# RANCANG BANGUN *VISUAL STUDIO CODE EXTENSION CODEVA* UNTUK *CODE GENERATOR JAVA* SECARA OTOMATIS MENGGUNAKAN *LARGE LANGUAGE MODELS* (LLMs)

**TUGAS AKHIR**


**DISUSUN OLEH:**

**MUHAMMAD ARIF RAKHMAN AZIZI**

**NIM. 21106050042**

**PROGRAM STUDI INFORMATIKA**

**FAKULTAS SAINS DAN TEKNOLOGI**

**UNIVERSITAS ISLAM NEGERI SUNAN KALIJAGA**

**YOGYAKARTA**




**2025**

# LEMBAR PENGESAHAN

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

## PENGESAHAN TUGAS AKHIR

Nomor : B-1808/Un.02/DST/PP.00.9/08/2025

Tugas Akhir dengan judul : Rancang Bangun Visual Studio Code Extension CodeVa Untuk Code Generator Java Secara Otomatis Menggunakan Large Language Models (LLMs)

yang dipersiapkan dan disusun oleh:

Nama : MUHAMMAD ARIF RAKHMAN AZIZI
Nomor Induk Mahasiswa : 21106050042
Telah diujikan pada : Kamis, 14 Agustus 2025
Nilai ujian Tugas Akhir : A

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

### TIM UJIAN TUGAS AKHIR

Ketua Sidang

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

Valid ID: 68a5508bb590c

Penguji I

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

Valid ID: 68a500868858fe

Penguji II

Eko Hadi Gunawan, M.Eng.
SIGNED

Valid ID: 689ff0ba97f6d

Yogyakarta, 14 Agustus 2025
UIN Sunan Kalijaga
Dekan Fakultas Sains dan Teknologi

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

Valid ID: 68a56937ac42e

# LEMBAR PERNYATAAN

**SURAT PERNYATAAN KEASLIAN/BEBAS PLAGIASI**

Yang bertanda tangan di bawah ini:

| | |
|---|---|
| Nama | : Muhammad Arif Rakhman Azizi |
| NIM | : 21106050042 |
| Program Studi | : Informatika |
| Fakultas | : Sains dan Teknologi |

Dengan ini menyatakan bahwa isi tugas akhir saya yang berjudul "Rancang Bangun *Visual Studio Code Extension CodeVa* Untuk *Code Generator Java* Secara Otomatis Menggunakan *Large Language Models* (LLMs)" merupakan hasil pekerjaan penulis sendiri, tidak terdapat karya yang pernah diajukan untuk memperoleh gelar sarjana di suatu Perguruan Tinggi, bukan duplikasi atau saduran dari karya orang lain, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan orang lain sepanjang pengetahuan penulis, kecuali yang secara tertulis diacu dalam tugas akhir ini dan disebutkan dalam daftar pustaka.

Yogyakarta, 12 Agustus 2025


Muhammad Arif Rakhman Azizi

NIM. 21106050042





ii

# LEMBAR PERSETUJUAN TUGAS AKHIR

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

**SURAT PERSETUJUAN SKRIPSI/TUGAS AKHIR**

Hal : Persetujuan Skripsi /Tugas Akhir
Lamp : -

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

*Assalammu'alaikum wr. wb.*

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

Nama : Muhammad Arif Rakhman Azizi
NIM : 21106050042
Judul Tugas Akhir : Rancang Bangun *Visual Studio Code Extension CodeVa* Untuk *Code Generator Java* Secara Otomatis Menggunakan *Large Language Models* (LLMs)

Sudah dapat diajukan kembali kepada Program Studi Informatika Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta sebagai salah satu syarat untuk memperoleh gelar Sarjana Strata Satu dalam Program Studi Informatika.

Dengan ini kami mengharap agar skripsi/tugas akhir Saudara dapat segera di-*munaqasyah*-kan. Atas perhatiannya saya ucapkan terima kasih.

*Wassalammu'alaikum wr. wb.*



Yogyakarta, 11 Agustus 2025
Pembimbing

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

# INTISARI

Dalam proses pengembangan perangkat lunak, tahapan implementasi *(Coding)* merupakan fase yang paling kompleks dan menyita waktu. Bahasa pemrograman Java sebagai salah satu bahasa yang populer memiliki kompleksitas kode yang seringkali menjadi tantangan para *developer*. Penulisan kode Java secara konvensional, sering kali tidak efisien, reptitif, bahkan rentang terhadap *human error*. Untuk mengatasi tantangan tersebut, dikembangkan sebuah *tool CodeVa,* sebuah ekstensi *Visual Studio Code* yang berfungsi sebagai *code generator Java* secara otomatis berbasis *Large Language Models.*

Metode pengembangan yang diterapkan adalah *Agile Extreme Programming* (XP). *Ekstensi CodeVa* memakai model *Open-source Qwen2.5 Instruct 7B* yang telah melalui proses *fine tuning* untuk tugas spesifik *Java Code Generation* menggunakan *dataset* yang relevan. *Fine tuning* menggunakan teknik *Parameter-Efficient-Fine-tuning* (PEFT) LoRA dengan bantuan *framework Unsloth.*

Hasil *fine tuning* mengalami peningkatan performa pada metrik *similarity-based matrics* terutama pada *CodeBLEU* dari **0.3204** menjadi **0.3521,** *ROUGE-1* dari **0.1736** menjadi **0.3705**, *ROUGE-2* dari **0.0251.** menjadi **0.2278**, serta *ROUGE-L* dari **0.0991** menjadi **0.3252**, serta *Pass@10* pada versi kulal dan chen dari **73.17%** menjadi **73.78%**. Pada pengujian *usability testing*, *ekstensi CodeVa* memiliki skor SUS 81.0 dan masuk dalam kategori *excellent, acceptable, marginal high*, serta *promoter*. Hal ini menunjukkan bahwa ekstensi *CodeVa* mampu meningkatkan produktivitas *developer* dalam penulisan kode Java secara cepat dan efisien.

**Kata Kunci:** Large Language Models, *Fine tuning*, Ekstensi, Unsloth.

# ABSTRACT

*The rapid advancement of technology has spurred a significant demand for software, yet the Coding phase, particularly in a complex language like Java, remains a time-consuming and error-prone process. To address this, a Visual Studio Code Extension named CodeVa was developed to automatically generate Java code using Large Language Models (LLMs).*

*Developed using the Agile Extreme Programming method ology, CodeVa integrates a fine-tuned Open-source model, Qwen2.5 Instruct 7B. This model was specifically trained for Java code generation using the Parameter-Efficient Fine-tuning (PEFT) technique, Low-Rank Adaptation (LoRA), and the Unsloth framework.*

*The fine-tuning process resulted in substantial improvements across various performance metrics, including an increase in the CodeBLEU score from 0.3204 to 0.3521 and significant gains in ROUGE scores ROUGE-1 from 0.1736 to 0.3705, ROUGE-2 from 0.0251 to 0.2278, and ROUGE-L from 0.0991 to 0.3252 and Pass@10 using kulal version and chen version from 73.17% to 73.78%. Usability testing of CodeVa yielded a System Usability Scale (SUS) score of 81.0, categorizing it as "excellent" and "acceptable." This indicates that CodeVa is a highly effective tool that can significantly enhance developer productivity by facilitating faster and more efficient Java Coding .*

***Keywords:*** *Large Language Models, Fine tuning, Ekstensi, Unsloth.*

# MOTTO

**"***Karena sesungguhnya sesudah kesulitan itu ada kemudahan (5). Sesungguhnya sesudah kesulitan itu ada kemudahan (6)." (**Q.S Al-Insyirah: 5-6**)*

*"Only the disciplined ones in life are free. If you are undisciplined, you are a slave to your moods and your passions." (**Eliud Kipchoge**)*

# HALAMAN PERSEMBAHAN

*Bismillahirrahmaanirrahiim*

*Allahumma sholli 'alaa sayyidinaa Muhammad wa 'alaa ali sayyidina Muhammad*

*Alhamdulillahirabbil'aalaamiin*, atas semua karunia yang telah diberikan oleh Allah SWT. Tugas akhir ini penulis dedikasikan dan persembahkan kepada:

*Kedua orang tua tercinta, Ibu Robihatin dan Bapak Nur Kholis S.Pd.I. yang telah membimbing, mendidik, memberi dukungan baik secara moril maupun materiil, selalu memberi semangat di kala sedih, serta yang selalu mendo'akan penulis agar selalu sukses dalam menjalani kehidupan. Tanpa mereka, penulis tidak akan sampai pada titik ini. Terima kasih yang tak terhingga dan tak bisa dihitung untuk segala pengorbanan kalian. Semoga Allah SWT membalas dengan balasan yang setimpa.*

*Kakakku tercinta Ilham Ulinnuha S.T. yang selalu memberi motivasi dan memberikan dukungan tentang arti kehidupan. Terima kasih sudah memberi cakrawala terhadap dunia yang sesungguhnya. Adikku tersayang Muhammad Zidny Al-Karim yang selalu menjadi teman bermain di kala jenuh.*

# KATA PENGANTAR

*Bismillahirrahmaanirrahiim. Alhamdulillahi rabbil 'aalaamiin* puji syukur penulis panjatkan kepada Allah SWT atas semua Karunia dan Rahmat-Nya, sehingga penulis dapat menyelesaikan tugas akhir yang berjudul "Rancang Bangun *Visual Studio Code Extension CodeVa* untuk *Code Generation Java* secara Otomatis Menggunakan *Large Language Models* (*LLMs*)" dengan lancar. Shalawat serta salam penulis haturkan kepada Baginda Nabi Agung Muhammad SAW yang telah membawa kita dari masa gelap menuju masa terang. Semoga kita semua termasuk golongan orang-orang yang mendapat syafa'atnya di akhir zaman ini.

Dalam proses penyelesaian tugas akhir ini butuh usaha dan perjuangan yang tidak mudah. Namun, semua itu tidak akan selesai tanpa dukungan, bimbingan, dan kerja sama dari berbagai pihak, baik secara langsung maupun tidak langsung. Oleh karena itu, penulis ingin menyampaikan ucapan terima kasih yang banyak kepada:

1. Prof. Noorhaidi Hasan, S.Ag., M.A., M.Phil., Ph.D. selaku Rektor UIN Sunan Kalijaga Yogyakarta periode 2024-2028.
2. Prof. Dr. Dra. Hj. Khurul Wardati, M.Si. selaku Dekan Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta periode 2024-2028.
3. Bapak Dr. Muhammad Mustakim, S.T., M.T. selaku Kepala Program Studi Informatika UIN Sunann Kalijaga Yogyakarta periode 2024-2028.
4. Ibu Dwi Otik Kurniawati, M.Eng. selaku Sekretaris Program Studi Informatika UIN Sunan Kalijaga Yogyakarta periode 2024-2028.
5. Bapak Nurochman, S.Kom., M.Kom. selaku Dosen Penasihat Akademik (DPA) penulis selama berkuliah di UIN Sunan Kalijaga Yogyakarta.

6. Bapak Dr. Agung Fatwanto, S.Si., M.Kom. selaku Dosen Pembimbing Skripsi (DPS) yang selalu sabar dalam membimbing dan menyusun tugas akhir.
7. Seluruh Dosen dan Staff Program Studi Informatika UIN Sunan Kalijaga Yogyakarta yang telah memberi ilmu yang luas kepada penulis selama perkuliahan.
8. Kedua orang tua penulis, Ibu Robihatin dan Bapak Nur Kholis S.Pd.I. yang telah merawat, membesarkan, mendidik, mendukung, mendo'akan, dan selalu memberikan semangat. Terima kasih atas semua yang telah kalian berikan kepada penulis sehingga bisa menempuh ke jenjang pendidikan yang tinggi.
9. Kakak penulis, Ilham Ulinnuha S.T. yang telah memberikan bantuan dan dukungan baik dari segi moril maupun materiil. Serta kepada adik saya Muhammad Zidny Al-Karim yang selalu menjadi teman ketika bermain ketika jenuh.
10. Keluarga besar penulis yang selalu memberikan dukungan dan do'a.
11. Teman-teman KKN Kolaborasi kelompok 117 angkatan 114 yang telah memberikan semangat kepada penulis.
12. Teman-teman Informatika angkatan 2021 yang telah menjadi teman seperjuangan pada masa perkuliahan.
13. Kepada diri saya sendiri, Muhammad Arif Rakhman Azizi. Terima kasih sudah berusaha maksimal dan tidak menyerah untuk menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa dalam penyusunan tugas akhir ini masih banyak kekurangan. Untuk itu, penulis mengharapkan kritik dan saran yang membangun agar bisa membuat tugas akhir ini menjadi lebih baik. Semoga tugas akhir ini dapat memberikan manfaat untuk pembaca semua.

Yogyakarta, 12 Agustus 2025

Penulis,

Muhammad Arif Rakhman Azizi

NIM. 21106050042

# DAFTAR ISI

# DAFTAR GAMBAR

# DAFTAR TABEL

# BAB I

# PENDAHULUAN

## 1.1 Latar Belakang

Pada era teknologi yang berkembang secara masif, kebutuhan akan perangkat lunak (*software*) telah menjadi elemen penting yang tidak dapat dipisahkan dari kehidupan manusia. Mulai dari aplikasi *mobile*, *desktop*, hingga layanan *cloud* telah memainkan peran penting yang mampu menunjang dalam berbagai bidang seperti bidang bisnis, pendidikan, perbankan, kesehatan, serta hiburan. Hal ini dapat dibuktikan melalui analisis bibliometrik oleh [1] yang menunjukkan peningkatan signifikan dalam jumlah publikasi akademik terkait pengembangan perangkat lunak dari tahun 2003—2022. Temuan tersebut menegaskan bahwa pengembangan perangkat lunak tidak hanya menjadi bagian dari inovasi teknologi, tapi juga strategi bisnis utama pada era digital global. Dengan peluang yang besar tersebut, membuka peluang bagi para pengembang (*developer*) untuk membuat perangkat lunak yang mampu memenuhi kebutuhan pengguna (*user* ).

Dalam dunia rekayasa perangkat lunak (*software engineering*), siklus pengembangan perangkat lunak terdiri dari beberapa tahapan yang dikenal dengan *system development life cycle*, yang terdiri dari: *planning* dan *analyisis* (perencanaan dan analisis), *design* (desain), *implementation* (konstruksi), *testing* (pengujian), *deployement* (perilisan), *maintenance* (pemeliharaan), dan *evaluation* (evaluasi) [2]. Setiap tahapan memiliki peran penting, tapi tahap *implementation* atau *Coding* merupakan tahapan yang paling kompleks dan menyita waktu bagi

developer. Dalam tahap implementation, tim *developer* akan membuat serangkaian sumber kode (source *code*) dari sepsifikasi desain yang telah ditentukan di tahap sebelumnya dengan mengikuti standard penulisan *code*, pemilihan bahasa Pemrograman, serta menerapkan *best practice* yang diterapkan secara sistematis untuk membangun komponen dan fungsionalitas *software* secara efisien dan terstruktur [3]. Pada tahap *implementation*, banyak tantangan yang akan dihadapi oleh tim developer, terutama dalam hal efisiensi penulisan *code*, kualitas *code*, produktivitas, serta keterbatasan waktu dalam menulis *code*. Selain itu, dalam proyek skala besar, penulisan *code* secara konvensional seringkali memakan waktu, membutuhkan tenaga ahli yang banyak, serta rentan terhadap kesalahan manusia [4]. Oleh karena itu, dibutuhkan pendekatan yang lebih efisien untuk mempercepat pengembangan perangkat lunak tanpa harus mengorbankan kualitas produk. Salah satu pendekatan yang saat ini berkembang pesat adalah pemanfaatan teknologi berbasis kecerdasan buatan (*artificial intelligence*/AI) dengan *Generative* AI, terutama melalui *Large Language Models*.

Perkembangan teknologi *Generative* AI telah berkembang pesat pada beberapa waktu terakhir ini, salah satu penerapan dari *Generative* AI adalah *Large Language Models* (LLMs). *Large Language Models* adalah model AI yang berbasis jaringan syaraf (*neural network*) yang mampu menghasilkan konten-konten baru secara otomatis seperti text, image, audio, video, bahkan *source code* berdasarkan instruksi berbahasa alami (*natural language*) [5]. Perkembangan dan kemajuan LLMs akhir-akhir ini telah mengubah perspektif dalam paradigma pengembangan *software*. Hal ini menjadi peluang besar dalam dunia *software* engineering dalam

memanfaatkan LLMs terutama kemampuannya untuk menghasilkan *source code* secara otomatis. LLMs yang telah dilatih menggunakan sejumlah *dataset* yang berisi sejumlah *source code* memiliki kemampuan dalam memahami pola-pola dalam *code* dan menghasilkan *code* secara otomatis berdasarkan konteks sesuai yang dideskripsikan dalam bahasa alami. Sejak munculnya model LLMs yaitu GPT oleh OpenAI pada Juli 2020, perkembangan LLMs terus mengalami peningkatan yang signifikan. Model-model LLMs lain mulai dari *Open-source* sampai closed-source telah muncul untuk mendukung pengembangan perangkat lunak dengan mamhami sintaksis dan semantaik dari suatu bahasa Pemrograman.

Model LLMs berbasis transformer seperti GPT, *Code*GeeX, serta *Code*T5 memiliki kemampuan mamahami konteks, menyarankan solusi, serta menghasilkan *code* secara efisien. Dengan memanfaatkan teknologi ini, akan membuat *developer* lebih fokus pada aspek-aspek kreatif lainnya dari proses pengembangan perangkat lunak karena tugas-tugas yang dilakukan secara repetitive seperti menulis *source code* dapat dibantu oleh model LLMs. Menurut penelitian [6] yang dilakukan oleh Zheng et.al pada tahun 2024, penggunaan model *Code*GeeX meningkatkan produktivitas dalam pengembangan perangkat lunak terutama pada tahap *Coding* dengan presentase 83.4%. Menurut laporan OpenAI, penggunaan LLMs dalam pengembangan perangkat lunak pada skala besar mampu mengurangi waktu sampai 30%.

Bahasa Pemrograman Java, yang diciptakan oleh James Gosling pada 1995 sebagai bahasa berbasis *general-purpose programming language* memungkinkan *developer* menulis *source code* yang bisa dieksekusi di berbagai *platforms* seperti

*web*, *mobile applications*, *desktop application*. Hal ini menjadikan bahasa Java sebagai bahasa yang populer di kalangan *developer* maupun akademisi untuk dijadikan penelitian di *software engineering*. Menurut survei dari *The TIOBE Organization* pada 2025, bahasa Java masuk dalam peringkat ke-4 sebagai bahasa populer. Data lain dari survei tahunan *Stack Overflow* pada tahun 2023 menyebut bahasa Java berada di peringkat teratas sebagai bahasa yang sering digunakan oleh pengembang perangkat lunak profesional. Java menawarkan *platform independensi*, keamanan, kemampuan *multithreading*, performa tinggi, serta skalabilitas adalah alasan utama kenapa Java begitu popular di kalangan *developer* [7]. Namun, dengan kompleksitas tersebut, justru menjadi tantangan bagi *developer* terutama dalam masalah penulisan *source code* yang memakan waktu lama sehingga menjadi hambatan yang cukup signifikan selama proses pengembangan.

Meskipun model LLMs mampu menghasilkan *code* secara otomatis, tetapi hasil *output code* tidak selalu sempurna, serta sering memerlukan validasi serta penyesuaian secara manual. Salah satu tantangan yang dihadapai adalah kurangnya pemahaman konteks terhadap kasus-kasus yang spesifik, terutama untuk bahasa pemrograman yang memiliki kompleksitas sintaksis seperti Java. Java, yang memiliki kompleksitas sintaksis dan memiliki aturan yang menurut sebagian besar *developer* sangat ketat, menjadi tantangan agar model LLMs mampu menghasilkan *code* yang relevan. Selain itu, penggunaan LLMs sebagai alat bantu dalam penulisan *code* juga sering dihadapi pada isu seperti masalah keamanan *code* program yang mungkin sensitif terhadap informasi dan *code* yang dihasilkan rentan terkena malware yang pada akhirnya merugikan perangkat lunak [8]. Namun,

dengan menerapkan strategi seperti memilih *dataset* yang tepat, memilih model LLMs yang tepat, serta strategi dalam menentukan *hyperparameter* yang tepat ketika *fine-tuning* model LLMs, akan menghasikan model LLMs yang memiliki kemampuan yang baik dalam menghasikan *code* yang relevan.

Pemilihan model LLMs yang tepat menjadi aspek yang krusial. Model LLMs yang digunakan harus memiliki kemampuan dalam memahami instruksi dalam bahasa alami dengan baik, menghasilkan kode yang sesuai konteks, serta mengatasi kompleksitas sintaksis dari bahasa Pemrograman yang dituju. Untuk memenuhi kebutuhan ini, penulis memilih model *Qwen2.5 7B Instruct* sebagai *foundational model* untuk dilakukan *fine tuning*. Model ini adalah model *Open-source* yang dikembangkan oleh Alibaba Group yang dirancang khusus untuk pemroses bahasa alami. Pemilihan model ini didasarkan atas: 1) Spesialisasi dalam tugas umum, termasuk *code generation*; 2) Performa Kompetitif; 3) Dukungan Bahasa Java; 4) Efisiensi dan Aksesibilitas; 5) *Open-source* dan Komunita aktif. Dari beberapa model LLMs berdasarkan pengujian yang dilakukan oleh [9], *Qwen2.5 7B instruct* memiliki performa yang paling tinggi. Hal ini dibuktikan dari data pada gambar 1.1 di bawah ini.

| Datasets | Gemma2-9B | Llama3.1-8B | Qwen2-7B | Qwen2.5-7B |
|---|---|---|---|---|
| *General Tasks* | | | | |
| MMLU-Pro | 52.1 | 48.3 | 44.1 | **56.3** |
| MMLU-redux | 72.8 | 67.2 | 67.3 | **75.4** |
| LiveBench 0831 | 30.6 | 26.7 | 29.2 | **35.9** |
| *Mathematics & Science Tasks* | | | | |
| GPQA | 32.8 | 32.8 | 34.3 | **36.4** |
| MATH | 44.3 | 51.9 | 52.9 | **75.5** |
| GSM8K | 76.7 | 84.5 | 85.7 | **91.6** |
| *Coding Tasks* | | | | |
| HumanEval | 68.9 | 72.6 | 79.9 | **84.8** |
| MBPP | 74.9 | 69.6 | 67.2 | **79.2** |
| MultiPL-E | 53.4 | 50.7 | 59.1 | **70.4** |
| LiveCodeBench | 18.9 | 8.3 | 23.9 | **28.7** |
| *Alignment Tasks* | | | | |
| IFEval | 70.1 | **75.9** | 54.7 | 71.2 |
| Arena-Hard | 41.6 | 27.8 | 25.0 | **52.0** |
| MTbench | 8.49 | 8.23 | 8.26 | **8.75** |

Gambar 1.1 Perbandingan Evaluasi Model LLMs [9]

Dari gambar 1.1 di atas, terapat empat model yang diuji dalam tugas berbagai tugas, salah satunya yaitu tugas *Coding*. Berdasarkan hasil pengujian, terlihat bahwa model *Qwen2.5 Instruct 7B* unggul dalam semua *test*, mengungguli model lain seperti *Gemma2 9B* dan *Llama3.1 8B* yang memiliki parameter yang lebih banyak. Model *Qwen2.5 Instruct 7B* dengan parameter yang lebih kecil dan unggul dalam tugas *Coding* menjadi indikator bahwa model tersebut lebih efisien dalam menghasilkan respon yang relevan. Oleh karena itulah, dalam tugas akhir ini menggunakan model *Qwen2.5 Instruct 7B* sebagai *foundational model* karena lebih efisien secara komputasi sehingga mudah diakses dengan sumber daya yang terbatas.

Untuk mengatasi permasalahan di atas dan untuk membantu *developer* dalam meningkatkan produktivitas serta menghemat sumber daya biaya dan waktu, diperlukan sebuah tool atau alat untuk menghasilkan *code* Java secara otomatis (*code generator*). Alat ini mampu memberikan manfaat secara signifikan: pertama,

secara signifikan menghemat waktu ketika menulis *code* aplikasi yang repetitif, terutama untuk *developer* pemula; kedua, *developer* menjadi lebih fokus pada logika bisnis dan fungsionalitas daripada menulis *code* secara repetitif. Selain itu, dengan memilih model LLMs yang tepat, akan menghasilkan *output* kode yang relavan, sesuai dengan instruksi yang diberikan oleh pengguna.

Tujuan dari tugas akhir ini adalah untuk merancang serta mengembangkan sebuah tool yang mengimplementasikan LLMs yang mampu menghasilkan (*generate*) *code* Java dalam bentuk *Visual Studio Code Extension*. Pemilihan tersebut didasari karena banyak *developer* Java yang menggunakan *Visual Studio Code* dalam mengembangkan aplikasi yang berbasis Java. Alat ini diharapkan mampu membantu *developer* dalam meningkatkan produktivitas serta mampu menghasilkan *code* yang berkualitas, memiliki keamanan tinggi, dan relevan. Kemudian, untuk jangka panjang, implementasi alat ini diharapkan dapat menjadi solusi inovatif dalam membantu dalam proses pengembangan perangkat lunak berbasis Java, sekaligus mendorong adopsi teknologi LLMs di dunia industri maupun akademik.



## 1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, maka masalah yang dikaji dalam tugas akhir ini adalah bagaimana merancang dan membangun *Visual Studio Code Extension* yang mampu menghasilkan *source code* Java berdasarkan instruksi bahasa alami.

## 1.3 Batasan Masalah

*Extension CodeVa* ini hanya digunakan dan diuji dalam ruang lingkup lokal, yaitu pengguna di negara Indonesia.

## 1.4 Tujuan

Dari rumusan masalah tersebut, maka tujuan yang ingin dicapai dalam tugas akhir ini adalah merancang dan membangun *Visual Studio Code Extension* yang mampu menghasilkan *source code* Java berdasarkan instruksi bahasa alami.

## 1.5 Manfaat

Hasil dari rancang bangun ini diharapkan dapat memberikan beberapa manfaat antara lain sebagai berikut:

1. Membantu pengembang (*developer*) Java dalam membangun aplikasi berbasis Java menjadi lebih efisien, cepat, dan tidak melakukan pekerjaan secara *repetitive*.
2. Meningkatkan produktivitas pengembang Java dengan mempercepat proses pembuatan *code* Java.
3. Berkontribusi pada pengembangan adopsi teknologi *Large Language Models* (LLMs) di dunia akademik maupun industri.

# BAB V

# KESIMPULAN DAN SARAN

## 5.1 Kesimpulan

Berdasarkan hasil perancangan dan pembangunan *Visual Studio Code Extension CodeVa*, dapat disimpulkan bahwa proses *fine-tuning* terhadap foundational mode *Qwen2.5 Instruct 7B* pada tugas *code generation* Java mampu memberikan peningkatan kinerja model. Pengujian *simliarity-based matrics* menggunakan metrik *Code*BLEU dan *ROUGE* menunjukkan bahwa *fine-tuning* mengalami peningkatan kinerja yang signifikan. Namun, pada pengujian *execution-based matrics*, performa pada pada Pass@1 dan Pass@5 mengalami penurunan kinerja, sementara pada Pass@10 menunjukkan peningkatan kinerja. Hal ini mengindikasikan bahwa model hasil *fine-tuning* memiliki peluang yang lebih besar untuk menghasilkan jawaban yang benar ketika diberikan lebih banyak kesempatan.

Dari sisi penerapan, *extension CodeVa* terbukti memberikan dampak positif terhadap peningkatan produktivitas pengembang dalam menulis kode Java. Hasil pengujian *usability* menggunakan instrumen SUS (*System Usability Scale*) menunjukkan skor 81.0 yang mendikasikan tingkat penerimaan yang tinggi. Dalam klasifikasi Bangor, Extension *CodeVa* berada dalam Acceptability Range pada kategori *Marginal High* hingga *Acceptable*, dengan *Grade Scale* pada rentang D hingga A, *Adjective Rating* pada rentang *Good* hingga *Best Imaginable*, serta masuk dalam kelas *Promoter* pada NPS.

## 5.2 Saran

Berdasarkan hasil di atas, berikut beberapa saran untuk pengembangan ekstensi ini di masa depan:

1. Meningkatkan akurasi model dengan melakukan *fine tuning* dengan *dataset* yang lebih besar dan beragam untuk meningkatkan performa pada semua metrik, terutama pada *execution-based metrics Pass@1* dan *Pass@5*.
2. Menambahkan fitur ekstensi untuk memahami kontek *code* di file yang tersedia.

# DAFTAR PUSTAKA


[1] P. N. Mata, J. M. Martins, and J. C. Ferreira, "New Software Product Development: Bibliometric Analysis," *J Knowl Econ*, Jun. 2024, doi: 10.1007/s13132-024-02095-5.

[2] Q. Yas, A. Alazzawi, and B. Rahmatullah, "A Comprehensive Review of Software Development Life Cycle methodologies: Pros, Cons, and Future Directions," *ijcsm*, pp. 173–190, Nov. 2023, doi: 10.52866/ijcsm.2023.04.04.014.

[3] Martin Otieno, David Odera, and Jairus Ekume Ounza, "Theory and practice in secure software development lifecycle: A comprehensive survey," *World J. Adv. Res. Rev.*, vol. 18, no. 3, pp. 053–078, Jun. 2023, doi: 10.30574/wjarr.2023.18.3.0944.

[4] A. Biswas and J. Singh, "SOFTWARE ENGINEERING CHALLENGES IN NEW MEDIA APPLICATIONS".

[5] S. Minaee *et al.*, "Large Language Models: A Survey," Mar. 23, 2025, *arXiv*: arXiv:2402.06196. doi: 10.48550/arXiv.2402.06196.

[6] Q. Zheng *et al.*, "CodeGeeX: A Pre-Trained Model for Code Generation with Multilingual Benchmarking on HumanEval-X," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, Long Beach CA USA: ACM, Aug. 2023, pp. 5673–5684. doi: 10.1145/3580305.3599790.

[7] D. D. Martinez, "A Review on Java Programming Language".

[8] Y. Wang, W. Wang, S. Joty, and S. C. H. Hoi, "CodeT5: Identifier-aware Unified Pre-trained Encoder-Decoder Models for Code Understanding and Generation," Sep. 02, 2021, *arXiv*: arXiv:2109.00859. doi: 10.48550/arXiv.2109.00859.

[9] Qwen *et al.*, "Qwen2.5 Technical Report," Jan. 03, 2025, *arXiv*: arXiv:2412.15115. doi: 10.48550/arXiv.2412.15115.

[10] J. Huang and K. C.-C. Chang, "Towards Reasoning in Large Language Models: A Survey," May 26, 2023, *arXiv*: arXiv:2212.10403. doi: 10.48550/arXiv.2212.10403.

[11] M. U. Hadi *et al.*, "Large Language Models: A Comprehensive Survey of its Applications, Challenges, Limitations, and Future Prospects," Nov. 16, 2023. doi: 10.36227/techrxiv.23589741.

[12] E. Arisoy, T. N. Sainath, B. Kingsbury, and B. Ramabhadran, "Deep Neural Network Language Models," in *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, B. Ramabhadran, S. Khudanpur, and E. Arisoy, Eds., Montréal, Canada: Association for Computational Linguistics, Jun. 2012, pp. 20–28. Accessed: Apr. 29, 2025. [Online]. Available: https://aclanthology.org/W12-2703/

[13] A. Vaswani *et al.*, "Attention Is All You Need," Aug. 02, 2023, *arXiv*: arXiv:1706.03762. doi: 10.48550/arXiv.1706.03762.

[14] N. Huynh and B. Lin, "Large Language Models for Code Generation: A Comprehensive Survey of Challenges, Techniques, Evaluation, and Applications," Apr. 02, 2025, *arXiv*: arXiv:2503.01245. doi: 10.48550/arXiv.2503.01245.

[15] H. Naveed *et al.*, "A Comprehensive Overview of Large Language Models," Apr. 09, 2024, *arXiv*: arXiv:2307.06435. Accessed: Sep. 05, 2024. [Online]. Available: http://arxiv.org/abs/2307.06435

[16] G. Nomad, "Unsloth.ai: Revolutionizing AI Model Fine-Tuning." [Online]. Available: https://medium.com/aimonks/unsloth-ai-revolutionizing-ai-model-fine-tuning-579c0afbddce

[17] E. J. Hu *et al.*, "LoRA: Low-Rank Adaptation of Large Language Models," Oct. 16, 2021, *arXiv*: arXiv:2106.09685. doi: 10.48550/arXiv.2106.09685.

[18] A. Shrivastava, I. Jaggi, N. Katoch, D. Gupta, and S. Gupta, "A Systematic Review on Extreme Programming," *J. Phys.: Conf. Ser.*, vol. 1969, no. 1, p. 012046, Jul. 2021, doi: 10.1088/1742-6596/1969/1/012046.

[19] A. Pambudi and W. Apriandari, "An Extreme Programming Approach for Instructor Performance Evaluation System Development," *INISTA*, vol. 5, no. 2, pp. 126–135, May 2023, doi: 10.20895/inista.v5i2.1050.

[20] H. Koç, A. M. Erdoğan, Y. Barjakly, and S. Peker, "UML Diagrams in Software Engineering Research: A Systematic Literature Review," in *The 7th International Management Information Systems Conference*, Basel Switzerland: MDPI, Mar. 2021, p. 13. doi: 10.3390/proceedings2021074013.

[21] Siska Narulita, Ahmad Nugroho, and M. Zakki Abdillah, "Diagram Unified Modelling Language (UML) untuk Perancangan Sistem Informasi Manajemen Penelitian dan Pengabdian Masyarakat (SIMLITABMAS)," *Bridge*, vol. 2, no. 3, pp. 244–256, Aug. 2024, doi: 10.62951/bridge.v2i3.174.

[22] E. Aquino, P. De Saqui-Sannes, and R. Vingerhoeds, “A Methodological Assistant for Use Case Diagrams:,” in *Proceedings of the 8th International Conference on Model-Driven Engineering and Software Development*, Valletta, Malta: SCITEPRESS - Science and Technology Publications, 2020, pp. 227–236. doi: 10.5220/0008938002270236.

[23] S. Sandfreni, M. B. Ulum, and A. H. Azizah, “ANALISIS PERANCANGAN SISTEM INFORMASI PUSAT STUDI PADA FAKULTAS ILMU KOMPUTER UNIVERSITAS ESA UNGGUL,” *Sebatik*, vol. 25, no. 2, pp. 345–356, Dec. 2021, doi: 10.46984/sebatik.v25i2.1587.

[24] I. K. Raharjana and A. Justitia, “PEMBUATAN MODEL SEQUENCE DIAGRAM DENGAN REVERSE ENGINEERING APLIKASI BASIS DATA PADA SMARTPHONE UNTUK MENJAGA KONSISTENSI DESAIN PERANGKAT LUNAK,” *JUTI*, pp. 133–142, Jul. 2015, doi: 10.12962/j24068535.v13i2.a482.

[25] S. Ramdany, “Penerapan UML Class Diagram dalam Perancangan Sistem Informasi Perpustakaan Berbasis Web,” *Journal of Industrial and Engineering System*, vol. 5, no. 1, Jul. 2024, doi: 10.31599/2e9afp31.

[26] M. Li *et al.*, “Superfiltering: Weak-to-Strong Data Filtering for Fast Instruction-Tuning,” Feb. 01, 2024, *arXiv*: arXiv:2402.00530. doi: 10.48550/arXiv.2402.00530.

[27] D. M. Anisuzzaman, J. G. Malins, P. A. Friedman, and Z. I. Attia, “Fine-Tuning Large Language Models for Specialized Use Cases,” *Mayo Clinic Proceedings: Digital Health*, vol. 3, no. 1, p. 100184, Mar. 2025, doi: 10.1016/j.mcpdig.2024.11.005.

[28] Z. Ma *et al.*, “LLaMoCo: Instruction Tuning of Large Language Models for Optimization Code Generation,” Mar. 05, 2024, *arXiv*: arXiv:2403.01131. doi: 10.48550/arXiv.2403.01131.

[29] C. Wang *et al.*, “RAG or Fine-tuning? A Comparative Study on LCMs-based Code Completion in Industry,” May 21, 2025, *arXiv*: arXiv:2505.15179. doi: 10.48550/arXiv.2505.15179.

[30] L. Xu, H. Xie, S.-Z. J. Qin, X. Tao, and F. L. Wang, “Parameter-Efficient Fine-Tuning Methods for Pretrained Language Models: A Critical Review and Assessment,” Dec. 19, 2023, *arXiv*: arXiv:2312.12148. doi: 10.48550/arXiv.2312.12148.

[31] L. Chen *et al.*, “A Survey on Evaluating Large Language Models in Code Generation Tasks,” Mar. 04, 2025, *arXiv*: arXiv:2408.16498. doi: 10.48550/arXiv.2408.16498.

[32] S. Ren *et al.*, “CodeBLEU: a Method for Automatic Evaluation of Code Synthesis,” Sep. 27, 2020, *arXiv*: arXiv:2009.10297. doi: 10.48550/arXiv.2009.10297.

[33] M. Chen *et al.*, “Evaluating Large Language Models Trained on Code,” Jul. 14, 2021, *arXiv*: arXiv:2107.03374. doi: 10.48550/arXiv.2107.03374.

[34] J. M. C. Bastien, “Usability testing: a review of some methodological and technical aspects of the method,” *International Journal of Medical Informatics*, vol. 79, no. 4, pp. e18–e23, Apr. 2010, doi: 10.1016/j.ijmedinf.2008.12.004.

[35] P. Sukmasetya, A. Setiawan, and E. R. Arumi, “PENGGUNAAN USABILITY TESTING SEBAGAI ALAT EVALUASI WEBSITE KRS ONLINE PADA PERGURUAN TINGGI,” vol. 9, no. 1, 2020.

[36] B. Lestari, P. I. Rifiani, and A. B. Gati, “The Use of the Usability Scale System as an Evaluation of the Kampung Heritage Kajoetangan Guide Ebook Application,” *European Journal of Business and Management Research*, vol. 6, no. 6, pp. 156–161, Dec. 2021, doi: 10.24018/ejbmr.2021.6.6.1113.

[37] S. I. Putri and K. Liu, “Assessing Ticket.com App Usability Through the System Usability Scale (SUS) Method,” *International Journal for Applied Information Management*, vol. 4, no. 1, pp. 30–40, Apr. 2024, doi: 10.47738/ijaim.v4i1.73.

[38] A. P. Sukma, R. Yusuf, and R. H. Dai, “ANALISIS PENGUKURAN USABILITY SISTEM INFORMASI MANAJEMEN BAZNAS (SIMBA) MENGGUNAKAN METODE SYSTEM USABILITY SCALE (SUS)”.