# SKRIPSI

# RANCANG BANGUN EKSTENSI VISUAL STUDIO CODE (VSCODE) TERINTEGRASI AI UNTUK GENERASI *UNIT TEST*

Sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer (S.Kom)


Oleh:

FANSURI FADEL FITRAH PRAKON

NIM: 21106050091





**PROGRAM STUDI INFORMATIKA**

**FAKULTAS SAINS DAN TEKNOLOGI**

**UNIVERSITAS ISLAM NEGERI SUNAN KALIJAGA**

**YOGYAKARTA**


**2025**

# PENGESAHAN TUGAS AKHIR

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-2520/Un.02/DST/PP.00.9/12/2025

Tugas Akhir dengan judul : Rancang Bangun Ekstensi Visual Studio Code (VSCode) Terintegrasi AI untuk Otomatisasi Unit Test

yang dipersiapkan dan disusun oleh:

| | |
|---|---|
| Nama | : FANSURI FADEL FITRAH PRAKON |
| Nomor Induk Mahasiswa | : 21106050091 |
| Telah diujikan pada | : Selasa, 16 Desember 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. Ir. Aulia Faqih Rifa'i, M.Kom.
SIGNED

Valid ID: 6945188e1b2e3

Penguji I

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

Valid ID: 6945057cf12e1

Penguji II

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

Valid ID: 69451697af106

Yogyakarta, 16 Desember 2025
UIN Sunan Kalijaga
Dekan Fakultas Sains dan Teknologi

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

Valid ID: 6948b8d26dc3a

# SURAT 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 Saudari:

| | | |
|---|---|---|
| Nama | : | Fansuri Fadel Fitrah Prakon |
| NIM | : | 21106050091 |
| Judul Skripsi | : | Rancang Bangun Ekstensi Visual Studio Code (VScode) Terintegrasi AI Untuk Otomatisasi Generasi Unit Test |

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

Dengan ini kami mengharap agar skripsi/tugas akhir Saudara dapat segera dimunaqasyahkan. Atas perhatiannya saya ucapkan terima kasih.

*Wassalammu'alaikum wr. wb.*

Yogyakarta, 10 Desember 2025
Pembimbing

Ir. Auliah Faqih Rifa'i, M.Kom.
NIP. 19860306 201101 1 009

## SURAT PERNYATAAN KEASLIAN

SURAT PERNYATAAN KEASLIAN

Yang bertanda tangan di bawah ini:

| | |
|---|---|
| Nama Lengkap | : Fansuri Fadel Fitrah Prakon |
| NIM | : 21106050091 |
| Tempat/Tgl lahir | : Lamakera, 09 November 2003 |
| Program Studi | : Informatika |

Dengan ini saya menyatakan dengan sesungguhnya bahwa skripsi yang saya susun dan ajukan merupakan hasil karya dan tulisan saya sendiri, bukan hasil penjiplakan atau pengambilan karya orang lain secara tidak sah. Seluruh sumber yang digunakan dalam penyusunan skripsi ini telah dicantumkan sesuai dengan kaidah penulisan ilmiah yang berlaku.

Apabila di kemudian hari terbukti bahwa skripsi ini merupakan hasil plagiarisme atau melanggar ketentuan akademik yang berlaku, maka saya bersedia menerima sanksi sesuai dengan peraturan dan ketentuan hukum yang berlaku.

Demikian surat pernyataan ini saya buat untuk dapat dipergunakan sebagaimana mestinya.



Senin, 08 Desember 2025

Yang membuat pernyataan,

mahasiswa


Fansuri Fadel Fitrah Prakon

NIM.21106050091

# MOTO

"Ilmu bukanlah yang dihafal, tetapi yang memberi manfaat."

– (Imam Syafi'i)

# INTISARI

Pengujian perangkat lunak merupakan tahapan krusial dalam pengembangan sistem untuk menjamin kualitas dan keandalan aplikasi. Namun, proses penulisan unit test secara manual masih memerlukan waktu dan usaha yang besar, sehingga sering kali menjadi beban bagi pengembang. Penelitian ini bertujuan untuk merancang dan membangun sebuah ekstensi Visual Studio Code bernama *Unittest Generator* yang terintegrasi dengan kecerdasan buatan berbasis Large Language Model (LLM) melalui Groq API dan OpenRouter API guna mengotomatisasi proses generasi unit test untuk berbagai bahasa pemrograman, yaitu Python, JavaScript, dan Java. Metode pengembangan yang digunakan adalah Extreme Programming (XP) yang dipadukan dengan pendekatan Research and Development (R&D) melalui dua iterasi pengembangan.

Hasil pengujian pada iterasi kedua menunjukkan bahwa seluruh kebutuhan fungsional sistem telah terpenuhi dengan baik. Evaluasi kualitas unit test yang dihasilkan menunjukkan nilai line coverage pada kisaran 85–92% dan mutation score sebesar 70–86%, yang mengindikasikan efektivitas unit test dalam mendeteksi kesalahan pada kode sumber. Selain itu, pengujian berbasis metrik BLEU dan ROUGE menunjukkan kesesuaian struktur dan substansi unit test terhadap referensi yang digunakan. Dari sisi performa, sistem mampu menghasilkan unit test dengan waktu respons rata-rata 4–8 detik, yang jauh lebih efisien dibandingkan penulisan unit test secara manual. Berdasarkan hasil tersebut, dapat disimpulkan bahwa ekstensi *Unittest Generator* mampu meningkatkan efisiensi, kualitas, dan produktivitas pengujian perangkat lunak, serta berpotensi menjadi solusi praktis dalam mendukung adopsi AI pada proses pengujian di lingkungan pengembangan modern.

**Kata kunci:** Generasi Unit Test, Ekstensi VSCode, Large Language Model, Otomatisasi Pengujian, Extreme Programming.

## *ABSTRACT*

*Software testing is a critical phase in the software development lifecycle to ensure system quality and reliability. However, manual unit test writing remains time-consuming and effort-intensive, often becoming a burden for developers. This study aims to design and develop a Visual Studio Code extension named Unittest Generator, integrated with artificial intelligence based on Large Language Models (LLMs) through Groq API and OpenRouter API, to automate unit test generation for multiple programming languages, namely Python, JavaScript, and Java. The development process applies the Extreme Programming (XP) methodology combined with a Research and Development (R&D) approach across two development iterations.*

*The testing results in the second iteration indicate that all functional requirements of the system have been successfully fulfilled. The quality evaluation of the generated unit tests shows line coverage ranging from 85–92% and mutation scores between 70–86%, indicating that the generated tests are effective in detecting faults within the source code. Furthermore, evaluation using BLEU and ROUGE metrics demonstrates a strong structural and semantic alignment between the generated unit tests and reference implementations. In terms of performance, the system achieves an average unit test generation time of 4–8 seconds, which is significantly faster than manual test creation. Based on these results, this research concludes that the Unittest Generator extension effectively improves testing efficiency, quality, and developer productivity, and serves as a practical solution to support AI adoption in modern software testing workflows.*



***Key words:*** *Unit Test Generation, VSCode Extension, Large Language Model, Test Automation, Extreme Programming*

# KATA PENGANTAR

Puji syukur selalu panjatkan ke kehadirat Allah SWT atas berkat dan rahmat-Nya, penulis dapat menyelesaikan penelitian ini yang berjudul "Rancang Bangun Ekstensi Visual Studio Code (VSCode) Terintegrasi AI untuk Otimatisasi *Unit test*" dengan lancar. Tak lupa sholawat serta salam senantiasa tercurahkan kepada baginda Nabi Muhammad SAW, yang telah memberikan syafaat teladan baik bagi umatnya dalam menuntut ilmu dan mengamalkannya.

Tentunya selama proses mengerjakan tugas akhir ini penulis tidak sendirian. Banyak dukungan dari berbagai pihak dalam bentuk semangat, bimbingan dan kontribusi selama proses penyusunan. Oleh karena itu penulis mengucapkan terima kasih yang sebesar-besarnya kepada:

Bapak Prof. Noorhaidi Hasan, S.Ag., M.A., M.Phil., Ph.D., selaku Rektor Universitas Islam Negeri Sunan Kalijaga Yogyakarta.

Ibu Prof. Dr. Dra. Hj. Khurul Wardati, M.Si., selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Sunan Kalijaga Yogyakarta.

Bapak Muhammad Mustakim, S.T., M.T., selaku Ketua Program Studi Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Sunan Kalijaga Yogyakarta.

Bapak Ir. Aulia Faqih Rifa'I, M.Kom. selaku Dosen Pembimbing Tugas Akhir yang telah bersedia meluangkan waktunya untuk memberikan arahan serta bimbingan selama proses mengerjakan tugas akhir.

Seluruh Dosen dan Karyawan Program Studi Informatika UIN Sunan Kalijaga yang telah kontribusinya selama proses menempuh pendidikan di bangku perkuliahan.

Kedua orang tua penulis Bpk. Muhammad Idris dan Ibu Saud Muhammad yang senantiasa memberikan dukungan, kasih sayang dan pembelajaran yang berharga sehingga penulis bisa berada di posisi saat ini.

Kaka penulis Virasuci Widyastuti yang telah menemani perjuangan penulis selama proses perkuliahan dari awal 2021 hingga saat ini.

Kaka penulis Nona Rosmiyati yang juga senantiasa memberikan dukungan, kasih sayang, serta pembelajaran berharga bagi menulis sehingga penulis senantiasa mampu menghadapi setiap kendala hadir.

Teman teman seperjuangan penulis, Hirzil, Rizal, Ihfan, Hanif Anggara, Rohman, Alega, Didit, teman-teman dari program studi informatika angkatan 2021 yang telah membersamai penulis dalam proses perkuliahan.

Penulis menyadari bahwa penyusunan tugas akhir ini masih banyak kekurangan dan keterbatasan. Oleh karena itu penulis mengharapkan semoga tugas akhir ini dapat memberikan manfaat dan kontribusi bagi pengembang pada bidang Informatika, serta menjadi referensi yang berguna bagi pembaca.

Yogyakarta, 10 Desember 2025

Penulis

Fansuri Fadel Fitrah Prakon

NIM.21106050091

# DAFTAR ISI

# DAFTAR GAMBAR

# DAFTAR TABEL

# BAB I
# PENDAHULUAN

## 1.1 Latar Belakang

Dalam beberapa tahun terakhir, kemajuan teknologi kecerdasan buatan (AI) telah mengubah proses pengembangan perangkat lunak, terutama pada tahap yang krusial yaitu pengujian untuk menjamin kualitas sistem. Melibatkan AI dalam pengujian terbukti ampuh dalam meningkatkan efisiensi, akurasi, dan cakupan pengujian secara signifikan. Namun, adopsi teknologi ini di industri nyatanya masih terbatas. Hal ini disebabkan adanya kendala teknis dan sulit dalam validasi hasil yang mengakibatkan pemanfaatan AI untuk generasi *test case*, analisis kode, dan otomatisasi cerdas menjadi tidak optimal [1]. Temuan tersebut selaras dengan servey yang dilakukan industri Tricentis (2024), di mana meskipun 60% profesional pengujian menganggap AI sebagai investasi strategis, namun masih banyak organisasi yang masih berada di tahap eksplorasi awal [2]. Alasannya adalah pada proses pembuatan skripnya yang membutuhkan waktu yang cukup lama. Seorang pengembang menghabiskan hampir 15,8% waktu mereka untuk menulis test, dan 42,72% untuk aktivitas terkait pengujian (termasuk debugging dan perbaikan) [3]. Hal ini menjadi beban manual bagi para pengembang untuk melakukan pengujian dengan *unit test*. Jadi, terlihat jelas bahwa ada perbedaan antara kemampuan besar AI dan cara penerapannya dalam otomatisasi pengujian, sebuah situasi yang memerlukan penelitian yang lebih lanjut.

Memperhatikan kesenjangan tersebut, penting untuk mencari strategi penerapan yang lebih efektif. Salah satu peluang besar terletak pada tren perkembangan ekosistem yang kini bergerak menuju cara kerja extension-in-IDE. Dalam pendekatan ini, teknologi baru seperti kecerdasan buatan bisa langsung terintegrasi ke dalam pekerjaan para pengembang melalui alat bantu ekstensi. Visual Studio Code (VSCode), sebagai salah satu IDE yang paling banyak digunakan dan bisa diatur sesuai kebutuhan, menjadi platform yang cocok untuk pendekatan ini. Dirancang dengan ekstensibilitas sebagai prinsip inti, hampir setiap

aspek VSCode dari antarmuka hingga pengalaman pengeditan dapat dikustomisasi dan ditingkatkan melalui API Ekstensi. Bahkan, banyak fitur inti yang dibangun sebagai ekstensi yang memanfaatkan API tersebut [4]. Perkembangan ini menjadi angin segar bagi pengembang untuk menghadirkan solusi pengujian berbasis AI secara lebih langsung dan kontekstual, berpotensi mengatasi kendala teknis dan adaptasi yang diidentifikasi dalam penelitian sebelumnya, sekaligus mewujudkan investasi strategis yang diharapkan oleh para pengembang profesional.

Oleh karena itu, integrasi teknologi AI secara langsung ke dalam IDE seperti Visual Studio Code (VSCode) tidak lagi terbatas pada konsep, tetapi telah didukung oleh kematangan platform AI khusus. Kemunculan platform AI yang menyediakan API seperti Groq API dan OpenRouter menjadi kunci dalam realisasi ide ini. Groq API menawarkan pemrosesan bahasa alami untuk pembuatan dan pengujian kode, sementara OpenRouter API menyediakan akses terpadu dan efisien ke berbagai model AI generatif yang mutakhir [5], [6]. Dengan memanfaatkan kemampuan kedua teknologi ini di dalam ekosistem ekstensi Visual Studio Code (VSCode), dapat diwujudkan sistem generasi *unit test* yang diotomatisasi. Diharapkan sistem ini tidak hanya dapat mengatasi kendala kecepatan dan validasi, tetapi juga mampu beradaptasi dengan konteks kode spesifik dan gaya pengkodingan masing-masing pengembang, sehingga menjawab langsung tantangan adopsi yang diungkapkan dalam penelitian sebelumnya.

Berdasarkan landasan tersebut, urgensi penelitian ini semakin jelas: menghadirkan solusi praktis yang langsung menjawab akar masalah rendahnya efisiensi dan adopsi otomatisasi pengujian. Penelitian ini bertujuan merancang dan membangun sebuah ekstensi Visual Studio Code yang memanfaatkan Groq dan OpenRouter API untuk mengotomatisasi generasi *unit test*. Dengan pendekatan "extension-in-IDE" yang didukung layanan AI khusus, solusi ini diharapkan dapat secara langsung mengatasi kendala teknis dan validasi yang diidentifikasi sebelumnya, sehingga berkontribusi dengan cara: 1) Mempercepat proses pengujian secara signifikan; 2) Mengurangi beban manual pengembang dalam menulis *test case*; dan 3) Meningkatkan cakupan serta kualitas pengujian perangkat lunak.

Dengan demikian, urgensi penelitian tidak hanya terletak pada inovasi integrasi teknologinya, tetapi lebih pada perannya sebagai jembatan yang menghubungkan potensi AI dari ranah akademis dan konseptual dengan alur kerja nyata di industri pengembangan perangkat lunak modern.

## 1.2 Rumusan Masalah

1. Bagaimana cara merancang dan mengembangun ekstensi Visual Studio Code (VSCode) yang mampu mengintegrasikan AI untuk menghasilkan *unit test* untuk berbagai bahasa pemrograman?
2. Bagaimana kualitas skrip pengujian yang dihasilkan oleh Ekstensi?

## 1.3 Tujuan Penelitian

Tujuan dari penelitian ini adalah sebagai berikut:

1. Merancang dan membangun ekstensi Visual Studio Code yang terintegrasi dengan Groq dan Openrouter API sebagai sarana otomatisasi dalam pembuatan *unit test*.
2. Mengevaluasi kualitas skrip *unit test* yang berhasil di generate oleh ekstensi.

## 1.4 Batasan Masalah

Agar penelitian ini lebih terarah dan fokus, maka ruang lingkup penelitian dibatasi pada beberapa aspek berikut:

1. Lingkup Pengembangan:

Penelitian ini hanya berfokus pada pengembangan ekstensi Visual Studio Code (VSCode) yang berfungsi untuk mengotomatisasi proses pembuatan *unit test*. Ekstensi ini tidak terintegrasi dengan IDE lain seperti PyCharm, Intellij IDEA, atau Eclipse.

2. Teknologi yang digunakan:

Kecerdasan buatan yang digunakan dalam penelitian ini terbatas pada Groq dan Openrouter API sebagai sumber pemrosesan LLM untuk menghasilkan kode

*unit test*. Penelitian ini tidak membahas atau membandingkan performa dengan API AI lainnya seperti OpenAI API, Gemini API, atau Mistral API.

3. Jenis pengujian:

Penelitian ini berfokus pada pembuatan dan pengujian *unit test* yang bersifat fungsi atau kelas. Pengujian integrasi (*Integration Testing*), *system testing,* atau *performance testing* tidak termasuk kedalam cakupan penelitian.

4. Evaluasi sistem:

Evaluasi dilakukan secara terbatas pada aspek fungsionalitas, kualitas hasil generasi *unit test*, perbandingan antara manual dan otomatis *testing* dan performa sistem tanpa melakukan analisis mendalam seperti pada aspek keamanan data dan lain-lain.

5. Lingkungan implementasi:

Kegiatan merancang dan membangun serta proses pengujian ekstensi dilakukan pada lingkungan pengembangan Visual Studio Code (VSCode) dengan dukungan sistem operasi Windows 11.

## 1.5 Manfaat Penelitian

Manfaat penelitian ini adalah membantu pengembang dalam meningkatkan efisiensi dan akurasi pembuatan *unit test*, serta memberikan kontribusi akademik berupa penerapan Groq dan OpenRouter API dalam konteks software testing berbasis IDE Extension.

# BAB V
# KESIMPULAN DAN SARAN

## 5.1 Kesimpulan

Berdasarkan hasil pengujian dan evaluasi yang dipaparkan pada bab sebelumnya, dapat disimpulkan bahwa ekstensi *Unittest Generator* yang dikembangkan telah berfungsi secara optimal baik dari sisi fungsionalitas, kualitas hasil unit test, maupun performa sistem. Seluruh kebutuhan fungsional pada iterasi pertama dan kedua berhasil dipenuhi, yang ditunjukkan melalui pengujian black-box dengan hasil *pass* pada seluruh skenario. Pada aspek kualitas unit test, hasil pengujian menunjukkan bahwa unit test yang dihasilkan oleh sistem memiliki tingkat line coverage yang tinggi, berada pada rentang 85–92%, serta mutation score antara 70–86%, yang mengindikasikan bahwa test yang dihasilkan efektif dalam mendeteksi perubahan perilaku kode. Evaluasi berbasis metrik BLEU dan ROUGE juga menunjukkan kesesuaian struktur dan substansi unit test terhadap referensi, sehingga memperkuat validitas hasil generasi kode. Dari sisi performa, waktu rata-rata generasi unit test berada pada kisaran 4–8 detik, yang secara signifikan lebih cepat dibandingkan penulisan unit test secara manual. Selain itu, perbandingan antara manual testing dan otomatis testing menunjukkan efisiensi waktu yang jauh lebih tinggi tanpa mengorbankan kualitas pengujian. Dengan demikian, hasil pengujian pada Bab IV membuktikan bahwa integrasi LLM melalui ekstensi Visual Studio Code tidak hanya layak secara teknis, tetapi juga efektif sebagai solusi praktis untuk meningkatkan efisiensi dan kualitas proses pengujian perangkat lunak.

## 5.2 Saran

Berdasarkan hasil penelitian dan pengembangan yang telah dilakukan, berikut adalah beberapa rekomendasi untuk pengembangan lebih lanjut:

1. Implementasi Mutation Testing Terintegrasi: Menambahkan mutation testing langsung dalam ekstensi untuk menilai efektivitas *unit test* secara

lebih akurat. Hal ini akan memberikan feedback yang lebih komprehensif kepada pengguna daripada hanya mengandalkan estimasi code *coverage*.

2. Pengembangan Test Repair System: Mengembangkan fitur otomatis untuk memperbaiki *unit test* yang gagal atau memiliki efektivitas rendah. Sistem ini dapat menganalisis kegagalan test dan merekomendasikan atau secara otomatis menghasilkan perbaikan berbasis LLM.
3. **Adaptive Prompt Engineering:** Mengembangkan model prompt yang dapat beradaptasi secara otomatis berdasarkan karakteristik kode pengguna, kompleksitas fungsi, dan pola pengembangan yang spesifik untuk meningkatkan relevansi dan kualitas test yang dihasilkan.
4. **Meningkatkan tampilan visual laporan kualitas**, misalnya melalui grafik atau heatmap *coverage* dan juga grafik untuk quality metric yang diterapkan.

Dengan menerapkan saran-saran tersebut, penelitian ini bisa diperluas menjadi solusi pengujian yang lebih kuat, menyeluruh, dan bermanfaat bagi para pengembang perangkat lunak di seluruh dunia.

# DAFTAR PUSTAKA


[1] K. Karhu, J. Kasurinen, and K. Smolander, “Expectations vs Reality -- A Secondary Study on AI Adoption in Software Testing,” pp. 1–26, 2025.

[2] Tricentis, “Results from the 2024 Techstrong Research and Tricentis survey,” 2024.

[3] E. Daka and G. Fraser, “A Survey on Unit Testing Practices and Problems”.

[4] V. S. Code, “Extension API.”

[5] Groq, “Groq Documentation.” https://console.groq.com/docs

[6] Openrouter, “Openrouter Docs.”

[7] Z. Yuan *et al.*, “Evaluating and Improving ChatGPT for Unit Test Generation,” vol. 1, no. July, pp. 1–24, 2024.

[8] T. Informatika, I. Teknologi, and S. Nopember, “PENGUJIAN PERANGKAT LUNAK DENGAN MENGGUNAKAN MODEL BEHAVIOUR UML Waskitho Wibisono , Fajar Baskoro,” pp. 43–50.

[9] A. N. Hasibuan and T. Dirgahayu, “Pengujian dengan Unit Testing dan Test case pada Proyek Pengembangan Modul Manajemen Pengguna.”

[10] N. Q. Dwiharani, Y. Wibisono, and Y. Wihardi, “Penerapan Large Language Models Dalam Pembaruan Artikel Biografi Wikipedia,” vol. 4, no. 2, pp. 804–813, 2025.

[11] E. Nijkamp *et al.*, “CODEGEN: AN OPEN LARGE LANGUAGE MODEL FOR CODE WITH MULTI-TURN PROGRAM SYNTHESIS,” pp. 1–25, 2023.

[12] S. Bhatia, T. Gandhi, D. Kumar, and P. Jalote, “Unit Test Generation using Generative AI : A Comparative Performance Analysis of Autogeneration Tools,” *Proc. - 2024 Int. Work. Large Lang. Model. Code, LLM4Code 2024*, pp. 54–61, 2024, doi: 10.1145/3643795.3648396.

[13] J. Jiang, F. Wang, J. Shen, S. Kim, and S. Kim, “A Survey on Large Language Models for Code Generation,” *J. ACM*, vol. 37, no. 4, pp. 1–70, 2018.

[14] S. Chodarev, “Visual augmentation of source code editors: A systematic mapping study,” vol. 49, pp. 46–59, 2018.

[15] E. Cal, T. Fulcini, R. Coppola, L. Laudadio, M. Torchiano, and P. Torino, “A Prototype VS Code Extension to Improve Web Accessible Development”.

[16] Microsoft, “Start developing extensions in Visual Studio,” 2024. https://learn.microsoft.com/en-us/visualstudio/extensibility/starting-to-develop-visual-studio-extensions?view=visualstudio&source=recommendations (accessed Nov. 16, 2025).

[17] D. Athanasiou, A. N. Member, J. V. Member, and I. C. Society, “Test Code Quality and Its Relation to Issue Handling Performance,” vol. 0, no. 0, pp. 1–25, 2000.

[18] S. Lukasczyk, F. Kroiß, and G. Fraser, “Automated Unit Test Generation for Python,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 12420 LNCS, pp. 9–24, 2020, doi: 10.1007/978-3-030-59762-7_2.

[19] F. Trautsch and J. Grabowski, “Are there any Unit Tests ? An Empirical Study on Unit Testing in Open Source Python Projects,” 2017.

[20] A. Tosun and M. Ahmed, “On the Effectiveness of Unit Tests in Test-driven Development”.

[21] S. S. Wibagso and I. Celesta, “Extreme Programming Approach in E-PANJO Design to Support Information Management at Nursing Home,” pp. 93–104.

[22] K. Fakhroutdinov, “UML 2.5 Diagrams Overview,” 2025. https://www.uml-diagrams.org/uml-25-diagrams.html

[23] M. R. V Chaudron, “An industrial case study on the use of UML in software maintenance and its perceived benefits and hurdles,” 2018.

[24] N. Sari and D. Cahyani, “Perancangan Sistem Informasi Monitoring Sertifikat Menggunakan Extreme Programming,” *J. Ilm. Comput. Sci.*, vol. 1, no. 1, pp. 1–6, 2022, doi: 10.58602/jics.v1i1.1.

[25] TESTINGMIND, “Future of QualityAssurance (Survey Report).”

[26] JetBrains, “Python Developers Survey 2023 Results,” 2023. https://lp.jetbrains.com/python-developers-survey-2023/

[27] S. Lukasczyk and G. Fraser, *Pynguin*, vol. 1, no. 1. Association for Computing Machinery, 2022. doi: 10.1145/3510454.3516829.

[28] C. Rupp, S. Queins, and die SOPHISTen, "Use-Case-Diagramm," *UML 2 Glas.*, no. September, pp. 239–262, 2012, doi: 10.3139/9783446431973.012.