu total	667869	11:07.869
	test_get_all_cat	2025-05-23 12	165916	2:45.916	Rata-rata	166967	2:46.967
	Continue Iterati	2025-05-23 12	80868	1:20.868			
Manajemen Ticket	agentSaya ingin	2025-05-24 03	186713	3:06.713	Jumlah iterasi	4	
	Continue Iterati	2025-05-24 03	21540	0:21.540	Waktu total	747068	12:27.068
	Retry prompt	2025-05-24 03	437226	7:17.226	Rata-rata	186767	3:06.767
	Continue Iterati	2025-05-24 03	101589	1:41.589			
Manajemen Notifikasi	Tentu, berikut p	2025-05-26 04	131702	2:11.702	Jumlah iterasi	6	
	Continue Iterati	2025-05-26 04	209048	3:29.048	Waktu total	1217928	20:17.928
	Continue Iterati	2025-05-26 04	181924	3:01.924	Rata-rata	202988	3:22.988
	lanjut buatkan te	2025-05-26 05	254634	4:14.634			
	Continue Iterati	2025-05-26 05	155964	2:35.964			
	Continue Iterati	2025-05-26 06	284656	4:44.656			
Tambah Statistik Ticket pada	agentSaya ingin	2025-05-29 02	99341	1:39.341	Jumlah iterasi	3	
TioHaidarHanif: agentSaya ingin mer	Continue Iterati	2025-05-29 02	103977	1:43.977	Waktu total	283848	4:43.848
	pada endpoint u	2025-05-29 02	80530	1:20.530	Rata-rata	94616	1:34.616
Manajemen Chat	agentSaya ingin	2025-05-30 12	130899	2:10.899	Jumlah iterasi	5	
TioHaidarHanif: agentSaya ingin mer	Continue Iterati	2025-05-30 12	387881	6:27.881	Waktu total	1169964	19:29.964
	Continue Iterati	2025-05-30 12	368544	6:08.544	Rata-rata	233993	3:53.993
	Terdapat error d	2025-05-30 13	46464	0:46.464			
	Terdapat error d	2025-05-30 13	236176	3:56.176			
Manajemen FAQ	agentSaya ingin	2025-05-30 14	131182	2:11.182	Jumlah iterasi	3	
TioHaidarHanif: agentSaya ingin mer	Continue	2025-05-30 14	166336	2:46.336	Waktu total	594182	9:54.182
	Continue Iterati	2025-05-30 14	296664	4:56.664	Rata-rata	198061	3:18.061
Kirim email	agentSaya ingin	2025-06-08 14	87107	1:27.107	Jumlah iterasi	3	
TioHaidarHanif: agentSaya ingin mer	continue iteratio	2025-06-08 14	173652	2:53.652	Waktu total	331225	5:31.225
	terdapat error T	2025-06-08 14	70466	1:10.466	Rata-rata	110408	1:50.408
Token Tiket	Saya ingin mengen	[illegible]	103123	1:43.123	Jumlah iterasi	3	
TioHaidarHanif: Saya ingin mengembangka	continue iteration	2025-06-18 01:	270370	4:30.370	Waktu total	494277	8:14.277
	hapus use Illuminat	[illegible]	120784	2:00.784	Rata-rata	164759	2:44.759
Tambah informasi chat	Saya ingin mengen	[illegible]	199715	3:19.715	Jumlah iterasi	4	
TioHaidarHanif: Saya ingin mengembangka	continue iteration	[illegible]	209196	3:29.196	Waktu total	658103	10:58.103
	masih terdapat err	[illegible]	117170	1:57.170	Rata-rata	164526	2:44.526
	SQLSTATE[HY00	[illegible]	132022	2:12.022			
Tambah Prioritas	Saya ingin mengen	[illegible]	228480	3:48.480	Jumlah iterasi	3	
TioHaidarHanif: Saya ingin mengembangka	continue iteration	[illegible]	315362	5:15.962	Waktu total	796883	13:16.883
	FAILED TestsIF	[illegible]	252441	4:12.441	Rata-rata	265628	4:25.628
Log Aktivitas	Saya ingin mengen	[illegible]	168681	2:48.681	Jumlah iterasi	2	
TioHaidarHanif: Saya ingin mengembangka	continue iteration	[illegible]	226996	3:46.996	Waktu total	395677	6:35.677
					Rata-rata	197839	3:17.839
Update Batsan Lampiran	Saya ingin memper	[illegible]	149475	2:29.475			
TioHaidarHanif: Saya ingin memperbaiki fit	.						

https://tiny.cc/WorkspaceTATioHaidar

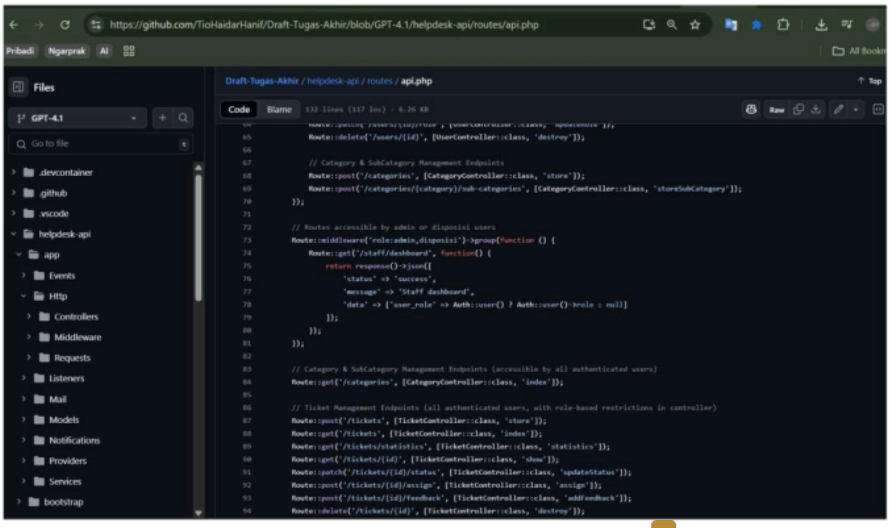
Lampiran 02. Sumber Kode

Project Awal



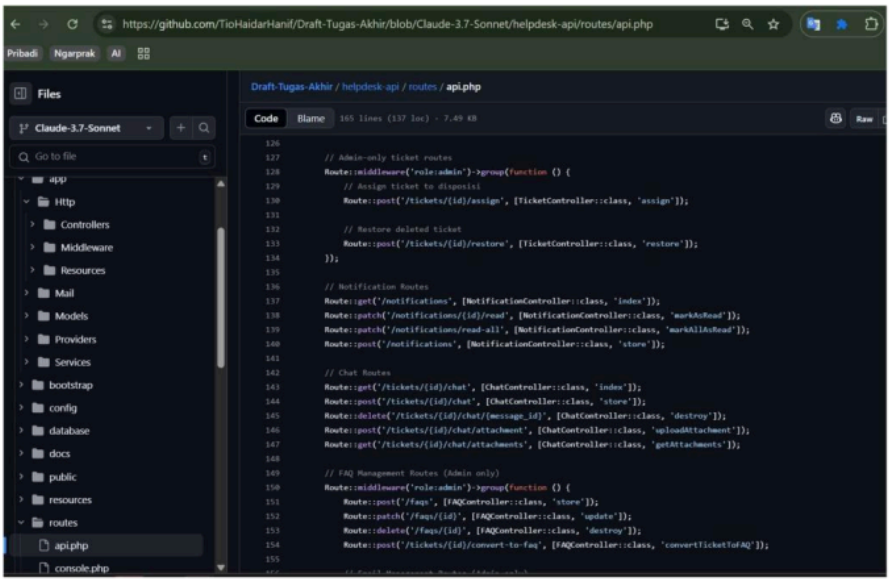
https://github.com/TioHaidarHanif/Draft-Tugas-Akhir/tree/project-awal-siap-uji

Project GPT 4.1



https://github.com/TioHaidarHanif/Draft-Tugas-Akhir/tree/GPT-4.1

Project Claude 3.7 Sonnet



https://github.com/TioHaidarHanif/Draft-Tugas-Akhir/tree/Claude-3.7-Sonnet

2b Buku TA

Perbandingan Efisiensi Model AI (GPT dan Claude) pada GitHub Copilot Agent Mode untuk melakukan Code Generation pada Pengembangan Software Backend API

Tugas Akhir

diajukan untuk memenuhi salah satu syarat memperoleh gelar sarjana

dari Program Studi S1 Rekayasa Perangkat Lunak

Fakultas Informatika

Universitas Telkom

1302210057

Tio Haidar Hanif

Program Studi Sarjana S1 Rekayasa Perangkat Lunak

Fakultas Informatika

Universitas Telkom

Bandung

2025

LEMBAR PENGESAHAN

Perbandingan Efisiensi Model AI (GPT dan Claude) pada GitHub Copilot Agent Mode untuk melakukan Code Generation pada Pengembangan Software Backend API

Comparison of Efficiency of AI Models (GPT and Claude) on GitHub Copilot Agent Mode for Backend API Software Development

1302210057

Tio Haidar Hanif

Tugas akhir ini telah diterima dan disahkan untuk memenuhu sebagai syarat memperoleh gelar pada Program Studi Sarjana <Nama Prodi>

Fakultas Informatika

Universitas Telkom

Bandung, 15 Agustus 2025

Menyetujui

Pembimbing I,	Pembimbing II,
Dana Sulistiyo Kusumo, S.T., M.T., PhD	Nungki Selviandro, S.Kom., M.Kom., Ph.D.
NIP: 02780011	NIP: 14880076

Ketua Program Studi
Sarjana S1 Rekayasa Perangkat Lunak,

15/08/2025

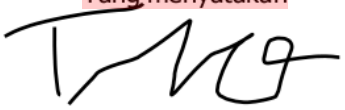
Dr. Eng. Jati Hiliamsyah Husen, S.T., M.Eng.
NIP: 20920040

LEMBAR ORISINALITAS

Dengan ini saya, Tio Haidar Hanif menyatakan sesungguhnya bahwa Tugas Akhir saya dengan judul "Perbandingan Efisiensi Model AI (GPT dan Claude) pada GitHub Copilot Agent Mode untuk melakukan Code Generation pada Pengembangan Software Backend API" berserta dengan seluruh isinya merupakan hasil karya saya sendiri, dengan tidak melakukan penjiplakan yang tidak sesuai dengan etika keilmuan yang berlaku dengan masyarakat keilmuan, serta produk dari tugas akhir ini bukan merupakan hasil dari *Generative AI*. Saya siap menanggung risiko/sanksi yang diberikan jika di kemudian hari ditemukan pelanggaran terhadap etika keilmuan dalam Laporan Tugas Akhir, atau jika ada klaim dari pihak lain terhadap keaslian karya.

Bandung, 15 Agustus 2025

Yang menyatakan



Tio Haidar Hanif

NIM 1302210057

ABSTRAK

Penggunaan AI sudah sangat marak digunakan, dan salah satu yang terdampak adalah pada ranah pengembangan aplikasi. Github Copilot yang merupakan salah satu *tools* berbasis AI yang dapat mempermudah pengembangan aplikasi serta meningkatkan efisiensi dalam pengembangan aplikasi. Namun, tingkat efisiensi pada Github Copilot juga bergantung dengan model AI yang digunakannya. Penelitian ini bertujuan untuk membandingkan secara kuantitatif efisiensi dari model AI GPT 4.1 dan Claude 3.7 Sonnet menggunakan Github Copilot dengan Mode Agent dengan studi kasus untuk melakukan pengembangan aplikasi *backend* REST API. Penelitian ini menggunakan pendekatan studi komparatif dengan melakukan eksperimen membuat beberapa fitur yang dikembangkan secara pararel pada masing-masing model. Pengukuran yang digunakan pada penelitian ini adalah menggunakan metode Goal Question Metric (GQM) dengan Metrik yang diukur adalah jumlah *prompt* dan durasi yang dibutuhkan oleh setiap *prompt* untuk menyelesaikan tiap tugas. Kesimpulan dari penelitian ini memperlihatkan adanya pengaruh antara kecepatan dengan model yang digunakan untuk *code generation*. GPT 4.1 menunjukkan keunggulan dari model ini yang dilihat dari durasi yang dibutuhkan pada model tersebut yang lebih cepat serta jumlah *prompt* uang dilakukan yang lebih sedikit. Akan tetapi, Claude 3.7 lebih efektif dalam menangani permintaan pembuatan kode yang lebih besar pada satu kali *prompt.* Dengan catatan, durasi *prompt* yang dilakukan cenderung lebih lama.

Kata Kunci:Generative AI, LLM4SE, Github Copilot, Rest API, Backend

ABSTRACT

The use of AI has become widespread, and one area that has been significantly impacted is application development. Github Copilot is one of the AI-based tools that can simplify application development and improve efficiency in the development process. However, the efficiency of Github Copilot also depends on the AI model it uses. This study aims to quantitatively compare the efficiency of the GPT 4.1 and Claude 3.7 Sonnet AI models using Github Copilot in Agent Mode, with a case study focused on developing a backend REST API application. The study employs a comparative approach by conducting experiments to develop several features in parallel across each model. The measurement used in this study is the Goal Question Metric (GQM) method, with the metrics measured being the number of prompts and the duration required by each prompt to complete each task. The conclusion of this study shows a correlation between speed and the model used for code generation. GPT 4.1 demonstrates the advantages of this model, as seen in the shorter duration required by the model and the fewer prompts needed. However, Claude 3.7 is more effective in handling larger code generation requests in a single prompt.

Keywords: Generative AI, LLM4SE, Github Copilot, Rest API, Backend

KATA PENGANTAR

Alhamdulillah, puji syukur atas kehadirat Allah yang telah memberikan segala nikmat karunia sehingga penulis bisa menyelesaikan Tugas Akhir dengan judul "Perbandingan Efisiensi Model AI (GPT dan Claude) pada GitHub Copilot Agent Mode untuk melakukan Code Generation pada Pengembangan Software Backend API". Penulisan TA ini merupakan syarat wajib untuk memperoleh gelar sarjana para program studi S1 Rekayasa Perangkat Lunak Universitas Telkom. Penulis mengambil tema ini karena penulis memiliki rasa keingintahuan yang tinggi terhadap teknologi AI. Untuk ke depannya penulis rasa AI akan menjadi bagian dari berbagai lini kehidupan, yang salah satunya menyangkut pada program studi ini. Proses penyusunan TA ini merupakan pembelajaran yang berharga untuk diri saya sendiri. Penulis ingin menyampaikan rasa Terima Kasih sebesar besarnya pada pihak yang berkontribusi baik secara langsung maupun tidak langsung pada proses penyusunan dokumen ini. Penulis juga menyadari bahwa Tugas Akhir ini masih sangat banyak kekurangan, baik dari segi penulisan atau konten dari Tugas Akhir ini. Untuk kritik yang membangun dari pembaca sangat kami harapkan agar perbaikan di masa mendatang. Semoga TA ini bisa memberikan manfaat dan bisa memberikan kontribusi positif dalam perkembangan ilmu pengetahuan.

UCAPAN TERIMA KASIH

Dengan segenap kerendahan hati, penulis haturkan puji dan syukur yang tak terhingga ke hadirat Allah Subhanahu Wa Ta'ala. Atas segala rahmat, karunia, dan nikmat-Nya yang tak terhingga—yang tak mungkin tertuliskan seluruhnya dalam dokumen ini—penulis akhirnya dapat menyelesaikan skripsi sebagai tahap akhir dari perjalanan studi ini.

Oleh karena itu, penulis ingin menyampaikan rasa terima kasih kepada:

1. Bapak Ode dan Mamah Ratna Astuti, kedua orang tua tercinta yang selalu menjadi sumber kekuatan. Terima kasih atas setiap doa, dukungan moril dan materiel
2. Bapak Dana Sulistiyo, S.T., M.T., PhD dan Bapak Nungki, S.Kom., M.Kom.,Ph.D selaku dosen pembimbing. Terima kasih atas kesabaran, waktu, ilmu, dan arahan yang sangat berharga dalam membimbing penulis dari awal hingga akhir pengerjaan skripsi ini.
3. Rekan-rekan satu tim TA Capstone (Sul, Nabiel, Ghaza, Burhan, Rizki). Terima kasih atas kerja sama, solidaritas, dan perjuangan bersama yang telah kita lalui sejak penyusunan proposal hingga saat ini.
4. Keluarga besar LDK Al-Fath, DKM Syamsul Ulum, dan Badan Mentoring. Terima kasih telah menjadi lingkungan yang positif, penuh inspirasi, dan selalu mengingatkan penulis pada tujuan yang lebih besar.
5. Seluruh pihak lain yang tidak dapat penulis sebutkan satu per satu, terima kasih atas segala bentuk bantuan dan dukungan yang telah diberikan.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1 PENDAHULUAN

1.1. Latar Belakang

Dalam pengembangan perangkat lunak, efisiensi merupakan aspek krusial yang mencakup waktu pengerjaan dan biaya pengerjaan. Di era teknologi dengan kompleksitas yang sangat tinggi, industri perangkat lunak memiliki tekanan untuk meningkatkan produktivitas serta mengurangi biaya yang berkaitan dengan pembuatan dan pemeliharaan perangkat lunak [1]. Akan tetapi, sering kali aspek ini kurang diperhatikan pada beberapa proyek. Berdasarkan laporan pada tahun 2025 dari gitnux.org[1], sekitar 55% proyek pengembangan perangkat lunak mengalami keterlambatan dan kelebihan biaya. Selain itu berdasarkan penelitian yang dilakukan oleh Standish Group's CHAOS Study[2], hanya 31% *project* IT yang benar-benar memenuhi target (termasuk ketepatan waktu dan anggaran). Oleh karena itu, efisiensi menjadi hal yang penting dalam pengembangan proyek aplikasi.

Pada era AI (*Artificial Intelligent*) seperti saat ini, solusi dari permasalahan dapat direalisasikan dengan lebih cepat dan lebih murah dengan bantuan AI [2]. Inovasi AI yang berupa Large Language Model (LLM) dan *chatbot* sangat berpengaruh dan mengubah cara menyelesaikan masalah serta proses pengembangan perangkat lunak [3]. Kemampuan AI ini bukan hanya sekedar memahami dan menghasilkan bahasa alami, tetapi juga dapat meningkatkan produktivitas dalam melakukan pengkodean program [3].

---

[1] Jannik Linder , *Software Development Industry Statistics* (https://gitnux.org/software-development-industry-statistics/, diakses pada 29 Juni 2025)
[2] Dillon Courts, *Software Project Failures: Why 70% Miss the Mark* (https://www.callibrity.com/articles/why-software-projects-miss-the-mark, diakses pada 22 Mei 2025, 2025)

Ada banyak *tools* berbasis AI yang dapat melakukan *code generation[4].* Salah satu dari *tools* tersebut yang cukup populer adalah Github Copilot[3]. Github Copilot merupakan *tools* untuk *pair programing* berbasis AI yang dapat terintegrasi langsung dengan *code editor [5]*. Keunggulan Github Copilot dalam membantu berbagai pekerjaan *programmer* telah dibuktikan melalui berbagai penelitian, baik yang dilakukan oleh Github langsung[4] maupun peneliti lainnya[6].

Berdasarkan penelitian yang dilakukan oleh Pandey et al. [6], menggunakan Github Copilot dapat menghemat waktu hingga 50% dalam dokumentasi kode, serta efisiensi 30% sampai 40% untuk tugas pengkodean berulang, pembuatan *unit test*, *debuging* dan *pair programming*. Serta mengurangi waktu pengkodean sebanyak 35% dari *baseline [C]*. Salah satu fitur baru dari Github Copilot adalah Agent Mode, yang menawarkan kemudahan dalam pengembangan aplikasi karena dengan fitur ini AI dapat melakukan operasi *file* pada proyek program, menjalankan perintah di terminal, merespons kesalahan dan masih banyak lagi[5]. Pada Github Copilot, pengguna diberikan kebebasan untuk memilih model AI yang digunakan[6].

Salah satu faktor yang menentukan kualitas dari hasil keluaran AI adalah model yang digunakannya [7]. Sehingga dapat diasumsikan tingkat efisiensi dari prosesnya juga tergantung pada model yang digunakan. Perbedaan tingkat efisiensi yang kecil pada proses pengembangan perangkat lunak dapat berpengaruh jika digunakan pada skala proyek besar. Sebagai contoh jika

---

[3] Shnu Vasudevan, *Github Copilot Adoption Trends: Insights from Real Data* (https://www.opsera.io/blog/github-copilot-adoption-trends-insights-from-real-data, diakses pada 6 Juni 2025)

[4] Ya Gao C GitHub Customer Research, *Research: Quantifying GitHub Copilot's impact in the enterprise with Accenture* (https://github.blog/news-insights/research/research-quantifying-github-copilots-impact-in-the-enterprise-with-accenture/, diakses pada 6 Juni 2025)

[5] Isidor Nikolic, *Introducing GitHub Copilot agent mode (preview).* (https://code.visualstudio.com/blogs/2025/02/24/introducing-copilot-agent-mode, diakses oada 8 Juni 2025)

[6] Isidor Nikolic, *Introducing GitHub Copilot agent mode (preview).* (https://code.visualstudio.com/blogs/2025/02/24/introducing-copilot-agent-mode, diakses oada 8 Juni 2025)

pada model A, proses pembuatan satu fitur adalah 5 menit, dan model B bisa sampai 6 menit, maka jika dihitung kasar pada proyek perangkat lunak dengan 100 fitur, model A dapat menghemat waktu sebanyak 100 menit. Bahkan, pada pengembangan perangkat lunak besar dapat sampai 1195 fitur yang dibuat [8]. Oleh karena itu perbandingan model AI diperlukan untuk memperkirakan tingkat efisiensi selama proses pengembangan perangkat lunak yang memanfaatkan AI.

Namun, dari penelitian yang sudah ditemukan, belum banyak membahas tentang perbandingan efisiensi pada proses penggunaan AI pada pengembangan perangkat lunak pada proyek nyata menggunakan Github Copilot Agent Mode. Banyak penelitian sebelumnya yang menggunakan dataset yang ada untuk melakukan pengujiannya, bukan berdasarkan kasus proyek nyata [3] [9] [10] [11]. Kemudian penelitian dengan kasus pengujian berdasarkan kasus nyata hanya membandingkan tingkat akurasinya[12] ataupun tingkat *correctness [13]*. Akurasi dan *correctness* dari model AI dalam *code generation* adalah hal yang penting, akan tetapi efisiensi dari proses juga penting.

Oleh karena itu, pada penelitian kali ini, penulis mengusulkan melakukan perbandingan antar model AI menggunakan Github Copilot Agent Mode untuk menguji efisiensi masing-masing model dalam melakukan pengembangan sistem aplikasi. Penelitian ini akan melakukan studi komparatif untuk membandingkan antara kedua model yaitu GPT dan Claude pada Github Agent Mode. Dengan adanya penelitian ini, diharapkan dapat menjadi wawasan kepada tim *developer* untuk memilih model AI yang paling sesuai dengan kebutuhan dan alur kerja.

1.2. Rumusan Masalah

Berdasarkan latar belakang yang sudah dituliskan sebelumnya, rumusan masalah dari penelitian ini adalah sebagai berikut

- Bagaimana perbandingan jumlah *prompt* serta durasi setiap prompt yang dibutuhkan pada setiap model AI (GPT dan Claude) untuk menyelesaikan serangkaian tugas yang sama dalam pengembangan perangkat lunak sistem *backend* REST API menggunakan Github Copilot Agent?
- Berdasarkan analisis data yang dilakukan serta pengalaman selama pengembangan, model AI mana yang memiliki tingkat efisiensi terbaik untuk melakukan tugas pengembangan perangkat lunak sistem *backend* REST API menggunakan Github Copilot Agent?

1.3. Tujuan dan Manfaat

Berdasarkan latar belakang yang ada, tujuan penelitian ini dapat dibagi dua yaitu

- Mengukur serta membandingkan tingkat efisiensi dari dua model AI untuk menghasilkan kode dalam pengembangan aplikasi menggunakan Github Copilot Agent Mode
- Memberikan data empiris sebagai dasar evaluasi terkait penggunaan Generative AI dalam mengembangkan aplikasi pada proyek nyata.

Adapun manfaat dari penelitian ini di antaranya adalah

- Untuk membantu pengguna atau *developer* dalam memilih model AI yang paling sesuai untuk membangun aplikasi dengan menggunakan Github Copilot Agent Mode
- Bagi peneliti, membantu memberikan referensi terkait efisiensi penggunaan AI pada lingkup *software development* terutama pada *code generation*

1.4. Batasan Masalah

Batasan masalah yang ada pada penelitian ini ada beberapa di antaranya adalah:

- Studi kasus pada penelitian ini dilaksanakan secara berkelompok sebagai bagian dari TA *Capstone,* dengan peneliti berperan sebagai *Backend Developer.* Fokus pengembangan aplikasinya dibuat hanya dari sisi *backend* dengan *output* berupa kode program menggunakan laravel 12 dan *API Documentation* yang di dihasilkan oleh AI. Sedangkan untuk rancangan kebutuhan aplikasinya ditentukan oleh *system analist* dan mengacu juga pada MVP yang dibuat *hustler* sampai tanggal 23 Juni 2025.
- Perbandingan yang dilakukan hanya dari sisi yang dirasakan oleh *developer* saja, tanpa melibatkan hal yang berkaitan langsung dengan *machine learning* seperti *fine tuning* dan lain sebagainya
- Akun yang digunakan pada Github Copilot menggunakan versi *student*. Sehingga ada kemungkinan perbedaan durasi setiap *prompt* antara Versi Pro dengan versi *student*.
- Kecepatan dari durasi *prompt* dapat tergantung dengan koneksi internet yang digunakan oleh peneliti.
- Penelitian dilaksanakan pada rentang waktu Mei 2025 sampai Juni 2025, sehingga mungkin di luar waktu tersebut ada perubahan dari Github Copilot yang dapat mengakibatkan hasil yang diberikan tidak sama persis dengan penelitian saat ini karena mungkin akan perubahan seperti peningkatan kualitas, baik dari model maupun dari *tools* itu sendiri.
- Model yang diujikan hanya 2 saja yaitu GPT 4.1 dan Claude 3.7 Sonnet yang ada pada Github Copilot.

1.5. Metode Penelitian

Penelitian ini menggunakan pendekatan studi komparatif dengan metode eksperimen untuk melakukan perbandingan efisiensi antara dua model AI yaitu GPT 4.1 dan Claude 3.7. pada *platform* Github Copilot Agent Mode. Pendekatan eksperimen ini digunakan untuk melakukan simulasi

pengembangan aplikasi pada dunia nyata, sehingga menggunakan *benchmark* standar yang ada.

Efisiensi diukur menggunakan metrik yang diadopsi dari *framework* yang bernama *Goal Question Metric* dengan metrik utamanya adalah jumlah *prompt* dan durasi setiap *prompt* yang dibutuhkan untuk menyelesaikan serangkaian tugas. Data kuantitatif akan dianalisis menggunakan statistika deskriptif sederhana untuk membandingkan kedua performa dari model tersebut.

BAB 2 TINJAUAN PUSTAKA

Bab ini menjelaskan tentang landasan teori dari penelitian ini serta penelitian terdahulu.

2.1. Landasan Teori

2.1.1. AI dan LLM

AI (Artificial Intelligent) merupakan teknologi meniru kecerdasan yang selayaknya manusia berfikir, mengambil keputusan, bahkan melakukan tindakan[7]. AI bekerja berdasarkan data yang dianalisis dan dijadikan sebuah pola untuk pengambilan keputusan. LLM (Large Language Model) merupakan salah satu penerapan dari Artificial Intelegece (AI) –Lebih tepatnya *Generative AI*[8]– yang dapat meniru dan memahami bahasa manusia dengan memanfaatkan *machine learning [14]*. LLM bekerja berdasarkan konsep *neuron network* dengan cara mempelajari pola dan probabilitas statistik dari sejumlah kata yang dilakukan anilisis yang memungkinkan LLM untuk memprediksi kata berikutnya, menjawab pertanyaan serta menghasilkan teks layaknya manusia[9].

LLM memahami dan melakukan tugas serta memberikan keluaran dengan menggunakan bahasa manusia sebagai perintah yang nantinya perintah tersebut akan disebut dengan *prompt [15].* LLM dapat melakukan pekerjaan manusia seperti *text generation, language translation* bahkan dapat melakukan pekerjaan yang lebih spesifik pada *software engineering* seperti *Code Generation, code summarization* dan masih banyak lagi [1]. Berdasarkan salah satu *Systematic Literatur Review* yang dipublikasikan pada

---

[7] *Artificial Intelligence: Pengertian, Cara Kerja dan Manfaatnya* (https://binus.ac.id/bandung/2024/02/artificial-intelligence-pengertian-cara-kerja-dan-manfaatnya/, diakses pada 06 Juni 2025)

[8] Lucas Mearian, *What are LLMs, and how are they used in generative AI?* (https://www.computerworld.com/article/1627101/what-are-large-language-models-and-how-are-they-used-in-generative-ai.html, diakses pada 6 Juni 2025)

[9] *Model Bahasa Besar* (https://www.solix.com/ms/kb/large-language-models/, diakses pada 06 Juni 2025)

Desember 2024 [16], terdapat lebih dari 300 penelitian yang menjelaskan tentang penerapan LLM pada *Software Engineering*. Mulai dari *Requirement Engineering, Software Design, Software Development, Software Management* sampai *Software Maintance*.

2.1.2. *Code Generation*

Code Generation merupakan proses untuk membuat kode secara otomatis berdasarkan spesifikasi atau kebutuhan yang ditentukan oleh *developer* [17]. *Code Generation* merupakan salah satu bagian dari penerapan LLM pada *Software Engineering* di bidang *Software Development* [16]. Bahkan dari temuan tersebut, lebih dari 115 penelitian yang membahas tentang *Code Generation.* Dengan *code generation, developer* dapat mengurangi *effort* dalam melakukan pengkodean serta memungkinkan untuk fokus pada tugas dan penyelesaian masalah yang lebih kompleks [17]. Hal ini menunjukan betapa banyak minat penelitian terkait bidang tersebut. *Code Generation* dengan menggunakan LLM dilakukan dengan cara memberikan perintah (*prompt*) dengan bahasa manusia ke dalam *tools* untuk *code generation* seperti Github Copilot.

2.1.3. Github Copilot

Github Copilot merupakan salah satu *coding assistant* yang dibuat oleh OpenAI dan Microsoft yang saat ini banyak diteliti terkait efektifitas dan keandalannya dalam melakukan *code generation*[6]. Github Copilot menggunakan teknik *machine learning* dan *codebase* yang luas sehingga memungkinkan *developer* untuk meningkatkan produktivitas serta mempercepat proses pengembangan perangkat lunak[18].

Github Copilot Agent Mode merupakan salah satu fitur dari Github Copilot yang dapat memberikan pengalaman yang berbeda dalam melakukan

pengkodean karena dapat memahami permintaan yang lebih kompleks[10]. Mode Agent menawarkan kemudahan dalam pengembangan aplikasi karena dengan fitur ini AI dapat melakukan operasi *file* pada proyek program, menjalankan perintah di terminal, merespons kesalahan dan masih banyak lagi[11].

2.1.4. Model: Claude G GPT

Salah satu faktor yang menentukan kualitas dari hasil AI merupakan model [7]. Model merupakan program yang sebelumnya sudah dilatih oleh sekumpulan data untuk mengenali pola tertentu atau membuat keputusan berdasarkan data tanpa intervensi dari manusia[12]. Ada berbagai macam model AI, model ini dibagi menjadi 3 jenis berdasarkan arsitekturnya seperti *Encoder Only*, *Encoder-Decoder*, dan *Decoder Only[1C]*. Akan tetapi, model yang umum digunakan untuk *code generation* adalah *Decoder Only[1C]* . Ada banyak macam model yang termasuk kategori *Decoder Only*, di antaranya ada Claude dari Antropic, Gemini dari Google, GPT dari OpenAI dan masih banyak lagi[16].

Dari sekian banyak model yang tersedia, penulis akan berfokus pada dua model yaitu Claude dan GPT. Kedua model tersebut dipilih karena memiliki keunggulan di antara model yang lain seperti pada penelitian tentang LLM4DS yang membandingkan beberapa model lain [19]. Pada penelitian berbeda – persisnya tentang perbandingan model AI untuk pembuatan kode untuk HRI– Claude memiliki performansi paling tinggi yaitu ada di angka 95% untuk ketepatan dalam pembuatan kode [12]. Kemudian pada penelitian [19], ChatGPT merupakan model yang memiliki keunggulan ketika menangani masalah yang sulit. Selain itu, ChatGPT-4 secara spesifik memiliki keunggulan

---

[10] Brylie Christopher Oxley, *GitHub Copilot Agent Mode: Enhancing Developer Workffows* (https://dev.to/brylie/github-copilot-agent-mode-enhancing-developer-workflows-2ae0, diakses pada 8 Juni 2925)

[11] Isidor Nikolic, *Introducing GitHub Copilot agent mode (preview).* (https://code.visualstudio.com/blogs/2025/02/24/introducing-copilot-agent-mode, diakses pada 8 Juni 2025)

[12] IBM, *What is an AI model?* (https://www.ibm.com/think/topics/ai-model, . 4 Mei 2025)

dibanding Gemini dalam mendeteksi dan memperbaiki *bug* pada kode program [1].

Pada penelitian kali ini peneliti berfokus pada dua model yaitu GPT dan Claude. Versi yang akan digunakan adalah GPT 4.1 dan Claude 3,7 Sonnet. Versi tersebut dipilih karena versi tersebut merupakan versi tertinggi (sebelum tanggal 22 Mei 2025) yang ada di Github Copilot ketika penelitian ini dibuat. GPT 4.1 rilis pada tanggal 14 April 2025[13] sedangkan Claude 3.7 Sonnet rilis pada tanggal 25 Februari 2025[14]. Sehingga dari tanggal rilis yang masih dalam tahun yang sama masih dianggap relevan untuk dilakukan pengujian.

Terdapat beberapa perbedaan kedua model yang digunakan[15] yang relevan pada penelitian ini terkait. GPT 4.1 memberikan fleksibilitas yang lebih besar karena memiliki dukungan 50 bahasa dan *context windows* yang banyak yaitu sebanyak satu juta *token*. Sementara Claude 3.7, sangat baik saat menggunakan bahasa inggris, serta menyediakan *context windows* sebanyak 200 ribu token. *Knowledge cutoff* pada GPT 4.1 ada pada Juni 2024, sedangkan Claude ada pada Maret 2024.

2.1.5. Rest API

Rest API yang merupakan singkatan dari *Represetational State Transsfer API* merupakan gaya arsitektur perangkat lunak yang dikemukakan oleh Roy Thomas Fleding pada tahun 2000 [20]. Rest API merupakan gaya arsitektur yang melibatkan jaringan untuk komunikasinya. Rest memisahkan antara *server* dan *client* sebagai penerapan prinsip *sepearation of concen*.

---

[13] Tempo, *OpenAI Buka Akses GPT-4.1 untuk Pelanggan Plus, Pro, dan Team* (https://www.tempo.co/sains/openai-buka-akses-gpt-4-1-untuk-pelanggan-plus-pro-dan-team--1443992, diakses pada 6 Jun 2025

[14] Tempo, *Anthropic Luncurkan Model AI Claude 3.7 Sonnet, Ungguli Model OpenAI dan DeepSeek?* (https://www.tempo.co/digital/anthropic-luncurkan-model-ai-claude-3-7-sonnet-ungguli-model-openai-dan-deepseek--1212031, diakses pada 9 Juni 2025)

[15]OpenAI GPT 4.1 vs Claude 3.7 vs Gemini 2.5: Which Is Best AI?(https://yourgpt.ai/blog/updates/openai-gpt-4-1-vs-claude-3-7-vs-gemini-2-5, diakses pada 21 Juli 2024)

Penggunaan Rest API digunakan untuk melakukan komunikasi antara *frontend* dengan *backend* pada proyek penelitian ini.

2.1.6. Efisiensi

Berdasarkan Kamus besar bahasa Indonesia, Efisiensi berarti 'Ketepatan waktu cara (usaha, kerja) dalam menjalankan sesuatu (dengan tidak membuang waktu, tenaga biaya)[16]. Dalam pengembangan aplikasi, efisiensi dapat diukur dengan menghitung waktu yang dibutuhkan untuk menyelesaikan suatu proses, serta jumlah sumber daya yang digunakan pada setiap proses [21]. Pada penelitian ini, efisien dilihat dari berapa banyak *prompt* yang digunakan, dan waktu yang dibutuhkan dalam melakukan *prompting*.

2.1.7. Goal Question Matrix (GQM)

Goal Question Matrix merupakan salah satu paradigma yang diteliti kembali oleh Basili dan Rombach [21] untuk melakukan pengukuran *software* dan prosesnya. Seperti namanya, paradigma ini terdiri dari tiga bagian yaitu Goals, Question dan Matrix.

Goals berarti tujuan yang ingin dicapai, Question berarti pertanyaan yang ingin dijawab dari tujuan tersebut, dan Metric berarti aspek pengukuran yang perlu dikumpulkan untuk menjawab pertanyaan yang diajukan[17]. Referensi penelitian yang menggunakan metode ini adalah penelitian untuk melakukan *code generation* untuk masalah *data science* [19].

2.2. Penelitian Terdahulu

Dari berbagai model AI yang tersedia pada Github Copilot, penelitian ini akan berfokus dalam menggunakan model GPT dari Open AI dan Claude sebagai objek penelitian. Pemilihan kedua model ini dipilih karena keunggulannya dibanding dengan model lain pada penelitian sebelumnya.

---

[16] KBBI, (https://kbbi.kemdikbud.go.id/entri/efisiensi, diakses pada 8 Juni 2025)
[17] Goal-Question-Metrics (GQMs), (https://www.productplan.com/glossary/goal-question-metrics/, diakses pada 8 Juni 2025)

Sebagai contoh studi komparatif yang dilakukan oleh Nascimento et al. [19] yang melakukan pengujian pada konteks pembuatan kode untuk keperluan *data science*. Dari berbagai model yang diuji seperti Microsoft Copilot (GPT-4 Turbo), ChatGPT (o1-preview), Claude (3.5 Sonnet), and Perplexity Labs (Llama-3.1-70b-instruct), ditemukan bahwa hanya GPT dan Claude yang memiliki tingkat keberhasilan diatas 60%. Pada penelitian tersebut juga ChatGPT menunjukkan konsistensi dari performanya pada setiap tingkatan tugas. Kemudian penelitian lain yang membahas studi komparatif pada pembuatan kode untuk Human-Robot Interaction (HRI), Claude berhasil unggul dan menunjukkan performa akurasi tertinggi hingga 95% [12].

Kebanyakan penelitian terkait perbandingan model AI menggunakan *dataset* seperti humaneval, leetcode, pass@k dan lain sebagainya[16] [3] [9] [10] [11]. Salah satu kekurangan menggunakan *dataset* adalah ada kemungkinan hasil perbandingannya tidak mewakili kompleksitas yang ada pada dunia nyata[18]. Oleh karena itu perlu juga dilakukan evaluasi berdasaran pengalaman saat proses pengembangan *software* yang harapannya dapat digunakan untuk keputusan praktis di industri.

Kemudian, perbandingan model AI tanpa menggunakan *dataset* adalah penelitian yang dilakukan oleh Sobo et al. [12]. Penelitian ini melakukan studi komparatif terkait efektivitas antara model GPT, Claude, dan Gemini dalam pembuatan kode untuk HRI. Penelitian ini menggunakan 20 prompt yang di desain untuk *generate task* pada *robotic task* seperti *basic movement* dan lain sebagainya Proses ini melakukan *prompt* pada tiap LLM yang diuji, lalu mengintegrasikannya dengan lingkungan uji yang sudah dibuat oleh peneliti. Setiap *logic* dari hasil LLM tidak ada yang diubah untuk memastikan integritas penelitian. Akan tetapi kesalahan minor secara *syntax* seperti kehilangan titik koma masih ditoleransi. Penelitian tersebut relevan dengan penelitian yang dilakukan saat ini. Akan tetapi penelitian ini bukan untuk mengukur efisiensi

[18] *What is HumanEval?* (https://www.deepchecks.com?m/glossary/humaneval/, Diakses pada 12 Mei 2025)

dari masing-masing model AI serta studi kasus yang digunakan bukan untuk membuat aplikasi *backend*.

Untuk penelitian terhadap proyek nyata, Pandey et al. [6] melakukan evaluasi terhadap penggunaan Github Copilot pada proyek *Software Engineering* di dunia nyata. efisiensi dan tantangan dalam penggunaannya adalah aspek yang dievaluasi pada penelitian tersebut dalam menggunakan Github Copilot. Penelitian ini memberi kesimpulan bahwa Github Copilot secara signifikan meningkatkan efisiensi dari developer. Github Copilot baik dalam mengurangi waktu developer pada tugas yang repetitif.

Terkait perhitungan efisiensi, penelitian yang dilakukan Wang et al [22] mengukur waktu lama latensi dalam *generate code* untuk tiap fungsi. Penelitian ini menghasilkan angka 0.63 sampai 2.34 menit dalam melakukan proses pembuatan kode pada setiap fungsi. Lalu pada penelitian oleh Vasiliniuc et al. [23] menghitung efisiensi dari tugas yang diberikan kepada peserta penelitian dan menghitung waktu yang dibutuhkan untuk mengerjakan tugasnya. Penelitian ini membandingkan antara yang menggunakan AI dan yang tidak. Tetapi kedua penelitian tersebut belum mencoba untuk mengukur waktu efisiensi dari durasi setiap *prompt* dan jumlah *prompt* untuk menyelesaikan fitur atau tugas.

BAB 3 PERANCANGAN SISTEM

Bab ini akan memaparkan gambaran umum terkait perancangan penelitian yang akan dilakukan. Bab ini meliputi desain penelitian seperti alur penelitian dan metrik yang digunakan, serta desain perancangan *backend* yang akan dikembangkan.

3.1. Desain G Alur Penelitian

Fokus utama penelitian ini adalah untuk membandingkan efisiensi pada dua model AI yang tersedia pada Github Copilot Agent Mode. Secara spesifik model yang akan diujikan adalah GPT 4.1 dan Claude 3.7 Sonnet. Kedua model ini akan diujikan dalam konteks pengembangan sistem Backend Rest API. Perlu ditekankan penelitian ini **bukan** menguji masing-masing model pada masing-masing *website* seperti chatgpt.com dan claude.ai, tetapi menguji masing-masing model pada satu *platform* yang sama yaitu Github Copilot dengan mode Agent.

Studi komparatif merupakan jenis penelitian deskriptif yang bertujuan menjawab sebab akibat dari suatu fenomena dengan menganalisis faktor yang menjadi penyebab terjadinya atau munculnya suatu fenomena. Pendekatan studi komparatif dipilih karena relevan untuk membandingkan dua atau lebih kelompok terhadap variabel tertentu [24]. Metode Penelitian ini dilandaskan pada penelitian sebelumnya yang menggunakan komparasi model AI untuk melakukan untuk *Human Robot Interaction [12]*.

Penelitian ini menggunakan studi komparatif sehingga perlu menetapkan beberapa variabel[19]. Pada penelitian ini, variabel bebasnya adalah model yang digunakan. Lalu variabel tergantungnya adalah durasi setiap *prompt* dan jumlah *prompt* (sesuai dengan *Goal Question Matrix*). Variabel pengganggu dari penelitian ini, di antaranya adalah koneksi internet,

[19] Gramedia Blog, Studi Komparatif: Pengertian, Manfaat, Variabel, Macam dan Contoh (https://www.gramedia.com/literasi/studi-komparatif/?srsltid=AfmBOopUMK7wCRPHaPevDzWJZCw3039wf_qJPa5sgqaoDiDtWBej071J, diakses pada 21 Juli 2025)

keterhubungan dengan konteks sebelumnya dan potensi adanya *error* internal proyek atau *tools* yang digunakan. Untuk itu, variabel kontrol dari penelitian ini adalah penggunaan *prompt* awal yang sama persis, serta menggunakan *environment* yang sama (seperti versi VSCode yang sama, *device* yang sama, versi Github Copilot yang sama dan lain sebagainya). Variabel mediator pada penelitian ini adalah variasi fitur yang akan dibuat. Untuk menunjang hasil yang diinginkan, variabel mediasi dari penelitian ini adalah hasil kode yang dibuat, serta *log* awal dan *log* akhir pada setiap *prompt* yang dilakukan.

Pengumpulan data dilakukan melalui serangkaian eksperimen. Data yang dicatat meliputi jumlah *prompt* yang dibutuhkan untuk melakukan setiap tugas serta mengukur durasi pada setiap melakukan *prompting*. Pendekatan ini dilakukan untuk menguji performa model AI dalam skenario proyek pengembangan aplikasi di dunia nyata. Jumlah *prompt* dan durasi setiap *prompt* adalah kedua hal yang memang dirasakan langsung oleh *programmer* dalam menggunakan AI dalam *code generation*. Peneliti tidak menggunakan *dataset benchmark* yang sudah tersedia dan biasa digunakan untuk pengujian model AI seperti Leetcode dan HumanEval[20] karena dengan menggunakan *dataset* seperti itu tidak sepenuhnya merepresentasikan kompleksitas pengembangan aplikasi di dunia nyata [1]. Terlebih lagi penelitian ini akan berfokus pada pengembangan backend API, sehingga diperlukan metode yang dapat melakukan evaluasi terhadap spesifik domain tertentu[19]. Untuk memandu pengumpulan dan analisis data, peneliti menggunakan Goal Question Matrix [21] yang diadaptasi sebagai berikut:

[20] *What is HumanEval?* (https://www.deepchecks.com/glossary/humaneval/, Diakses pada 12 Mei 2025)

Untuk memandu pengumpulan dan analisis data, peneliti menggunakan Goal Question Matrix [21] yang diadaptasi sebagai berikut:

Tabel 1. Goal Question Metric

Goal	Question	Metrix
Membandingkan efisiensi model AI dalam proses *Code Generation* dari sudut pandang *programmer*	Berapa banyak *prompt* yang dibutuhkan untuk menghasilkan satu fitur ?	1. Jumlah *prompt* per fitur hingga fiturnya sesuai dengan kebutuhan 2. Jumlah *prompt* untuk menyelesaikan satu proyek
	Berapa lama durasi yang dibutuhkan AI untuk merespons *prompt* dan menyelesaikan satu fitur?	1. durasi setiap satu kali *prompt* 2. total durasi *prompt* yang dibutuhkan untuk menyelesaikan satu proyek

Teknik analisis data menggunakan statistika deskriptif. Angka yang dihasilkan dari tiap metrik akan dianalisis untuk mendapatkan nilai rata-rata, median, modus, standar deviasi, nilai minimum dan nilai maksimum. Statistika deskriptif dianggap cukup untuk membandingkan tingkat efisiensi dari kedua model AI yang diuji.

3.2. Desain Perancangan Sistem

Pelaksanaan percobaan melibatkan pengembangan sebuah aplikasi *backend* REST API dengan daftar fitur yang sebelumnya sudah ditentukan oleh peran *hustler* dan *system analyst* pada pengerjaan TA Capstone ini. Aplikasi ini akan dibuat menggunakan *framework* PHP yaitu Laravel versi 12. Laravel dipilih didasarkan pada popularitas penggunaan *framework* yang luas sampai saat ini[21]. Laravel dapat digunakan untuk membuat website salah satunya membuat Rest API. Pengembangan sistem ini dibantu sepenuhnya oleh Github Copilot Agent Mode.

3.2.1. Teknik *Prompting*

[21] Aljaz Mughal, *Laravel Usage Statistics 2024* (https://aundigital.ae/blog/laravel-usage-statistics/, Diakses pada 22 Mei 2025, 2024)

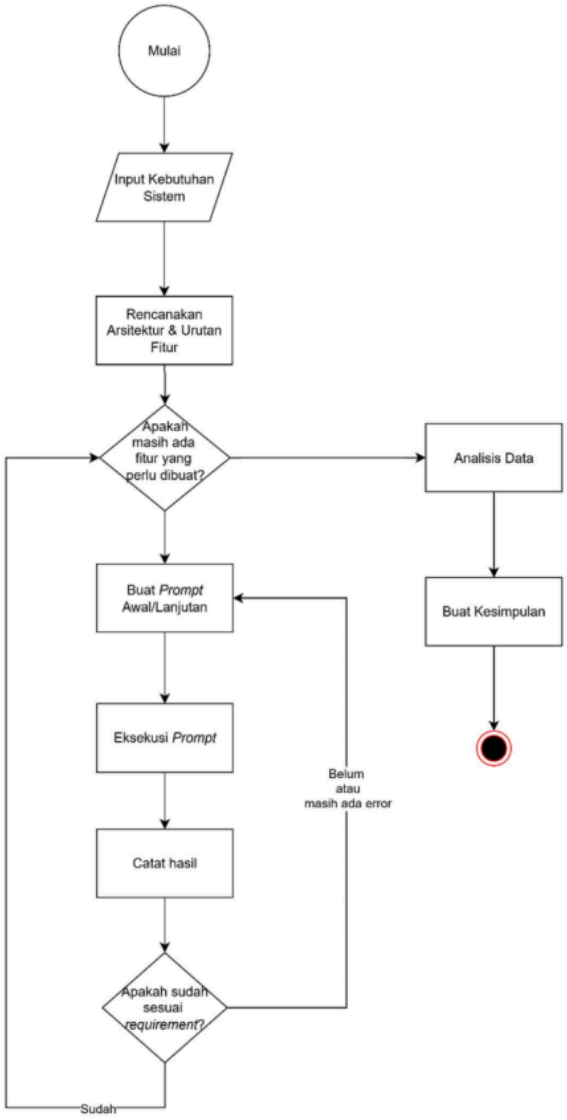
Prompt yang digunakan dibagi menjadi dua yaitu *prompt* utama untuk membuat fitur, dan *prompt* tambahan untuk *follow* up jika ada kesalahan atau ada kekurangan pada pelaksanaan tugas. Penyempurnaan *prompt* diperlukan untuk mendapatkan hasil yang lebih maksimal[25]. Strategi *prompt* utama pada penelitian ini mayoritas mengandalkan pembuatan *prompt* dengan bantuan dari AI juga. Penulis memberikan konteks permasalahan atau fitur yang ingin dibuat, kemudian AI akan membuat *prompt* versi lebih lengkap dan komprehensif untuk menghasilkan keluaran yang lebih efektif. Struktur *prompt* utama meliputi konteks proyek, spesifikasi fitur, langkah-langkah implementasi, batasan teknis dan pencatatan. Setiap model akan memiliki *prompt* utama yang sama, tetapi untuk *prompt* tambahan akan menyesuaikan dengan kondisi kode yang telah dibuat.

3.2.2. Alur Proses Pelaksanaan

Proses eksperimen akan dimulai dengan menentukan fitur yang akan dikembangkan terlebih dahulu. Setelah itu, dilakukan proses *prompting* untuk menghasilkan kode program serta *unit test* yang sesuai dengan fitur tersebut.

Jika proses hasil kodenya belum sesuai, maka dilakukan *prompting* kembali. Jika kebutuhan pada fitur tersebut sudah terpenuhi, maka di lanjut ke fitur berikutnya yang belum dibuat. Kebutuhan fitur dikatakan terpenuhi jika *unit/feature test* yang dibuat telah berhasil dijalankan. Pengujian *unit/feature* test ini juga untuk mendukung *correctness* dari pembuatan kode.

Pembuatan kode dibuat perfitur agar lebih mudah untuk di kelola. Setiap proses *prompting* yang dilakukan akan dicatat waktu proses tunggu hasilnya dan jumlah *prompt* yang dilakukan tiap tugas. Setelah semua berhasil, dilakukan analisis data dan pembuatan kesimpulan. Berikut gambaran *ffowchart*nya



Gambar 1. Alur Perancangan

Setiap tugas nantinya akan dihitung jumlah *prompt* yang dibutuhkan serta durasi pada tiap *prompt*. Setiap *prompt* dapat memiliki perintah yang bervariasi. Kebanyakan *prompt* dilakukan untuk melanjutkan proses *code generation* yang belum dan juga melakukan *prompting* untuk perbaikan kode yang belum sempurna.

Data waktu diambil dari *log* pemrosesan pada Github Copilot Chat dengan cara menghitung selisih antara waktu *log* pertama dan terakhir yang paling masuk akal pada setiap *prompt.*

Penting untuk diperhatikan bahwa pengukuran waktu ini tidak sepenuhnya merepresentasikan waktu pemrosesan AI secara keseluruhan. Hal ini disebabkan karena pada Github Copilot Agent mode, terdapat proses yang memerlukan konfirmasi dari pengguna untuk eksekusi suatu perintah seperti pembuatan *folder* atau pengeksekusian kode pada *command line*. Sehingga waktu yang digunakan pada *command line* akan terhitung juga pada total durasi pemrosesan *code generation*. Meskipun demikian, peneliti berupaya melakukan perhitungan seobjektif mungkin berdasarkan kemunculan *log* pertama dan *log* terakhir yang paling masuk akal.

3.2.3. *Fairness* dalam eksperimen

Proses penelitian ini dibuat seadil mungkin dengan beberapa kontrol yang ditetapkan.

- Perangkat lunak dan perangkat keras yang digunakan pada tiap percobaan yang sama akan menggunakan perangkat yang sama (seperti komputer, internet) dan versi perangkat lunak yang sama (seperti versi VSCode, Github Copilot, Codespace) Keadilan dalam penelitian ini didasarkan oleh penelitian yang dilakukan oleh Sobo dan lainnya [12]
- Untuk menghindari terbawanya konteks dari interaksi sebelumnya, setiap tugas baru (seperti membuat fitur baru) akan memulai *session chat* yang baru (*new chat*) pada masing-masing model AI.
- *Prompt* utama (*prompt* awal tiap tugas) akan dibuat sama persis untuk kedua model yang akan diuji.
- Sebagai upaya untuk meminimalisir bias urutan (*sequence bias*), maka diterapkan strategi proses *prompting* dengan urutan yang bergantian. Kumpulan fitur awal (manajemen pengguna sampai fitur kirim email),

pengembangan di mulai dari Claude 3.7 kemudian baru menggunakan GPT 4.1. Selain fitur yang telah disebutkan sebelumnya, proses *code generation* dimulai dari GPT 4.1 terlebih dahulu. Pendekatan ini diterapkan untuk meminimalisir salah satu model mendapat keuntungan dari konteks dan solusi yang mungkin dihasilkan oleh model sebelumnya.

BAB 4 HASIL PERCOBAAN DAN ANALISIS

Bab ini memaparkan hasil percobaan dan analisis dari eksperimen yang dikerjakan. Bab ini akan menjelaskan tentang proses pengerjaan dan hasil yang di dapat selama penelitian.

4.1. Skenario Percobaan

Eksperimen dilakukan dengan tahapan yang sudah di paparkan pada bab 3. Adapun contoh dari *prompt* utama pada setiap fitur sebagai berikut:

Tabel 2. *Sample* **Rencana** *Prompt*

Fitur	Prompt
Manajemen Pengguna	Saya ingin mengembangkan fitur "Manajemen User" untuk sistem Helpdesk Ticketing. Berdasarkan #file:progress.md fitur manajemen user yang lengkap belum diimplementasikan. Fitur ini membutuhkan: Controller yang menangani operasi CRUD untuk User (UserController) Implementasi middleware role untuk memastikan hanya admin yang dapat mengakses fitur ini Endpoint API untuk mengelola user dan melihat statistik user Sesuai dengan #file:api.md endpoint yang perlu dibuat adalah: GET /users untuk mendapatkan semua user (Admin only) GET /users/{id} untuk mendapatkan detail user (Admin only) PATCH /users/{id} untuk memperbarui informasi user (Admin only) PATCH /users/{id}/role untuk memperbarui role user (Admin only) DELETE /users/{id} untuk menghapus user (Admin only) GET /users/statistics untuk mendapatkan statistik user (Admin only) Mohon bantu saya untuk: Membuat implementasi backend untuk fitur manajemen user dengan memastikan hanya admin yang bisa mengakses

	Membuat unit test dan feature test untuk memastikan fitur berfungsi Mendokumentasikan perubahan di #file:progress.md kententuan ada pada #file:copilot-instruction.md lifecycle laravel 12 #fetch https://laravel.com/docs/12.x/lifecycle Lakukan langkah demi langkah
Tambah Statistik Ticket pada user	agent Saya ingin mengedit fitur manajemen user agar menambahkan statistik dan daftar ticket dari setiap user ketika get all user dan get user by id. Berdasarkan progress.md, fitur manajemen user sudah diimplementasikan tetapi belum memiliki statistik detail dan daftar ticket per user. Fitur yang perlu ditambahkan: - Statistik jumlah ticket yang telah dibuat oleh user - Daftar ticket (id, judul, status) yang dibuat oleh user tersebut - Informasi ini harus ditampilkan pada endpoint GET /users dan GET /users/{id} Sesuai dengan api.md, endpoint yang perlu dimodifikasi adalah: - GET /users untuk mendapatkan semua user dengan tambahan statistik per user - GET /users/{id} untuk mendapatkan detail user dengan tambahan statistik dan daftar ticket dari user tersebut Mohon bantu saya untuk: 1. Memodifikasi UserController untuk menambahkan statistik ticket dan daftar ticket pada response 2. Menambahkan relasi antara User model dan Ticket model jika belum ada 3. Mengoptimalkan query database untuk menghindari N+1 problem saat mengambil data ticket 4. Membuat unit test dan feature test untuk memastikan fitur berfungsi dengan benar 5. Mendokumentasikan perubahan di progress.md Lakukan langkah demi langkah dengan mempertimbangkan lifecycle Laravel 12 untuk memastikan implementasi yang efisien dan sesuai dengan best practice.

Manajemen Chat	agent Saya ingin mengembangkan fitur ""Manajemen Chat"" untuk sistem Helpdesk Ticketing. Berdasarkan progress.md, fitur chat belum diimplementasikan. Fitur ini membutuhkan: - Menggantikan sistem ""feedback"" yang saat ini ada pada setiap ticket - Model ChatMessage untuk menyimpan pesan chat - Model ChatAttachment untuk menangani lampiran pada chat - Controller untuk operasi CRUD pada pesan chat - Implementasi real-time chat (opsional, jika memungkinkan) Sesuai dengan api.md, endpoint yang perlu dibuat adalah: - GET /tickets/{id}/chat untuk mendapatkan semua pesan chat pada ticket tertentu - POST /tickets/{id}/chat untuk menambahkan pesan chat baru pada ticket - POST /tickets/{id}/chat/attachment untuk mengunggah lampiran pada chat - DELETE /tickets/{id}/chat/{message_id} untuk menghapus pesan chat (oleh penulis pesan saja) - GET /tickets/{id}/chat/attachments untuk mendapatkan semua lampiran chat Mohon bantu saya untuk: 1. Membuat migrasi database untuk tabel chat_messages dan chat_attachments 2. Membuat model ChatMessage dan ChatAttachment dengan relasi yang sesuai 3. Membuat controller untuk menangani operasi chat 4. Implementasi otorisasi agar pengguna hanya bisa mengakses chat ticket yang relevan dengan mereka 5. Integrasi dengan sistem notifikasi yang sudah ada agar pengguna mendapat notifikasi saat ada pesan chat baru 6. Membuat unit test dan feature test untuk memastikan fitur berfungsi dengan benar 7. Mendokumentasikan perubahan di progress.md Lakukan langkah demi langkah dengan mempertimbangkan lifecycle Laravel 12 untuk memastikan implementasi yang efisien dan sesuai dengan best practice.
Manajemen FAQ	agent

	Saya ingin mengembangkan fitur ""Manajemen FAQ"" untuk sistem Helpdesk Ticketing. Berdasarkan progress.md, fitur FAQ belum diimplementasikan. Fitur ini membutuhkan: - Model FAQ untuk menyimpan pertanyaan yang sering diajukan dan jawabannya - Controller untuk operasi CRUD pada FAQ - Kemampuan untuk mengkonversi ticket yang sudah ada menjadi FAQ - Implementasi otorisasi agar hanya admin yang dapat mengelola FAQ - Endpoint API untuk mengakses FAQ secara publik (tanpa autentikasi) Sesuai dengan best practice API, endpoint yang perlu dibuat adalah: - GET /faqs untuk mendapatkan semua FAQ (publik) - GET /faqs/{id} untuk mendapatkan detail FAQ (publik) - POST /faqs untuk membuat FAQ baru (admin only) - POST /tickets/{id}/convert-to-faq untuk mengkonversi ticket menjadi FAQ (admin only) - PATCH /faqs/{id} untuk mengupdate FAQ (admin only) - DELETE /faqs/{id} untuk menghapus FAQ (admin only) - GET /faqs/categories untuk mendapatkan kategori FAQ (publik) Mohon bantu saya untuk: 1. Membuat migrasi database untuk tabel faqs dengan kolom-kolom yang diperlukan 2. Membuat model FAQ dengan relasi ke Category dan User 3. Membuat controller FAQController untuk menangani operasi CRUD 4. Mengimplementasikan endpoint untuk konversi ticket menjadi FAQ 5. Membuat middleware otorisasi agar hanya admin yang bisa mengelola FAQ 6. Membuat unit test dan feature test untuk memastikan fitur berfungsi dengan benar 7. Mendokumentasikan perubahan di progress.md Lakukan langkah demi langkah dengan mempertimbangkan lifecycle Laravel 12 untuk memastikan implementasi yang efisien dan sesuai dengan best practice.

Sebagai catatan, fitur autentikasi dijadikan kondisi awal proyek sebelum pengujian. Sehingga pada saat eksperimen proses *prompting*, fitur yang sudah ada sejak awal adalah autentikasi. Artinya, fitur yang akan di uji adalah fitur manajemen pengguna, tiket, notifikasi dan lain sebagainya.

4.2. Hasil Pengumpulan data

Proses pengumpulan data tidak sepenuhnya berjalan dengan lancar. Beberapa kali sering terjadi masalah yang tidak berkaitan dengan model AI seperti stabilitas jaringan, masalah pada *code editor* bahkan isu teknis yang ada pada lingkungan pengembangan. Oleh karena itu, Data yang disajikan pada bagian ini merupakan data yang berhasil mengatasi kendala-kendala tersebut.

Pengumpulan data efisiensi diambil berdasarkan metrix yang telah ditetapkan pada Goal Question Metrix pada bab 3

Tabel 3. Hasil Percobaan

GPT 4.1				
Kode	Fitur	Jumlah *Prompt*	Rata-rata waktu (Menit)	Total Waktu (Menit)
F1	Manajemen Pengguna	2	00:54.8	01:49.6
F2	Manajemen Kategori & Subkategori	4	01:27.3	05:49.2
F3	Manajemen Tiket	6	01:18.0	07:47.9
F4	Manajemen Notifikasi	3	01:28.5	04:25.6
F5	Tambah Statistik Ticket pada user	3	01:08.6	03:25.9
F6	Manajemen Chat	2	01:40.3	03:20.5
F7	Manajemen FAQ	5	01:39.3	08:16.4
F8	Kirim email	4	01:30.8	06:03.1
F9	Tambah Token Ticket	2	02:56.2	05:52.3
F10	Tambah informasi chat	2	01:46.0	03:32.0
F11	Tambah Prioritas	3	01:29.0	04:27.1
F12	Log Aktivitas	2	01:58.6	03:57.1
	Keseluruhan	38	1:32.808	58:46.694
	Min	2	00:54.8	01:49.6
	Max	6	02:56.2	08:16.4
	Median	3	01:29.9	04:26.4
	Standar Deviasi	1.280190958	0.000334784	0.001269856

Claude 3.7				
Kode	Fitur	Jumlah *Prompt*	Rata-rata waktu (Menit)	Total Waktu (Menit)
F1	Manajemen Pengguna	4	03:43.6	14:54.3
F2	Manajemen Kategori & Subkategori	4	02:47.0	11:07.9
F3	Manajemen Tiket	4	03:06.8	12:27.1
F4	Manajemen Notifikasi	6	03:23.0	20:17.9
F5	Tambah Statistik Ticket pada user	3	01:34.6	04:43.8
F6	Manajemen Chat	5	03:54.0	19:30.0
F7	Manajemen FAQ	3	03:18.1	09:54.2
F8	Kirim email	3	01:50.4	05:31.2
F9	Tambah Token Ticket	3	02:44.8	08:14.3
F10	Tambah informasi chat	4	02:44.5	10:58.1
F11	Tambah Prioritas	3	04:25.6	13:16.9
F12	Log Aktivitas	2	03:17.8	06:35.7
	Keseluruhan	44	3:07.530	137:31.313
	Min	2	01:34.6	04:43.8
	Max	6	04:25.6	20:17.9
	Median	3.5	03:12.3	11:03.0
	Standar Deviasi	1.027402334	0.000538021	0.003332316

4.3. Pembahasan Hasil

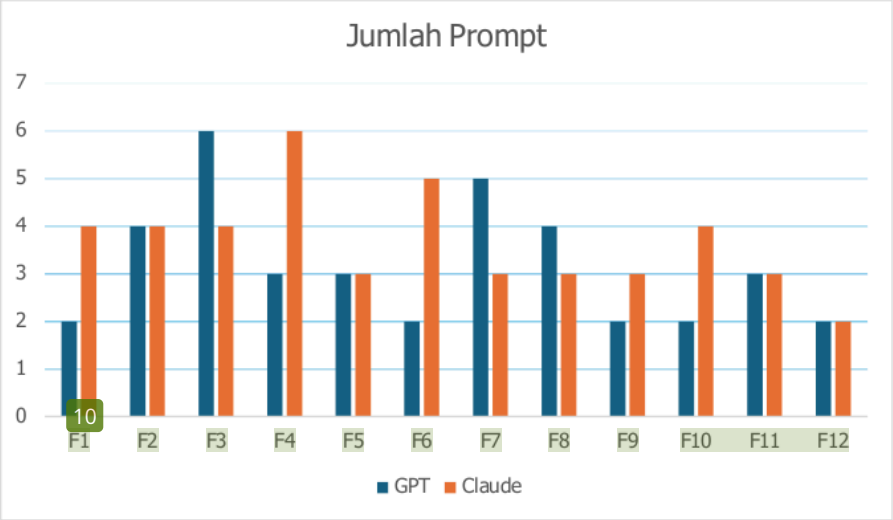
Hasil dari kode yang dibuat pada percobaan ini telah dilakukan dengan mempertimbangkan *correctness*, sehingga seluruh fitur yang diujikan pada penleitian ini sudah lolos unit/feature test yang dibuat saat proses *code generation* yang dilakukan oleh LLM. Bagian ini akan lebih lanjut membahas tentang hasil temuan yang didapatkan selama proses eksperimen.

4.3.1. Analisis Metrik Efisiensi Kuantitatif

Berikut analisis dari hasil percobaan terhadap 12 tugas yang dikembangkan. Analisis ini meliputi jumlah *prompt* serta waktu yang dibutuhkan pada proses pembuatan kode.

Analisis Jumlah *Prompt*

Dari proses *prompting* yang dilakukan, didapat grafik yang menunjukkan jumlah *prompt* yang diperlukan masing-masing model pada pengerjaan setiap tugas.



Gambar 2. Diagram Batang Jumlah *Prompt*

Grafik tersebut memperlihatkan bahwa secara umum, model GPT 4.1 memiliki jumlah *prompt* yang lebih sedikit dibandingkan model Claude 3.7. Pada GPT 4.1, secara keseluruhan jumlah *prompt* yang dibutuhkan adalah 38 kali, sedangkan pada Claude membutuhkan 44 kali *prompt* untuk menyelesaikan semua tugas yang diberikan peda percobaan ini.

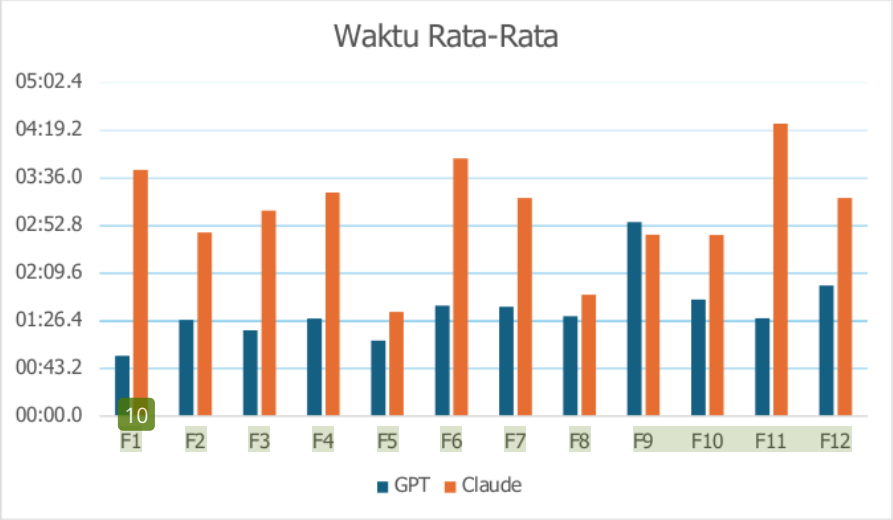
Berdasarkan percobaan yang telah dilakukan, penulis menyimpulkan bahwa kompleksitas fitur, konteks proyek, serta solusi yang ditawarkan oleh AI sangat berpengaruh pada jumlah *prompt* yang dibutuhkan. Hal ini juga sesuai dengan literatur yang digunakan oleh Sobo et al yang menjelaskan kalau kompleksitas tugas berpengaruh pada kinerja LLM [12]. Ada beberapa fitur spesifik yang mengakibatkan GPT lebih banyak melakukan *prompt* daripada Claude seperti manajemen tiket, FAQ dan kirim email. Fitur manajemen tiket dan FAQ pada model GPT membutuhkan jumlah *prompt* lebih banyak karena pada kedua fitur tersebut di model GPT, kode yang dihasilkan tidak langsung terbuat semua. Sehingga peneliti harus melakukan *prompt* tambahan untuk menyelesaikan semua bagian yang belum dikerjakan. Sementara pada fitur kirim email di model GPT, terdapat *error* pada

pembuatan kode yang menyebabkan AI berulang kali mencari solusi dari permasalahan tersebut.

Secara umum jumlah *prompt* terbanyak secara akumulatif ada pada Claude. Karena pada Claude, keluaran yang dihasilkan lebih komprehensif dan lebih lengkap daripada menggunakan GPT. Meskipun demikian, dari data yang telah dipaparkan menunjukkan bahwa performa dari masing-masing model pada jumlah *prompt* yang dilakukan tidak bersifat mutlak. tetapi juga sangat bergantung pada jenis tugas yang dikerjakan.

Analisis Waktu Pengerjaan

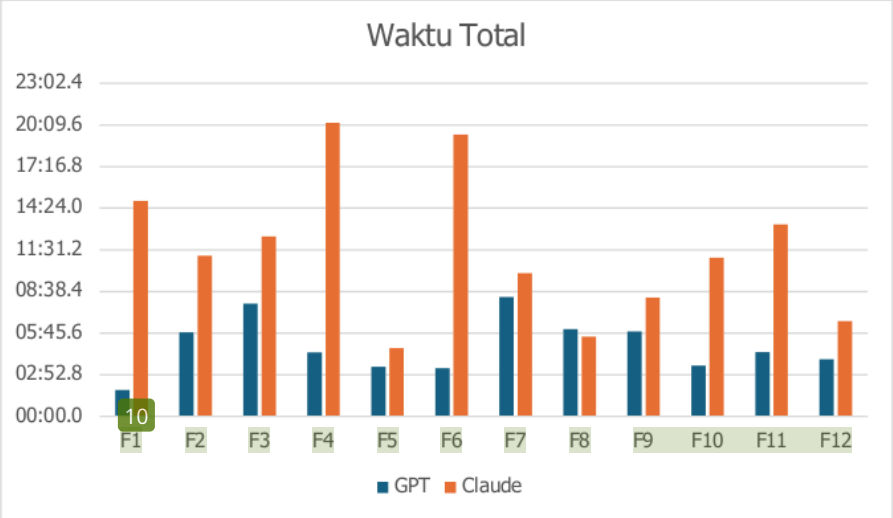
Berdasarkan waktu yang dihasilkan oleh masing-masing model melalui log (penjelasan cara mendapatkan data ada di bab 3 bagian Alur Proses Pelaksanaan), dapat dilihat bahwa GPT memiliki waktu eksekusi *prompt* yang lebih konsisten daripada Claude. Berikut merupakan grafik yang merepresentasikan waktu yang dibutuhkan tiap model proses *code generation* pada setiap tugas.



Gambar 3. Diagram Batang Waktu Rata-rata Per Fitur

Berdasarkan data tersebut, dapat disimpulkan bahwa waktu yang dibutuhkan oleh model Claude 3.7 lebih banyak daripada GPT 4.1. Claude 3.7 mendapati

rata-ratanya adalah 3 menit, sedangkan GPT 4.1 memiliki rata-rata selama sekitar 1,5 menit saja.



Gambar 4. Diagram Batang Waktu Total Tiap Fitur

Jika ditotal secara keseluruhan, waktu yang dibutuhkan Claude lebih banyak dibandingkan GPT. Waktu yang dibutuhkan Claude untuk menyelesaikan seluruh tugas ini adalah 137 menit, sedangkan pada GPT hanya butuh waktu 58 menit. Meskipun GPT memiliki durasi setiap *prompt* yang lebih cepat dalam berbagai macam fitur, ada banyak faktor yang di luar kendali dari model tersebut yang mempengaruhi waktu tunggunya.

Ada beberapa faktor eksternal yang mempengaruhi durasi waktu untuk menyelesaikan suatu *prompt*. Di antara faktornya sebagai berikut:

a. Waktu Eksekusi pada Terminal

Beberapa kasus, AI memilih untuk melakukan eksekusi beberapa hal pada terminal, seperti operasi *file* atau bahkan melakukan pengujian (*testing*) yang dapat menyebabkan waktu tunggunya menjadi lebih lama.

b. Waktu tunggu *trigger* dari pengguna

Pada Github Copilot Agent Mode, terkadang terdapat eksekusi *prompt* yang memerlukan persetujuan dari pengguna sebelum melakukan eksekusinya. seperti pembuatan folder atau lainnya yang dapat mempengaruhi waktu yang diperlukan untuk melakukan *prompting*

c. Kompleksitas konteks proyek dan fitur yang akan dibuat

Hal ini sangat berpengaruh pada durasi *prompt*. semakin kompleks fitur yang dibuat atau semakin kompleks *project* yang sudah ada, maka waktu untuk eksekusi *prompt*-nya semakin lama karena AI harus membaca konteks dari proyek tersebut, memikirkan solusi yang harus dilakukan pada proyek tersebut, dan membuat kode sesuai dengan kebutuhan.

d. Batas maksimum durasi setiap *prompt* pada model

Pada GPT, durasi paling lama dalam satu *prompt* itu ada di waktu sekitar dua setengah menit untuk satu kali *prompt*. Jika lebih dari 2 setengah menit atau sampai kurang dari tiga menit, AI akan meminta untuk melakukan *prompting* lagi (*Continue Iteration*). Sedangkan pada Claude paling lama 6 menit lebih untuk satu kali *prompt* sebelum akhirnya meminta untuk *prompt* berikutnya.

4.3.2. Analisis Kualitatif

Secara umum, peneliti mendapatkan hasil kode yang dibuat oleh masing-masing model dapat berbeda. Sebagai contoh, fitur manajemen tiket pada kedua model menghasilkan kode yang berbeda dan pendekatan yang berbeda, walaupun tetap sesuai dengan kebutuhan fitur. Kemudian pada fitur notifikasi, meskipun *prompt* awal yang dibuat sama, tetapi konteks proyek yang mulai berbeda menyebabkan kedua model tersebut tidak menghasilkan metode yang sama dalam membuat kode manajemen notifikasi. Pada Claude, fitur notifikasi menggunakan metode *service* biasa, sedangkan pada GPT menggunakan metode *events & listener.* Meskipun metode yang digunakan berbeda, setidaknya hasilnya kurang lebih sesuai dengan kebutuhan.

Peneliti juga menemukan beberapa kesimpulan secara kualitatif pada masing-masing model sebagai berikut:

GPT 4.1

- Efektivitas lebih tinggi untuk mengerjakan fitur yang tidak begitu kompleks. Kode yang dihasilkan terkadang ringkas dan langsung dapat menjawab kebutuhan.
- Akurasi dalam membuat kodenya konsisten untuk berbagai kesulitan dan juga di dukung dengan penelitian sebelumnya *pada Generate Code* untuk LLM4DS pada ChatGPT [19].
- Terkadang terdapat kode yang belum lengkap dibuat, jadi peneliti harus perlu melakukan *prompting* ulang untuk melengkapi kekurangan kode tersebut
- Kesulitan melakukan *debuging* kompleks, terutama jika dihadapkan pada *bug* yang terjadi pada proyek tersebut. Terkadang model ini melakukan solusi yang berulang ulang dan tidak efektif

Claude 3.7 Sonnet

- Bagus untuk fitur yang lebih kompleks, biasanya menghasilkan kode yang lebih lengkap daripada GPT.
- Pembuatan kode terkadang dilakukan secara berlebihan. Sebagai contoh pada fitur Manajemen kategori, yang peneliti butuhkan hanya fungsi *get* dan *post* saja, tetapi oleh model ini justru membuat lengkap CRUD-nya
- Anomali saat melakukan *unit test*. Model ini dapat menghasilkan test case yang cukup banyak. bahkan setelah semua fungsionalitas sudah terbukti benar dengan *unit test* yang dibuat sebelumnya, model ini tiba-tiba membuat *test case* yang lain yang mengakibatkan kode menjadi salah sehingga merusak *project* yang sudah benar. Penulis bahkan

harus menghentikan paksa proses *prompting* agar *error*-nya tidak semakin melebar.

Pada penelitian ini juga, pada *functional correctness* dilakukan validasi dengan menggunakan *unit/feature test* yang juga dibuat oleh AI. Fitur atau *task* yang diberikan dianggap selesai hanya ketika semua *test* yang sudah dibuat berhasil dijalankan dengan baik. Sehingga dari 12 *task* pada penelitian ini berhasil memenuhi kriteria dari *test* yang sudah ditentukan.

Secara umum, model GPT sepertinya sudah didukung dengan baik oleh Github Copilot. Hal itu disebabkan karena ketika *prompting* dilakukan, Github Copilot dapat menampilkan proses pengkodean dari *file* yang diubah. Sedangkan pada Claude tidak secara langsung menampilkan perubahannya, tetapi perubahan tersebut muncul ketika satu file tersebut sudah sesleai melakukan *code generation.*

BAB 5 KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan hasil eksperimen pengembangan *backend* Rest API menggunakan Github Copilot dengan mamastikan *correctness* dari kode yang dibuat bedasarkan *unit/feature test*, kesimpulan utama untuk menjawab rumusan masalah yaitu sebagai berikut:

a. Perbandingan jumlah *prompt*

Berdasarkan data yang didapat, model Claude 3.7 memerlukan proses *prompt* yang lebih banyak dibandingkan dengan GPT 4.1 –Claude 3.7 lebih banyak 6 *prompt* daripada GPT– sehingga dapat disimpulkan bahwa GPT 4.1 lebih efisien dibanding model Claude 3.7 Sonnet. Akan tetapi, Claude 3.7 menyelesaikan tugas lebih andal dibandingkan GPT 4.1, terutama untuk fitur yang lebih kompleks.

b. Perbandingan durasi setiap *prompt*

Pada durasi setiap *prompt,* GPT secara signifikan lebih cepat dibandingkan Claude 3.7. Angka tersebut dapat dilihat dari rata-rata maupun akumulasi keseluruhan, yaitu perbedaannya adalah 78 menit untuk waktu total dan 1,5 menit untuk rata-rata. Hal ini disebabkan karena durasi maksimal yang dibutuhkan GPT 4.1 pada satu kali *prompt* lebih singkat dibanding Claude 3.7

c. Kesimpulan model yang menangani efisiensi terbaik

Pada penelitian ini, GPT 4.1 terbukti lebih efisien untuk melakukan tugas yang ringkas dan tidak begitu besar. Keunggulan ini dilihat dari jumlah *prompt* yang umumnya lebih sedikit serta durasi setiap *prompt* yang lebih singkat.

Meskipun durasi *prompt* yang dibutuhkan lebih banyak, Claude 3.7 lebih efektif untuk melakukan pembuatan kode yang berskala lebih besar pada satu kali *prompt*. Perlu dicatat bahwa efisiensi antara model AI bergantung juga pada tingkat kompleksitas fitur AI konteks yang diberikan pada AI.

5.2. Batasan Penelitian

Dari percobaan yang dilakukan, peneliti menyadari bahwa penelitian ini memiliki beberapa keterbatasan yang perlu dipertimbangkan. Keterbatasan tersebut di antaranya:

a. Lingkup Proyek yang terbatas

Penelitian ini hanya berfokus pada lingkup pengembangan backend Rest API menggunakan Laravel dengan tugas yang spesifik. Oleh karena itu, untuk generalisasi hasil penelitian ke bahasa pemrograman, *framework* atau pengembangan yang lain perlu memerlukan validasi lebih lanjut.

b. Potensi Subjektivitas Peneliti

Proses evaluasi dan perhitungan jumlah *prompt* yang dilakukan oleh peneliti besar kemungkinan mengandung hasil yang bersifat subjektif. Meskipun demikian, peneliti berusaha sebisa mungkin mengupayakan proses penelitian dengan objektif.

c. Metrik yang terbatas

Pengukuran efisiensi berfokus pada jumlah *prompt* dan durasi setiap *prompt.* Penelitian ini belum menganalisis secara mendalam aspek lainnya seperti kualitas dari kode, *maintability*, keamanan, performansi dan lain sebagainya.

d. Perkembangan Teknologi

Mengingat kecepatan evolusi AI yang sangat pesat, hasil penelitian ini merefleksikan performa model pada versi dan waktu tertentu. Relevansi dari performa model AI dapat berubah pada waktu dan versi tertentu, sehingga relevansi dari penelitian ini dapat berkurang seiring berjalannya waktu.

e. Kesalahan minor

Pada proses *code generation* ini tidak luput pada kesalahan minor sehingga beberapa bagian yang tidak sepenuhnya sama persis dengan *requirement* awal. Oleh karena itu, kriteria keberhasilan pada pengujiannya ditentukan

berdasarkan keberhasilan dari skenario pengujian unit test yang dihasilkan oleh AI.

5.3. Saran

Setiap model memiliki kekurangan dan kelebihannya masing-masing. Alangkah lebih baik lagi jika di kombinasikan penggunaannya berdasarkan tingkat kompleksitas dari fitur yang dibuat dengan karakteristik dari masing-masing model AI dibanding memilih satu model AI untuk mengerjakan semua proyek.

Untuk memperdalam penelitian tentang bidang ini, berikut beberapa arah penelitian yang dapat dieksplorasi ke depannya:

a. Evaluasi pada model AI terbaru

Mengingat perkembangan AI yang sangat pesat, penelitian selanjutnya sangat dianjurkan untuk melakukan evaluasi ulang AI pada model terbaru untuk memastikan relevansi temuan.

b. Percobaan dengan konteks berbeda

Untuk menguji generalisasi temuan, penelitian ini dapat di replikasi dengan *framework*, bahasa pemrograman, atau konteks proyek yang berbeda yang berbeda. Pengujian ini dilakukan untuk mengetahui konsistensi dari tingkat efisiensinya pada model yang diujikan.

c. Perluas Metrik Evaluasi

Penelitian ke depan disarankan untuk memperluas metrik pengujian di luar efisiensi yang sudah ditentukan sebelumnya. Metrik yang dimaksud diharapkan dapat mencakup analisis terhadap kualitas kode yang dihasilkan seperti performansi kode, kompleksitas, kemudahan, keamanan dan lain sebagainya.

d. Penerapan MCP pada pengujian model

Akhir-akhir ini muncul teknologi yang bernama Model Context Protocol pada ranah *code generation* yang dapat meningkatkan efektivitas dan efisiensi proses *code generation* menggunakan AI. Penelitian selanjutnya dapat mengukur dampak dari penerapan MCP pada proses *Code Generation*.

e. Penggunaan bahasa inggris untuk melakukan *prompting*

Penelitian yang dilakukan saat ini masih menggunakan bahasa Indonesia sebagai bahasa yang digunakan pada proses *prompting*. Untuk penelitian berikutnya dapat dicoba menggunakan bahasa lain pada penggunaan LLM.

f. Validasi dengan lebih dari satu developer

Salah satu keterbatasan penelitian ini adalah dilakukan oleh satu orang peneliti. Sedangkan pada *prompt* lanjutan -- untuk *follow up* jika kode yang dibuat belum memenuhi *requirement*-- sangat bergantung pada *developer* yang melakukannya. Oleh karena itu perlu melibatkan lebih dari seatu *developer* untuk menjaga konsistensi dan reabilitas temuan.

DAFTAR PUSTAKA

[1] Y. Grace, E. Wen, and H. Sun, "Comparative Analysis of ChatGPT-4 and Gemini Advanced in Erroneous Code Detection and Correction," 2024.

[2] R. Dwi Natasya, "IMPLEMENTASI ARTIFICIAL INTELLIGENCE (AI) DALAM TEKNOLOGI MODERN," *Jurnal Komputer dan Teknologi Sains (KOMTEKS)*, vol. 2, no. 1, pp. 22–24, Dec. 2023, [Online]. Available: https://ojs.unm.ac.id/pengabdi/article/view/46

[3] M. K. Siam, H. Gu, and J. Q. Cheng, "Programming with AI: Evaluating ChatGPT, Gemini, AlphaCode, and GitHub Copilot for Programmers," *IEEE International Conference on Program Comprehension*, vol. 2022-March, pp. 36–47, Nov. 2024, Accessed: Apr. 30, 2025. [Online]. Available: http://arxiv.org/abs/2411.09224

[4] J. Jiang, F. Wang, J. Shen, S. Kim, and S. Kim, "A Survey on Large Language Models for Code Generation," Jun. 2024, [Online]. Available: http://arxiv.org/abs/2406.00515

[5] B. Yetiştiren, I. Özsoy, M. Ayerdem, and E. Tüzün, "Evaluating the Code Quality of AI-Assisted Code Generation Tools: An Empirical Study on GitHub Copilot, Amazon CodeWhisperer, and ChatGPT," Apr. 2023, [Online]. Available: http://arxiv.org/abs/2304.10778

[6] R. Pandey, P. Singh, R. Wei, and S. Shankar, "Transforming Software Development: Evaluating the Efficiency and Challenges of GitHub Copilot in Real-World Projects," Jun. 2024, [Online]. Available: http://arxiv.org/abs/2406.17910

[7] K. Mohamed, M. Yousef, W. Medhat, E. H. Mohamed, G. Khoriba, and T. Arafa, "Hands-on analysis of using large language models for the auto evaluation of programming assignments," Feb. 01, 2025, *Elsevier Ltd*. doi: 10.1016/j.is.2024.102473.

[8] M. Cataldo and J. D. Herbsleb, "Factors leading to integration failures in global feature-oriented development: an empirical analysis," in *2011 33rd International Conference on Software Engineering (ICSE)*, 2011, pp. 161–170. doi: 10.1145/1985793.1985816.

[9] M. Chen *et al.*, "Evaluating Large Language Models Trained on Code," Jul. 2021, [Online]. Available: http://arxiv.org/abs/2107.03374

[10] D. Huang, Y. Qing, W. Shang, H. Cui, and J. M. Zhang, "EffiBench: Benchmarking the Efficiency of Automatically Generated Code," May 2025, [Online]. Available: http://arxiv.org/abs/2402.02037

[11] L. B. Heitz, J. Chamas, and C. Scherb, "Evaluation of the Programming Skills of Large Language Models," May 2024, [Online]. Available: http://arxiv.org/abs/2405.14388

[12] A. Sobo, A. Mubarak, A. Baimagambetov, and N. Polatidis, "Evaluating LLMs for Code Generation in HRI: A Comparative Study of ChatGPT, Gemini, and Claude," *Applied Artificial Intelligence*, vol. 39, no. 1, 2025, doi: 10.1080/08839514.2024.2439610.

[13] N. Nguyen and S. Nadi, "An Empirical Evaluation of GitHub Copilot's Code Suggestions," in *Proceedings - 2022 Mining Software Repositories Conference, MSR 2022*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 1–5. doi: 10.1145/3524842.3528470.

[14] C. Sriwilailak, Y. Higo, P. Lapvikai, C. Ragkhitwetsagul, and M. Choetkiertikul, "Autorepairability of ChatGPT and Gemini: A Comparative Study."

[15] S. Schulhoff *et al.*, "The Prompt Report: A Systematic Survey of Prompt Engineering Techniques," Jun. 2024, [Online]. Available: http://arxiv.org/abs/2406.06608

[16] H. Xu *et al.*, "Large Language Models for Software Engineering: A Systematic Literature Review," *ACM Transactions on Software Engineering and Methodology*, Dec. 2024, doi: 10.1145/3695988.

[17] F. Liu *et al.*, "Exploring and Evaluating Hallucinations in LLM-Powered Code Generation," Apr. 2024.

[18] Z. C. Ani, Z. A. Hamid, and N. N. Zhamri, "The Recent Trends of Research on GitHub Copilot: A Systematic Review," 2024, pp. 355–366. doi: 10.1007/978-981-99-9589-9_27.

[19] N. Nascimento, E. Guimaraes, S. S. Chintakunta, and S. A. Boominathan, "LLM4DS: Evaluating Large Language Models for Data Science Code Generation," Nov. 2024, [Online]. Available: http://arxiv.org/abs/2411.11908

[20] R. T. Fielding and R. N. Taylor, "Architectural styles and the design of network-based software architectures," 2000.

[21] Ian. Sommerville, *Software engineering*. Pearson, 2011.

[22] C. Wang *et al.*, "Teaching Code LLMs to Use Autocompletion Tools in Repository-Level Code Generation," Jul. 2024, [Online]. Available: http://arxiv.org/abs/2401.06391

[23] M.-S. Vasiliniuc and A. Groza, "Case Study: Using AI-Assisted Code Generation In Mobile Teams," Sep. 2023, [Online]. Available: http://arxiv.org/abs/2308.04736

[24] W. P. Zayu, H. Herman, and G. Vitri, "Studi Komparatif Pelaksanaan Tugas Besar Perencanaan Geometrik Jalan Secara Daring Dan Luring," *Jurnal Penelitian Dan Pengkajian Ilmiah Eksakta*, vol. 2, no. 1, pp. 92–96, Feb. 2023, doi: 10.47233/jppie.v2i1.762.

[25] J. D. Velásquez-Henao, C. J. Franco-Cardona, and L. Cadavid-Higuita, "Prompt Engineering: a methodology for optimizing interactions with AI-Language Models in the field of engineering," *Dyna (Medellin)*, vol. 90, no. 230, pp. 9–17, Nov. 2023, doi: 10.15446/dyna.v90n230.111700.

LAMPIRAN

Lampiran 01. Tabel perhitungan

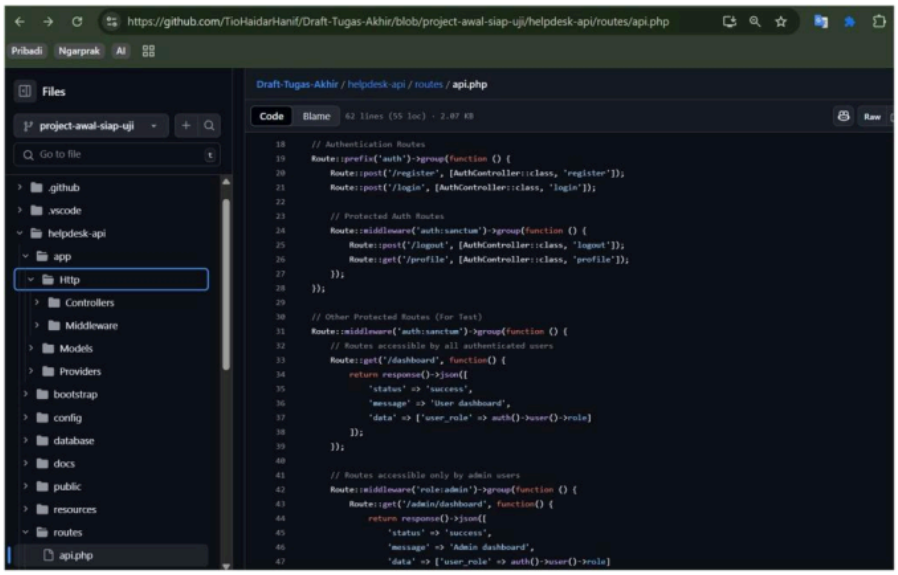
GPT 4.1							
Fitur	**Prompt**	**Log**	**Waktu (MS)**	**Readable Time**	**Keterangan**	**Hasil**	**Readable**
Manajemen Pengguna	Saya ingin mengembangkan fitur "Manajem	2025-05-23 01	96656	1:36.656	Jumlah Iterasi	2	
	Continue Iteration	2025-05-23 01	12894	0:12.894	Waktu Total	109550	1:49.550
					Rata rata	54775	0:54.775
Manajemen Kategori dan Subk	Saya ingin mengembangkan fitur "Manajem	2025-05-24 02	60107	1:00.107	Jumlah Iterasi	4	
	Continue Iteration	2025-05-24 02	142173	2:22.173	Waktu Total	349199	5:49.199
	Continue Iteration	2025-05-24 02	104827	1:44.827	Rata rata	87300	1:27.300
	pada fitur manajemen kategori, tolong ses	2025-05-24 01	42092	0:42.092			
Manajemen Ticket	agentSaya ingin mengembangkan fitur "Ma	2025-05-24 04	71459	1:11.459	Jumlah Iterasi	6	
	Continue Iteration	2025-05-24 04	92672	1:32.672	Waktu Total	467889	7:47.889
	lakukan implementasi pada #file:TicketCor	2025-05-24 04	153655	2:33.655	Rata rata	77982	1:17.982
	Continue Iteration	2025-05-24 04	60675	1:00.675			
	Belum semua API pada #file:api.php untuk	2025-05-24 04	49996	0:49.996			
	tolong perbaiki agar attachment dari ticket	2025-05-24 04	39432	0:39.432			
Manajemen Notifikasi	Saya ingin mengembangkan fitur "Manajem	2025-05-26 06	66047	1:06.047	Jumlah Iterasi	3	
	Continue Iteration	2025-05-26 06	158076	2:38.076	Waktu Total	265585	4:25.585
	Continue Iteration	2025-05-26 06	41462	0:41.462	Rata rata	88528	1:28.528
Tambah Statistik Ticket pada	agentSaya ingin mengedit fitur manajemen	2025-05-29 02	70847	1:10.847	Jumlah Iterasi	3	
TioHaidarHanif: agentSaya ingin mer	Continue Iteration	2025-05-29 02	106113	1:46.113	Waktu Total	205881	3:25.881
	pada endpoint user/{id}, di setiap ticket yan	2025-05-29 02	28921	0:28.921	Rata rata	68627	1:08.627
Manajemen Chat	agentSaya ingin mengembangkan fitur ""M	2025-05-30 13	168976	2:48.976	Jumlah Iterasi	2	
TioHaidarHanif: agentSaya ingin mer	Continue Iteration	2025-05-30 13	31553	0:31.553	Waktu Total	200529	3:20.529
					Rata rata	100265	1:40.265
Manajemen FAQ	agentSaya ingin mengembangkan fitur ""M	2025-05-30 14	112670	1:52.670	Jumlah Iterasi	5	
	lanjutkan ke langkah selanjutnya	2025-05-30 14	135588	2:15.588	Waktu Total	496404	8:16.404
	Continue Iteration	2025-05-30 14	72971	1:12.971	Rata rata	99281	1:39.281
	lanjutkan untuk membuat test pada setiap	2025-05-30 15	123723	2:03.723			
	ubah urutan posisi Ploute::get('/faqs/categ	2025-05-30 15	51452	0:51.452			
Kirim email	agentSaya ingin mengembangkan fitur "Kirir	2025-06-08 14	76506	1:16.506	Jumlah Iterasi	4	
TioHaidarHanif: agentSaya ingin mer	Continue Iteration	2025-06-08 14	94985	1:34.985	Waktu Total	363143	6:03.143
	Continue Iteration	2025-06-08 14	169718	2:49.718	Rata rata	90786	1:30.786
	file #file:ManualMailTest.php dihapus saja,	2025-06-08 14	21934	0:21.934			
Tambah Token Ticket	Saya ingin mengembangkan fitur "Token Rahasia	[illegible]	156876	2:36.876	Jumlah Iterasi	2	
	Continue Iteration	[illegible]	195429	3:15.429	Waktu Total	352305	5:52.305
					Rata rata	176153	2:56.153
Tambah informasi chat	Saya ingin mengembangkan fitur "Informasi Jumla	[illegible]	141156	2:21.156	Jumlah Iterasi	2	
TioHaidarHanif: Saya ingin mengembangka	Continue Iteration	[illegible]	70816	1:10.816	Waktu Total	211972	3:31.972
					Rata rata	105986	1:45.986
Tambah Prioritas	Saya ingin mengembangkan fitur "Prioritas Ticket	[illegible]	84810	1:24.810	Jumlah Iterasi	3	
TioHaidarHanif: Saya ingin mengembangka	Continue Iteration	[illegible]	163595	2:43.595	Waktu Total	267129	4:27.129
	FAILED Tests\Feature\TicketPriorityTest > cre	[illegible]	18724	0:18.724	Rata rata	89043	1:29.043
Log Aktivitas	Saya ingin mengembangkan fitur "Log Aktivitas A	[illegible]	137385	2:17.385	Jumlah Iterasi	2	
TioHaidarHanif: Saya ingin mengembangka	masih ada error pada : FAILED Tests\Feature\	[illegible]	99723	1:39.723	Waktu Total	237108	3:57.108
					Rata rata	118554	1:58.554
Update Batasan Lampiran	Saya ingin memperbaiki fitur upload file pada tick	[illegible]	67339	1:07.339			
Berikut hasil identifikasi dan rencana perbi							

Claude 3.7 Sonnet							
Fitur	**Prompt**	**Log**	**Waktu (MS)**	**Readable Time**	**Keterangan**	**Hasil**	**Readable**
Manajemen Pengguna	Saya ingin meng	2025-05-23 02	156114	2:36.114	Jumlah iterasi	4	
	Continue Iterati	2025-05-23 03	292194	4:52.194	Waktu total	894289	14:54.289
	Continue Iterati	2025-05-23 03	278959	4:38.959	Rata-rata	223572	3:43.572
	Tolong buatkan	2025-05-23 03	167022	2:47.022			
Manajemen Kategori dan Subk	Saya ingin meng	2025-05-23 10	177474	2:57.474	Jumlah iterasi	4	
	Continue Iterati	2025-05-23 10	243611	4:03.611	Waktu total	667869	11:07.869
	test_get_all_cat	2025-05-23 12	165916	2:45.916	Rata-rata	166967	2:46.967
	Continue Iterati	2025-05-23 12	80868	1:20.868			
Manajemen Ticket	agentSaya ingin	2025-05-24 03	186713	3:06.713	Jumlah iterasi	4	
	Continue Iterati	2025-05-24 03	21540	0:21.540	Waktu total	747068	12:27.068
	Retry prompt	2025-05-24 03	437226	7:17.226	Rata-rata	186767	3:06.767
	Continue Iterati	2025-05-24 03	101589	1:41.589			
Manajemen Notifikasi	Tentu, berikut p	2025-05-26 04	131702	2:11.702	Jumlah iterasi	6	
	Continue Iterati	2025-05-26 04	209048	3:29.048	Waktu total	1217928	20:17.928
	Continue Iterati	2025-05-26 04	181924	3:01.924	Rata-rata	202988	3:22.988
	lanjut buatkan te	2025-05-26 05	254634	4:14.634			
	Continue Iterati	2025-05-26 05	155964	2:35.964			
	Continue Iterati	2025-05-26 06	284656	4:44.656			
Tambah Statistik Ticket pada	agentSaya ingin	2025-05-29 03	99341	1:39.341	Jumlah iterasi	3	
TioHaidarHanif: agentSaya ingin mer	Continue Iterati	2025-05-29 03	103977	1:43.977	Waktu total	283848	4:43.848
	pada endpoint u	2025-05-29 03	80530	1:20.530	Rata-rata	94616	1:34.616
Manajemen Chat	agentSaya ingin	2025-05-30 12	130899	2:10.899	Jumlah iterasi	5	
TioHaidarHanif: agentSaya ingin mer	Continue Iterati	2025-05-30 12	387881	6:27.881	Waktu total	1169964	19:29.964
	Continue Iterati	2025-05-30 12	368544	6:08.544	Rata-rata	233993	3:53.993
	Terdapat error d	2025-05-30 13	46464	0:46.464			
	Terdapat error d	2025-05-30 13	236176	3:56.176			
Manajemen FAQ	agentSaya ingin	2025-05-30 14	131182	2:11.182	Jumlah iterasi	3	
TioHaidarHanif: agentSaya ingin mer	Continue	2025-05-30 14	166336	2:46.336	Waktu total	594182	9:54.182
	Continue Iterati	2025-05-30 14	296664	4:56.664	Rata-rata	198061	3:18.061
Kirim email	agentSaya ingin	2025-06-08 14	87107	1:27.107	Jumlah iterasi	3	
TioHaidarHanif: agentSaya ingin mer	continue iteratio	2025-06-08 14	173652	2:53.652	Waktu total	331225	5:31.225
	terdapat error T	2025-06-08 14	70466	1:10.466	Rata-rata	110408	1:50.408
Token Tiket	Saya ingin menger	[illegible]	103123	1:43.123	Jumlah iterasi	3	
TioHaidarHanif: Saya ingin mengembangka	continue iteration	2025-06-18 01:	270370	4:30.370	Waktu total	494277	8:14.277
	hapus use Illuminat	[illegible]	120784	2:00.784	Rata-rata	164759	2:44.759
Tambah informasi chat	Saya ingin menger	[illegible]	139715	3:19.715	Jumlah iterasi	4	
TioHaidarHanif: Saya ingin mengembangka	continue iteration	[illegible]	209196	3:29.196	Waktu total	658103	10:58.103
	masih terdapat err	[illegible]	117170	1:57.170	Rata-rata	164526	2:44.526
	SQLSTATE[HY00	[illegible]	132022	2:12.022			
Tambah Prioritas	Saya ingin menger	[illegible]	228480	3:48.480	Jumlah iterasi	3	
TioHaidarHanif: Saya ingin mengembangka	continue iteration	[illegible]	315962	5:15.962	Waktu total	796883	13:16.883
	FAILED Tests\F	[illegible]	252441	4:12.441	Rata-rata	265628	4:25.628
Log Aktivitas	Saya ingin menger	[illegible]	168681	2:48.681	Jumlah iterasi	2	
TioHaidarHanif: Saya ingin mengembangka	continue iteration	[illegible]	226996	3:46.996	Waktu total	395677	6:35.677
					Rata-rata	197839	3:17.839
Update Batsan Lampiran	Saya ingin memper	[illegible]	149475	2:29.475			
TioHaidarHanif: Saya ingin memperbaiki fit	.						

https://tiny.cc/WorkspaceTATioHaidar

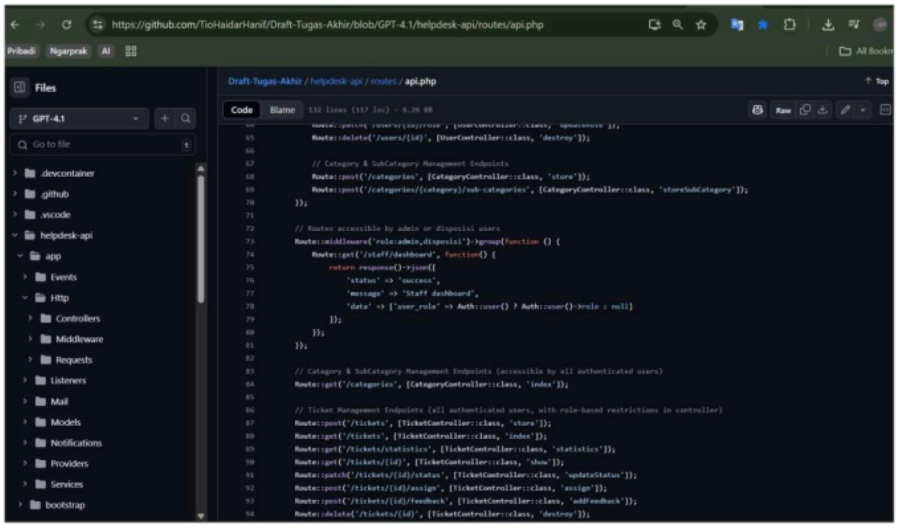
Lampiran 02. Sumber Kode

Project Awal



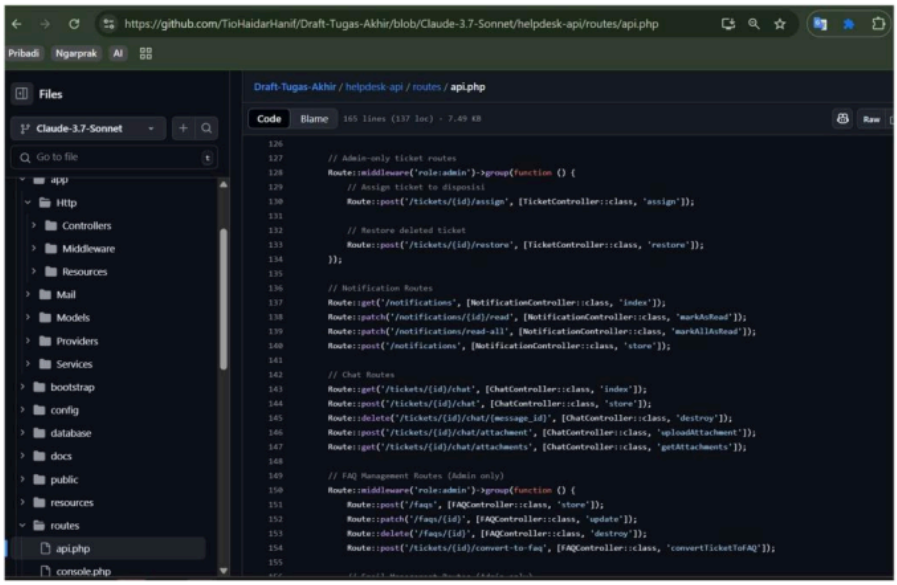
https://github.com/TioHaidarHanif/Draft-Tugas-Akhir/tree/project-awal-siap-uji

Project GPT 4.1



https://github.com/TioHaidarHanif/Draft-Tugas-Akhir/tree/GPT-4.1

Project Claude 3.7 Sonnet



https://github.com/TioHaidarHanif/Draft-Tugas-Akhir/tree/Claude-3.7-Sonnet

1302210057

ORIGINALITY REPORT

13%
SIMILARITY INDEX

PRIMARY SOURCES

1	Submitted to Telkom University Your Indexed Documents	118 words — 1%
2	arxiv.org Internet	110 words — 1%
3	www.tempo.co Internet	110 words — 1%
4	jyx.jyu.fi Internet	108 words — 1%
5	Submitted to Telkom University Your Indexed Documents	96 words — < 1%
6	www.schneider.im Internet	86 words — < 1%
7	text-id.123dok.com Internet	76 words — < 1%
8	eprints.uny.ac.id Internet	74 words — < 1%
9	www.coursehero.com Internet	74 words — < 1%
10	www.mdpi.com Internet	

		72 words — < 1%
11	Submitted to Telkom University Your Indexed Documents	68 words — < 1%
12	openlibrary.telkomuniversity.ac.id Internet	62 words — < 1%
13	repository.its.ac.id Internet	62 words — < 1%
14	www.theseus.fi Internet	62 words — < 1%
15	yourgpt.ai Internet	56 words — < 1%
16	asrjetsjournal.org Internet	44 words — < 1%
17	huggingface.co Internet	40 words — < 1%
18	repository.radenintan.ac.id Internet	38 words — < 1%
19	dspace.uii.ac.id Internet	36 words — < 1%
20	repository.unika.ac.id Internet	36 words — < 1%
21	Submitted to Telkom University Your Indexed Documents	32 words — < 1%
22	repository.ub.ac.id	

Internet

32 words — < 1%

23 Ilona Petersone, Elina Dovgaluka, Justine Gudzuka, Roberts Kartenko, Andrejs Romanovs. "An Analysis of IT Outsourcing Risks in Post-COVID World", 2023 IEEE 64th International Scientific Conference on Information Technology and Management Science of Riga Technical University (ITMS), 2023
Crossref

30 words — < 1%

24 repository.ubharajaya.ac.id
Internet

28 words — < 1%

25 www.yuinfo.org
Internet

28 words — < 1%

26 journal.politeknik-pratama.ac.id
Internet

27 words — < 1%

27 "Marketing and Smart Technologies", Springer Science and Business Media LLC, 2025
Crossref

26 words — < 1%

28 es.scribd.com
Internet

26 words — < 1%

29 www.publications.scrs.in
Internet

25 words — < 1%

30 id.scribd.com
Internet

24 words — < 1%

31 jom.fti.budiluhur.ac.id
Internet

24 words — < 1%

32 repositori.ubs-ppni.ac.id:8080
Internet

		24 words — < 1%
33	www.fujinet.net Internet	24 words — < 1%
34	Submitted to Telkom University Your Indexed Documents	22 words — < 1%
35	Submitted to Telkom University Your Indexed Documents	22 words — < 1%
36	depositonce.tu-berlin.de Internet	22 words — < 1%
37	id.123dok.com Internet	22 words — < 1%
38	katalog.ukdw.ac.id Internet	22 words — < 1%
39	repository.mercubuana.ac.id Internet	22 words — < 1%
40	repository.uigm.ac.id Internet	22 words — < 1%
41	repository.uph.edu Internet	22 words — < 1%
42	ai4cyber.eu Internet	21 words — < 1%
43	hyatiy.top Internet	20 words — < 1%
44	lontar.ui.ac.id	

	Internet	20 words — < 1%
45	repository.unja.ac.id Internet	20 words — < 1%
46	digilib.uinsby.ac.id Internet	18 words — < 1%
47	miftahulimam.wordpress.com Internet	18 words — < 1%
48	repository.eka-prasetya.ac.id Internet	18 words — < 1%
49	repository.uksw.edu Internet	18 words — < 1%
50	repository.unuja.ac.id Internet	18 words — < 1%
51	123dok.com Internet	16 words — < 1%
52	Submitted to Telkom University Your Indexed Documents	16 words — < 1%
53	digilib.uin-suka.ac.id Internet	16 words — < 1%
54	e-journal.lppmdianhusada.ac.id Internet	16 words — < 1%
55	eprints.kwikkiangie.ac.id Internet	16 words — < 1%
56	eprints.uns.ac.id	

Internet

16 words — < 1%

57 etd.uinsyahada.ac.id
Internet

16 words — < 1%

58 jurnal.staibsllg.ac.id
Internet

16 words — < 1%

59 kth.diva-portal.org
Internet

16 words — < 1%

60 www.doctorwish.com
Internet

16 words — < 1%

61 www.slideshare.net
Internet

16 words — < 1%

62 Mohammed M. Kanani, Arezu Monawer, Lauryn Brown, William E. King et al. "High-Performance Prompting for LLM Extraction of Compression Fracture Findings from Radiology Reports", Journal of Imaging Informatics in Medicine, 2025
Crossref

13 words — < 1%

63 J. Scholtz, M. Hansen. "Usability testing a minimal manual for the Intel SatisFAXtion faxmodem", IEEE Transactions on Professional Communication, 1993
Crossref

12 words — < 1%

64 repository.usd.ac.id
Internet

12 words — < 1%

65 stmik-budidarma.ac.id
Internet

12 words — < 1%

66 infotech.org.rs
Internet

11 words — < 1%

67 Azeeza Eagal, Kathryn T. Stolee, John-Paul Ore. "Analyzing the dependability of Large Language Models for code clone generation", Journal of Systems and Software, 2025 — 10 words — < 1%
Crossref

68 Yuan Gao, Dokyun Lee, Gordon Burtch, Sina Fazelpour. "Take caution in using LLMs as human surrogates", Proceedings of the National Academy of Sciences, 2025 — 9 words — < 1%
Crossref

69 htwk-leipzig.qucosa.de — 9 words — < 1%
Internet

70 polynoe.lib.uniwa.gr — 9 words — < 1%
Internet

71 repository.uin-suska.ac.id — 9 words — < 1%
Internet

72 www.utupub.fi — 9 words — < 1%
Internet

73 llm-guidelines.org — 8 words — < 1%
Internet

74 revistas.ulasalle.edu.pe — 8 words — < 1%
Internet

75 warse.org — 8 words — < 1%
Internet

76 Chong Wang, Jian Zhang, Yebo Feng, Tianlin Li, Weisong Sun, Yang Liu, Xin Peng. "Teaching Code LLMs to Use Autocompletion Tools in Repository-Level Code Generation", ACM Transactions on Software Engineering and Methodology, 2025

5 words — < 1%

Crossref

EXCLUDE QUOTES	ON	EXCLUDE SOURCES	OFF
EXCLUDE BIBLIOGRAPHY	ON	EXCLUDE MATCHES	OFF