TUGAS AKHIR

RANCANG BANGUN EKSTENSI VISUAL STUDIO CODE UNTUK OTOMATISASI PENGEMBALIAN FOKUS PADA AI-ASSISTED PROGRAMMING

DISUSUN OLEH :

AKHMAD IKHSAN SAYYID AL HAQ

NIM. 22106050045

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

PROGRAM STUDI INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI SUNAN KALIJAGA

YOGYAKARTA

2026

LEMBAR PERNYATAAN KEASLIAN

SURAT PERNYATAAN KEASLIAN

Yang bertanda tangan dibawah ini:

Nama	: Akhmad Ikhsan Sayyid Al Haq
NIM	: 22106050045
Program Studi	: Informatika
Fakultas	: Sains dan Teknologi

Menyatakan dengan sesungguhnya, bahwa skripsi saya yang berjudul RANCANG BANGUN EKSTENSI VISUAL STUDIO CODE UNTUK OTOMATISASI PENGEMBALIAN FOKUS PADA AI-ASSISTED PROGRAMMING merupakan hasil pekerjaan penulis sendiri sepanjang pengetahuan penulis, bukan duplikasi atau saduran dari karya orang lain kecuali bagian tertentu yang penulis ambil sebagai bahan acuan. Apabila terbukti pernyataan ini tidak benar, sepenuhnya menjadi tanggung jawab penulis.

Yogyakarta, 27 Agustus 2026



Akhmad Ikhsan Sayyid Al Haq
22106050045

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

LEMBAR 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-2400/Un.02/DST/PP.00.9/10/2026

Tugas Akhir dengan judul : RANCANG BANGUN EKSTENSI VISUAL STUDIO CODE UNTUK OTOMATISASI PENGEMBALIAN FOKUS PADA AI-ASSISTED PROGRAMMING

yang dipersiapkan dan disusun oleh:

Nama : AKHMAD IKHSAN SAYYID AL HAQ
Nomor Induk Mahasiswa : 22106050045
Telah diujikan pada : Senin, 31 Agustus 2026
Nilai ujian Tugas Akhir : A-

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

TIM UJIAN TUGAS AKHIR

Ketua Sidang

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

Valid ID: 6abdca5a205c7

Penguji I

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

Valid ID: 6abccdbf52b11

Penguji II

Nurochman, S.Kom., M.Kom
SIGNED

Valid ID: 6abebd48d89fd

Yogyakarta, 31 Agustus 2026
UIN Sunan Kalijaga
Dekan Fakultas Sains dan Teknologi

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

Valid ID: 6abe134d4f6c2

1/1 01/10/2026

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

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

Assalamu'alaikum wr. wb.

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

Nama	:	Akhmad Ikhsan Sayyid Al Haq
NIM	:	22106050045
Judul Skripsi	:	RANCANG BANGUN EKSTENSI VISUAL STUDIO CODE UNTUK OTOMATISASI PENGEMBALIAN FOKUS PADA AI-ASSISTED PROGRAMMING

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 tersebut di atas dapat segera dimunaqasyahkan. Atas perhatiannya kami ucapkan terima kasih.

Wassalamu'alaikum wr. wb.

Yogyakarta, 26 Agustus 2026
Pembimbing

Ir. Maria Ulfah Siregar, S.Kom., MIT., Ph.D.
NIP. 19780106 200212 2 001

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

LEMBAR PEDOMAN PENGGUNAAN TUGAS AKHIR

Tugas Akhir ini tidak dipublikasikan, tetapi tersedia di perpustakaan dalam lingkungan UIN Sunan Kalijaga, yang diperkenankan dipakai sebagai referensi kepustakaan, tetapi pengutipan harus seizin penyusun, dan harus menyebutkan sumbernya sesuai dengan kebiasaan ilmiah. Dokumen Tugas Akhir ini merupakan hak milik UIN Sunan Kalijaga Yogyakarta.

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

HALAMAN MOTTO

"Allah tidak membebani seseorang melainkan sesuai dengan kesanggupannya..."

— QS. Al-Baqarah: 286

"*Twenty years from now you will be more disappointed by the things you didn't do than by the ones you did. So throw off the bowlines. Sail away from the safe harbor. Catch the trade winds in your sails. Explore. Dream. Discover.*"

— Mark Twain

"Jika memang jawaban tak kau dapatkan, jangan tangisi, coba lalui teka-tekinya. Relakanlah, karena pada akhirnya, kau lebih kuat daripada yang kau kira"

— Kaela Kovalskia

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

HALAMAN PERSEMBAHAN

Alhamdulillahi Rabbil 'Alamin, puji syukur kepada Allah Subhanahu wa Ta'ala atas segala rahmat dan karunia-Nya. Dalam mengerjakan tugas akhir ini, hal utama bagi penulis adalah meniatkannya lillahi ta'ala. Dengan niat itu pula penulis dapat menyelesaikan tugas akhir ini dan mempersembahkannya khusus untuk:

1. Kedua Orang Tua Tercinta
2. Keluarga Besar
3. Dosen Pembimbing
4. Sahabat Seperjuangan
5. Diri Sendiri
6. Almamater Tercinta

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

KATA PENGANTAR

Alhamdulillahi Rabbil 'Alamin, puji dan syukur senantiasa penulis panjatkan ke hadirat Allah *Subhanahu Wa Ta'ala*, karena atas rahmat, ridha, dan karunia-Nya penulis dapat menyelesaikan penulisan Tugas Akhir yang berjudul "Rancang Bangun Ekstensi Visual Studio Code untuk Otomatisasi Pengembalian Fokus pada *AI-Assisted Programming*" dengan baik. Shalawat serta salam tak lupa dihaturkan kepada junjungan kita, Nabi Muhammad *Shallallahu 'Alaihi Wasalam*, yang selalu kita nantikan syafaatnya.

Penyusunan Tugas Akhir ini diajukan untuk memenuhi salah satu syarat dalam memperoleh gelar Sarjana pada Program Studi Informatika, Fakultas Sains dan Teknologi, Universitas Islam Negeri Sunan Kalijaga. Penulis menyadari bahwa proses penyelesaian skripsi ini tidak terlepas dari bimbingan, arahan, serta dukungan moril maupun materil dari berbagai pihak. Oleh karena itu, pada kesempatan ini penulis ingin menyampaikan ucapan terima kasih yang sebesar-besarnya kepada:

1. Bapak Prof. Noorhaidi, S.Ag., M.A., M.Phil., Ph.D., selaku Rektor Universitas Islam Negeri Sunan Kalijaga Yogyakarta.
2. Ibu Prof. Dr. Dra. Hj. Khurul Wardati, M.Si., selaku Dekan Fakultas Sains dan Teknologi, Universitas Islam Negeri Sunan Kalijaga Yogyakarta.
3. Bapak Dr. Muhammad Mustakim, S.T., M.T., selaku Ketua Program Studi Informatika, Fakultas Sains dan Teknologi, Universitas Islam Negeri Sunan Kalijaga Yogyakarta.
4. Bapak Agus Mulyanto, S.Si., M.Kom., selaku Dosen Penasihat Akademik yang telah memberikan arahan dan bimbingan kepada penulis selama masa perkuliahan.
5. Ibu Ir. Maria Ulfah Siregar, S.Kom., MIT., Ph.D., selaku Dosen Pembimbing yang telah dengan sabar meluangkan waktu, tenaga, dan pikirannya untuk memberikan bimbingan, masukan, serta arahan yang sangat berharga dari awal hingga selesainya skripsi ini.
6. Ibu Dr. Fitri Wulandari, S.Si., M.Kom. dan Bapak Nurrochman, S.Kom., M.Kom., selaku Dosen Penguji yang telah memberikan banyak evaluasi, kritik membangun, serta wawasan baru guna menyempurnakan penelitian ini.
7. Seluruh Bapak dan Ibu Dosen serta Karyawan Program Studi Informatika yang telah membekali penulis dengan ilmu pengetahuan, pengalaman, serta bantuan yang bermanfaat selama masa perkuliahan.

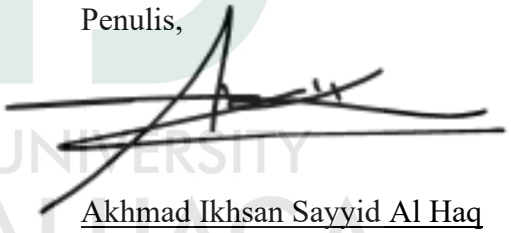
STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

8. Kedua orang tua tercinta, Bapak HK dan Ibu KS, dan saudara/i tercinta, Kakak AA dan adik IAM, yang tak pernah lelah memanjatkan doa, serta memberikan kasih sayang dan dukungan penuh baik secara moral maupun finansial.
9. Rekan-rekan seperjuangan, Mahasiswa Informatika Angkatan 2022 yang telah menjadi bagian dari perjalanan akademik dan penguji independen yang telah meluangkan waktunya untuk berpartisipasi dalam proses pengujian sistem, sehingga skripsi ini dapat tervalidasi dengan baik.
10. Rekan-rekan Gabut Bang dan Bukan Kelas Koding yang senantiasa menjadi teman belajar, bermain, bereksplorasi, maupun diskusi dan penulisan tugas akhir bagi penulis.
11. Seluruh pihak yang tidak dapat penulis sebutkan satu per satu, yang telah memberikan bantuan dan kelancaran selama masa penyusunan tugas akhir ini.

Penulis menyadari bahwa penulisan skripsi ini masih jauh dari kata sempurna karena keterbatasan pengetahuan dan pengalaman penulis. Oleh karena itu, penulis sangat mengharapkan kritik dan saran yang membangun dari seluruh pembaca. Akhir kata, penulis berharap semoga skripsi ini dapat memberikan manfaat, wawasan, serta kontribusi positif bagi pengembangan ilmu pengetahuan, khususnya di bidang rekayasa perangkat lunak.

Yogyakarta, 5 Oktober 2026

Penulis,



Akhmad Ikhsan Sayyid Al Haq

NIM. 22106050045

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

INTISARI

Penggunaan asisten pemrograman berbasis kecerdasan buatan seperti GitHub Copilot Chat memungkinkan pengembang beralih ke aplikasi lain selama menunggu respons AI selesai diproses. Kondisi tersebut mengharuskan pengembang melakukan perpindahan jendela secara manual untuk kembali ke Visual Studio Code setelah respons tersedia. Penelitian ini bertujuan merancang dan membangun ekstensi Visual Studio Code bernama copilot-focus yang mengotomatisasi pengembalian fokus jendela setelah aktivitas GitHub Copilot Chat selesai. Pengembangan ekstensi dilakukan menggunakan metode Extreme Programming (XP) melalui tiga iterasi. Sistem yang dibangun terdiri atas mekanisme pemantauan berkas log lokal secara berkala, penguraian (parsing) data log menggunakan regular expression, deteksi penyelesaian aktivitas Copilot, serta layanan pengembalian fokus yang disesuaikan dengan platform melalui Win32 API pada Windows dan AppleScript melalui osascript pada macOS. Fitur pendukung berupa notifikasi visual, notifikasi suara, dan konfigurasi pengaturan juga turut diimplementasikan. Pengujian dilakukan menggunakan metode Black-Box Testing berdasarkan empat skenario yang diturunkan dari acceptance criteria, baik secara internal maupun dengan melibatkan 13 penguji independen. Hasil pengujian menunjukkan tingkat keberhasilan fungsional sebesar 92%, di mana ekstensi berhasil mengembalikan fokus jendela tanpa intervensi manual pengguna. Evaluasi independen turut memvalidasi berjalannya mekanisme fallback pelindung (berupa kedipan ikon pada Taskbar Windows) ketika sistem operasi memblokir pemindahan fokus paksa, serta mengonfirmasi perlunya izin aksesibilitas manual pada macOS. Meskipun demikian, pada beberapa percobaan ditemukan kondisi di mana pemantauan log tidak langsung berjalan saat inisialisasi awal, sehingga diperlukan pemuatan ulang ekstensi atau pembukaan kembali Visual Studio Code.

Kata kunci: auto-focus switching, ekstensi Visual Studio Code, GitHub Copilot Chat, pemantauan log, Win32 API, AppleScript

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

ABSTRACT

The use of artificial intelligence-based programming assistants such as GitHub Copilot Chat allows developers to switch to other applications while waiting for the AI response to finish processing. This condition requires developers to manually switch windows to return to Visual Studio Code once the response is available. This research aims to design and build a Visual Studio Code extension named copilot-focus that automates the window focus return after the GitHub Copilot Chat activity is completed. The extension development was conducted using the Extreme Programming (XP) methodology through three iterations. The constructed system consists of a periodic local log file monitoring mechanism, log data parsing using regular expressions, Copilot activity completion detection, and a platform-tailored focus return service utilizing the Win32 API on Windows and AppleScript via osascript on macOS. Supporting features such as visual notifications, audio notifications, and configuration settings were also implemented. Testing was conducted using the Black-Box Testing method based on four scenarios derived from the acceptance criteria, performed both internally and by involving 13 independent testers. The test results demonstrated a functional success rate of 92%, wherein the extension successfully returned the window focus without manual user intervention. The independent evaluation also validated the execution of a protective fallback mechanism (in the form of a blinking Taskbar icon on Windows) when the operating system blocked forced focus switching, and confirmed the necessity of manual accessibility permissions on macOS. Nevertheless, in some trials, a condition was found where log monitoring did not run immediately upon initial initialization, thus requiring an extension reload or restarting Visual Studio Code.

Keywords: auto-focus switching, Visual Studio Code extension, GitHub Copilot Chat, log monitoring, Win32 API, AppleScript

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

DAFTAR ISI

LEMBAR PERNYATAAN KEASLIAN ii
LEMBAR PENGESAHAN TUGAS AKHIR iii
LEMBAR PERSETUJUAN TUGAS AKHIR iv
LEMBAR PEDOMAN PENGGUNAAN TUGAS AKHIR v
HALAMAN MOTTO vi
HALAMAN PERSEMBAHAN vii
KATA PENGANTAR viii
INTISARI x
ABSTRACT xi
DAFTAR ISI xii
DAFTAR TABEL xv
DAFTAR GAMBAR xvi
DAFTAR LAMPIRAN xvii
BAB I 1
PENDAHULUAN 1
1.1. Latar Belakang 1
1.2. Rumusan Masalah 3
1.3. Batasan Masalah 3
1.4. Tujuan Penelitian 4
1.5. Manfaat Penelitian 4
BAB II 6
KAJIAN PUSTAKA 6
2.1. Penelitian Terdahulu 6
2.1.1. Penelitian terkait AI-Assisted Programming dan Developer Experience 6

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

2.1.2. Penelitian terkait Interupsi, Context Switching, dan Task Resumption dalam Pemrograman 10
2.1.3. Ekstensi/Tool Sejenis 14
2.2. Konsep dan Teknologi Pendukung 18
2.2.1. Visual Studio Code 18
2.2.2. Visual Studio Code Extension 18
2.2.3. AI-Assisted Programming dan GitHub Copilot 19
2.2.4. Application Log dan Log Monitoring 20
2.2.5. Window Management dan Window Focus 22
2.2.6. Windows dan Win32 API 23
2.2.7. MacOS dan AppleScript 24
2.2.8. Extreme Programming (XP) 25
2.2.9. Black-Box Testing 27
BAB III 30
METODE PENGEMBANGAN SISTEM 30
3.1. Alat dan Bahan 30
3.1.1. Perangkat Keras 30
3.1.2. Perangkat Lunak 30
3.2. Metode Pengembangan Sistem 31
3.2.1. Tahap Perencanaan (Planning) 32
3.2.2. Tahap Perancangan (Design) 33
3.2.3. Tahap Implementasi (Coding) 33
3.2.4. Tahap Pengujian (Testing) 34
BAB IV 36
PERANCANGAN DAN IMPLEMENTASI SISTEM 36
4.1. Perencanaan 36
4.1.1. User Stories 36
4.1.2. Kebutuhan Fungsional 37
4.1.3. Kebutuhan Non-Fungsional 38
4.1.4. Acceptance Criteria 39
4.1.5. Release Plan 40

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

4.1.6. Iteration Plan 41
4.2. Perancangan Sistem 41
4.2.1. Use Case Diagram 41
4.2.2. Activity Diagram 42
4.2.3. Sequence Diagram 44
4.2.4. Diagram Struktur Modul dan Komponen 46
4.3. Implementasi Sistem 48
4.3.1. Implementasi Iterasi 1: Kode Inti 48
4.3.2. Implementasi Iterasi 2: Dukungan Lintas Platform 51
4.3.3. Implementasi Iterasi 3: Fitur Tambahan 55
4.3.4. Packaging 58
4.4. Pengujian 58
4.4.1. Metode dan Lingkungan Pengujian 58
4.4.2. Skenario Pengujian 59
4.4.3. Hasil Pengujian 60
4.4.4. Analisis Hasil Pengujian 62
BAB V 64
KESIMPULAN DAN SARAN 64
5.1. Kesimpulan 64
5.2. Saran 65
DAFTAR PUSTAKA 66
LAMPIRAN 70

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

DAFTAR TABEL

Tabel 2. 1 Ringkasan Penelitian terkait AI-Assisted Programming dan Developer Experience.... 8
Tabel 2. 2 Ringkasan Penelitian terkait Interupsi, Context Switching, dan Task Resumption dalam Pemrograman 12
Tabel 2. 3 Perbandingan Karakteristik Ekstensi dan Tool Sejenis 16

Tabel 4. 1 User Stories..... 36
Tabel 4. 2 Kebutuhan Fungsional 38
Tabel 4. 3 Kebutuhan Non-Fungsional..... 39
Tabel 4. 4 Acceptance Criteria..... 40
Tabel 4. 5 Iteration Plan..... 41
Tabel 4. 6 Skenario Pengujian Sistem 59
Tabel 4. 7 Rekapitulasi Hasil Pengujian Internal..... 61

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

DAFTAR GAMBAR

Gambar 3. 1 Diagram Siklus Metode Extreme Programming 31

Gambar 4. 1 Use Case Diagram 42
Gambar 4. 2 Activity Diagram 43
Gambar 4. 3 Sequence Diagram 45
Gambar 4. 4 Diagram Struktur Modul dan Komponen 47

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

DAFTAR LAMPIRAN

Lampiran 1. Hasil Pengujian Independen 70
Lampiran 2. Kode Sumber Ekstensi 72

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

BAB I

PENDAHULUAN

1.1. Latar Belakang

Tren penggunaan asisten pemrograman berbasis kecerdasan buatan (*AI-Assisted Programming)* bertumbuh pesat seiring dengan kemajuan *Large Language Models* (LLMs) seperti GitHub Copilot. Pemanfaatan asisten AI dalam penulisan kode terbukti memberikan peningkatan produktivitas yang sangat terukur. Dalam sebuah eksperimen terkontrol [1], kelompok pengembang yang memiliki akses ke *AI pair programmer* mampu menyelesaikan tugas pemrograman 55,8% lebih cepat dibandingkan dengan kelompok yang tidak menggunakannya. Selain mempercepat penulisan kode, asisten AI juga digunakan untuk membantu pengembang dalam berbagai aktivitas pemrograman termasuk eksplorasi kode dan pembelajaran sintaks atau struktur pemrograman baru, terutama oleh pengembang yang masih memiliki pengalaman terbatas [2], [3].

Meskipun AI terbukti mempercepat penyelesaian tugas secara keseluruhan, interaksi dengan asisten AI juga memperkenalkan aktivitas dan biaya waktu baru dalam alur kerja pengembang. Studi observasi oleh Mozannar menemukan bahwa pengembang menghabiskan lebih dari setengah waktu sesi mereka, yaitu 51,5% untuk berinteraksi dengan AI, dengan 34,3% dari total durasi kerja dihabiskan hanya untuk memverifikasi dan mengedit saran kode yang dihasilkan AI [4]. Temuan tersebut menunjukkan bahwa penggunaan AI tidak sepenuhnya menghilangkan aktivitas pengembang, tetapi mengubah bentuk pekerjaan yang dilakukan selama proses pengembangan. Selain aktivitas verifikasi dan penyuntingan, interaksi dengan AI juga melibatkan waktu tunggu ketika sistem memproses permintaan dan menghasilkan respons. *Response latency* dapat memengaruhi perilaku dan persepsi pengguna selama interaksi dengan sistem AI [5], [6].

Dalam konteks pengembangan perangkat lunak, waktu tunggu dan interaksi dengan AI menjadi relevan karena bantuan AI dapat memengaruhi kontinuitas *workflow* pengembang. Penelitian mengenai *proactive AI programming support* menunjukkan bahwa bantuan AI dapat meningkatkan efisiensi, tetapi pada kondisi tertentu juga menimbulkan *workflow disruption* [7]. Selain itu, waktu pemberian intervensi AI turut memengaruhi respons pengembang; intervensi

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

yang diberikan pada titik yang sesuai dengan alur kerja cenderung lebih mudah diterima dibandingkan intervensi yang muncul di tengah aktivitas [8]. Dengan demikian, persoalan interaksi AI tidak hanya berkaitan dengan kemampuan sistem menghasilkan respons, tetapi juga kapan respons tersebut tersedia dan bagaimana keberadaannya berinteraksi dengan *workflow* pengembang.

Gangguan terhadap *workflow* tersebut menjadi penting karena *interruption* dan *task switching* merupakan bagian dari aktivitas pengembangan perangkat lunak. Perpindahan tugas merupakan bagian dari aktivitas pengembangan perangkat lunak, dan tingkat gangguan yang ditimbulkannya dapat dipengaruhi oleh karakteristik tugas, waktu, jenis, dan sumber interupsi [9]. Ketika kembali ke tugas yang sebelumnya terinterupsi, pengembang juga tidak selalu dapat langsung melanjutkan pekerjaan. Analisis terhadap sekitar 10.000 sesi pemrograman dari 85 *programmer* profesional menunjukkan bahwa hanya sekitar 10% sesi memulai aktivitas pengkodean dalam waktu kurang dari satu menit, sementara sebagian besar sesi memerlukan aktivitas tambahan sebelum pengembang kembali melakukan pengeditan [10]. Hal tersebut menunjukkan bahwa biaya *interruption* tidak hanya terjadi ketika perhatian pengembang berpindah, tetapi juga ketika pengembang harus melakukan *task resumption* dan membangun kembali konteks tugas yang ditinggalkan.

Kondisi tersebut menjadi relevan dalam penggunaan *AI programming assistant*, khususnya ketika pengembang harus menunggu sistem menyelesaikan suatu permintaan. Periode tunggu dapat menjadi bagian dari dinamika *workflow* yang lebih luas, sementara *latency* dan *timing* respons AI dapat memengaruhi interaksi pengembang dengan sistem [5], [6]. Apabila perhatian pengembang telah beralih dari aktivitas utama ketika respons AI selesai, proses kembali ke tugas sebelumnya berpotensi memerlukan pemulihan konteks sebagaimana terjadi pada *task* yang mengalami *interruption* [9], [10]. Oleh karena itu, permasalahan yang relevan bukan hanya bagaimana mempercepat pemrosesan AI, tetapi juga bagaimana membantu pengembang mengenali penyelesaian proses AI dan kembali ke lingkungan pengembangan secara tepat waktu.

Dalam penggunaan *AI programming assistant* pada IDE, informasi mengenai penyelesaian respons dapat disampaikan melalui berbagai mekanisme antarmuka, seperti perubahan tampilan, notifikasi, atau indikator status. Namun, mekanisme tersebut pada dasarnya hanya memberikan informasi kepada pengguna dan tidak secara khusus menangani kondisi ketika pengembang sedang berada di luar jendela IDE. Berdasarkan penelusuran terhadap lima ekstensi/*tool* yang dipilih

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

sebagai pembanding, belum ditemukan mekanisme yang dapat diverifikasi untuk mengintegrasikan deteksi penyelesaian respons GitHub Copilot Chat dengan pengembalian fokus jendela Visual Studio Code secara otomatis tanpa intervensi manual pengguna. Kondisi tersebut menunjukkan adanya peluang untuk mengembangkan mekanisme yang tidak hanya memberi tahu pengembang bahwa proses AI telah selesai, tetapi juga membantu mengembalikan perhatian pengembang ke lingkungan pengembangan.

Berdasarkan permasalahan tersebut, penelitian ini mengusulkan rancang bangun sebuah ekstensi untuk Visual Studio Code yang dilengkapi dengan mekanisme *auto-focus switching*. Ekstensi dirancang untuk memantau penyelesaian aktivitas GitHub Copilot Chat melalui log yang dihasilkan oleh lingkungan Visual Studio Code. Ketika respons Copilot telah selesai, ekstensi memicu mekanisme untuk mengembalikan jendela Visual Studio Code ke *foreground*, sehingga pengguna dapat kembali ke jendela Visual Studio Code secara otomatis setelah aktivitas Copilot selesai.

1.2. Rumusan Masalah

a. Bagaimana merancang arsitektur perangkat lunak yang mampu mengintegrasikan ekstensi Visual Studio Code dengan fasilitas pengelolaan jendela pada sistem operasi yang didukung untuk menerapkan mekanisme *auto-focus switching*?
b. Bagaimana membangun mekanisme deteksi berbasis log yang mampu mengenali status penyelesaian aktivitas GitHub Copilot Chat secara berkelanjutan dan akurat?
c. Bagaimana mengimplementasikan dan menguji mekanisme *auto-focus switching* untuk mengembalikan fokus ke jendela Visual Studio Code setelah aktivitas GitHub Copilot Chat selesai?

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

1.3. Batasan Masalah

a. Pengembangan perangkat lunak ini dilakukan secara spesifik hanya untuk *Integrated Development Environment* (IDE) Visual Studio Code.
b. Deteksi status penyelesaian aktivitas AI dibatasi pada aktivitas GitHub Copilot Chat yang tercatat dalam berkas log di Visual Studio Code.

c. Mekanisme pengalihan fokus jendela (*window switching*) diimplementasikan menggunakan Win32 API pada Windows dan mekanisme otomasi aplikasi melalui AppleScript pada macOS.
d. Ekstensi ini hanya berfokus pada pengelolaan fokus antarmuka pada tingkat sistem operasi dan tidak melakukan manipulasi terhadap proses komputasi AI, model yang digunakan, maupun koneksi jaringan untuk mengurangi waktu pemrosesan atau *latency* respons GitHub Copilot.

1.4. Tujuan Penelitian

a. Menghasilkan artefak perangkat lunak berupa ekstensi Visual Studio Code yang mengintegrasikan mekanisme deteksi aktivitas GitHub Copilot dengan fasilitas pengelolaan fokus jendela pada sistem operasi yang didukung.
b. Membangun mekanisme deteksi berbasis log pada ekstensi Visual Studio Code untuk mengenali status penyelesaian aktivitas GitHub Copilot secara berkelanjutan dan akurat.
c. Mengimplementasikan dan menguji mekanisme *auto-focus switching* yang mampu mengembalikan fokus ke jendela Visual Studio Code setelah aktivitas GitHub Copilot selesai.

1.5. Manfaat Penelitian

Penelitian ini diharapkan dapat memberikan manfaat dalam aspek teoritis maupun praktis. Dari sisi teoritis, penelitian ini dapat menambah referensi dalam bidang rekayasa perangkat lunak dan Interaksi Manusia-Komputer (*Human-Computer Interaction*), terutama mengenai pengembangan ekstensi Visual Studio Code yang memanfaatkan informasi aktivitas asisten pemrograman berbasis AI untuk melakukan interaksi dengan antarmuka sistem operasi.

Dari sisi praktis, penelitian ini menghasilkan sebuah ekstensi yang dapat membantu pengembang dalam menggunakan GitHub Copilot melalui mekanisme pengembalian fokus ke Visual Studio Code setelah aktivitas AI selesai. Dengan demikian, pengguna dapat kembali ke lingkungan pengembangan tanpa harus melakukan perpindahan jendela secara manual. Selain itu, implementasi dan hasil pengujian penelitian ini dapat dimanfaatkan sebagai referensi bagi pengembang ekstensi maupun peneliti lain yang ingin mengembangkan mekanisme serupa, khususnya dalam menghubungkan pemantauan aktivitas aplikasi dengan pengelolaan jendela pada

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

sistem operasi Windows dan macOS. Penelitian ini juga dapat menjadi salah satu rujukan bagi penelitian berikutnya yang membahas pengembangan alat bantu untuk mendukung interaksi dan alur kerja dalam *AI-assisted programming*.

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

BAB V

KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan hasil perencanaan, perancangan, implementasi, dan pengujian ekstensi copilot-focus, diperoleh kesimpulan sebagai berikut.

Pertama, perancangan arsitektur perangkat lunak untuk mengintegrasikan ekstensi Visual Studio Code dengan mekanisme pengelolaan jendela sistem operasi berhasil diwujudkan melalui pemisahan antara FocusService dan FocusProvider. Pemilihan implementasi pengendalian fokus dilakukan berdasarkan platform melalui WindowsFocusProvider pada Windows dan MacFocusProvider pada macOS. Implementasi Windows memanfaatkan Win32 API melalui pustaka koffi, sedangkan implementasi macOS memanfaatkan AppleScript melalui utilitas osascript.

Kedua, mekanisme deteksi berbasis log berhasil dibangun melalui pemantauan perubahan berkas log GitHub Copilot Chat, penguraian (*parsing*) baris log, dan pendeteksian *event* penyelesaian aktivitas. Sistem melakukan pemantauan terhadap pertambahan ukuran berkas secara berkala, kemudian mengolah data baru menggunakan pola *regular expression*. *Event success* dari komponen yang berakhiran panel/editAgent digunakan untuk mengidentifikasi kondisi conversationFinished.

Ketiga, mekanisme *auto-focus switching* beserta fitur notifikasi tambahan berhasil diimplementasikan dan tervalidasi dengan tingkat keberhasilan fungsional mencapai 92% berdasarkan data pengujian independen. Hasil pengujian juga membuktikan bahwa kerangka kerja lintas platform berfungsi dengan baik, yang ditandai dengan berjalannya mekanisme *fallback* pelindung (kedipan *Taskbar*) secara otomatis pada Windows ketika pemindahan fokus paksa diblokir oleh sistem operasi.

Keempat, meskipun fungsi utama sistem dapat dijalankan, pengujian menemukan catatan pada proses inisialisasi pemantauan log. Pada beberapa percobaan, ekstensi tidak langsung mendeteksi berkas log GitHub Copilot Chat yang sesuai setelah Visual Studio Code dijalankan sehingga diperlukan pemuatan ulang ekstensi atau pembukaan kembali Visual Studio Code.

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

Temuan ini menjadi keterbatasan implementasi yang perlu diperhatikan pada pengembangan berikutnya.

5.2. Saran

Berdasarkan hasil pengujian dan keterbatasan yang ditemukan selama pengembangan, saran untuk pengembangan selanjutnya adalah sebagai berikut.

Pertama, penyempurnaan mekanisme inisialisasi *log*. Mekanisme pencarian dan penentuan berkas *log* GitHub Copilot Chat dapat dikembangkan agar lebih konsisten ketika Visual Studio Code baru dijalankan, sehingga pemantauan dapat dimulai tanpa memerlukan pemuatan ulang ekstensi atau pembukaan kembali aplikasi.

Kedua, peningkatan ketahanan terhadap perubahan format *log*. Mekanisme *parser* dan *completion detection* dapat dikembangkan agar lebih adaptif terhadap perubahan struktur berkas *log* yang dapat terjadi akibat pembaruan Visual Studio Code atau GitHub Copilot.

Ketiga, pengembangan evaluasi pada sisi pengguna. Penelitian selanjutnya dapat menguji dampak penggunaan *auto-focus switching* terhadap waktu kembali ke tugas, *task resumption*, pengalaman pengguna, atau produktivitas pengembang. Penelitian ini sendiri berfokus pada pengujian fungsional sistem dan belum mengukur aspek-aspek tersebut.

Keempat, pengembangan dukungan lintas platform dan asisten AI. Mekanisme yang telah dirancang dapat dikembangkan untuk mendukung platform atau asisten pemrograman berbasis AI lainnya setelah dilakukan penyesuaian terhadap mekanisme deteksi dan pengelolaan fokus yang digunakan.

Kelima, penanganan izin keamanan (*Security Permissions*) khusus pada lingkungan macOS. Mengingat mekanisme pemindahan jendela menggunakan AppleScript sangat bergantung pada izin aksesibilitas (*Accessibility*) dari sistem operasi, pengembangan selanjutnya disarankan untuk menambahkan fitur deteksi otomatis atau panduan penyelesaian masalah (*troubleshooting UI*) agar pengguna macOS menyadari perlunya memberikan izin tersebut sebelum ekstensi dapat beroperasi dengan normal.

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

DAFTAR PUSTAKA

[1] S. Peng, E. Kalliamvakou, P. Cihon, and M. Demirer, "The Impact of AI on Developer Productivity: Evidence from GitHub Copilot," Feb. 2023, doi: 10.48550/arXiv.2302.06590.

[2] S. Barke, M. B. James, and N. Polikarpova, "Grounded Copilot: How Programmers Interact with Code-Generating Models," *Proceedings of the ACM on Programming Languages*, vol. 7, no. OOPSLA1, Apr. 2023, doi: 10.1145/3586030.

[3] J. Prather *et al.*, "'It's Weird That it Knows What I Want': Usability and Interactions with Copilot for Novice Programmers," *ACM Transactions on Computer-Human Interaction*, vol. 31, no. 1, Nov. 2023, doi: 10.1145/3617367.

[4] H. Mozannar, A. Fourney, G. Bansal, and E. Horvitz, "Reading Between the Lines: Modeling User Behavior and Costs in AI-Assisted Programming," in *Conference on Human Factors in Computing Systems - Proceedings*, Association for Computing Machinery, May 2024. doi: 10.1145/3613904.3641936.

[5] J. Bogon, S. Hößl, C. Wolff, N. Henze, and D. Halbhuber, "Cognitive Integration of Delays: Anticipated System Delays Slow Down User Actions," in *Conference on Human Factors in Computing Systems - Proceedings* , Association for Computing Machinery, Apr. 2025. doi: 10.1145/3706598.3713475.

[6] F. Fang, Y. Tan, M. A. Messerschmidt, W. Yin, O. Nov, and A. Messerschmidt, "The Impact of Response Latency and Task Type on Human-LLM Interaction and Perception," *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems (CHI '26), April 13â•fi17, 2026, Barcelona, Spain*, vol. 1, p. 17, 2026, doi: 10.1145/3772318.

[7] K. Pu *et al.*, "Assistance or Disruption? Exploring and Evaluating the Design and Trade-offs of Proactive AI Programming Support," in *Conference on Human Factors in Computing Systems - Proceedings* , Association for Computing Machinery, Apr. 2025. doi: 10.1145/3706598.3713357.

[8] N. Kuo, A. Sergeyuk, V. Chen, and M. Izadi, "Developer Interaction Patterns with Proactive AI: A Five-Day Field Study," in *International Conference on Intelligent User Interfaces, Proceedings IUI*, Association for Computing Machinery, Mar. 2026, pp. 349–362. doi: 10.1145/3742413.3789148.

STATE ISLAMIC UNIVERSITY SUNAN KALIJAGA YOGYAKARTA

[9] Z. S. H. Abad, O. Karras, K. Schneider, K. Barker, and M. Bauer, “Task interruption in software development projects: What makes some interruptions more disruptive than others?,” in *ACM International Conference Proceeding Series*, Association for Computing Machinery, Jun. 2018. doi: 10.1145/3210459.3210471.

[10] C. Parnin and S. Rugaber, “Resumption Strategies for Interrupted Programming Tasks,” Jun. 2009. doi: 10.1109/ICPC.2009.5090030.

[11] Y. Ma, Y. Huang, and K. Leach, “Breaking the Flow: A Study of Interruptions During Software Engineering Activities,” in *Proceedings - International Conference on Software Engineering*, IEEE Computer Society, May 2024, pp. 2283–2294. doi: 10.1145/3597503.3639079.

[12] A. N. Meyer *et al.*, “Detecting Developers’ Task Switches and Types,” *IEEE Transactions on Software Engineering*, vol. 48, no. 1, pp. 225–240, Jan. 2022, doi: 10.1109/TSE.2020.2984086.

[13] A. Razzaq, J. Buckley, Q. Lai, T. Yu, and G. Botterweck, “A Systematic Literature Review on the Influence of Enhanced Developer Experience on Developers’ Productivity: Factors, Practices, and Recommendations,” Oct. 07, 2024, *Association for Computing Machinery*. doi: 10.1145/3687299.

[14] Microsoft, “Why did we build Visual Studio Code?” Accessed: Aug. 27, 2026. [Online]. Available: https://code.visualstudio.com/docs/editor/whyvscode

[15] Microsoft, “Extension Anatomy.” Accessed: Aug. 27, 2026. [Online]. Available: https://code.visualstudio.com/api/get-started/extension-anatomy

[16] Microsoft, “Your First Extension.” Accessed: Aug. 27, 2026. [Online]. Available: https://code.visualstudio.com/api/get-started/your-first-extension

[17] Microsoft, “Visual Studio Code Extension API,” VS Code Documentation. Accessed: Sep. 25, 2026. [Online]. Available: https://code.visualstudio.com/api

[18] X. Hou *et al.*, “Large Language Models for Software Engineering: A Systematic Literature Review,” *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 8, Dec. 2024, doi: 10.1145/3695988.

[19] GitHub, “Getting started with GitHub Copilot,” GitHub Docs. Accessed: Sep. 30, 2026. [Online]. Available: https://docs.github.com/en/copilot/using-github-copilot/getting-started-with-github-copilot

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

[20] GitHub, "Using GitHub Copilot Chat in your IDE," GitHub Docs. Accessed: Sep. 30, 2026. [Online]. Available: https://docs.github.com/en/copilot/github-copilot-chat/using-github-copilot-chat-in-your-ide

[21] S. He, P. He, Z. Chen, T. Yang, Y. Su, and M. R. Lyu, "A Survey on Automated Log Analysis for Reliability Engineering," Jan. 31, 2022, *Association for Computing Machinery*. doi: 10.1145/3460345.

[22] J. Zhu *et al.*, "Tools and Benchmarks for Automated Log Parsing," in *Proceedings - 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice, ICSE-SEIP 2019*, Institute of Electrical and Electronics Engineers Inc., May 2019, pp. 121–130. doi: 10.1109/ICSE-SEIP.2019.00021.

[23] D. El-Masri, F. Petrillo, Y. G. Guéhéneuc, A. Hamou-Lhadj, and A. Bouziane, "A systematic literature review on automated log abstraction techniques," *Inf. Softw. Technol.*, vol. 122, Jun. 2020, doi: 10.1016/j.infsof.2020.106276.

[24] A. Dix, J. E. Finlay, G. D. Abowd, and R. Beale, "Human-Computer Interaction (3rd Edition)," Dec. 2003.

[25] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating System Concepts Tenth Edition*. 2018.

[26] Microsoft, "Windows API Index," Microsoft Learn. Accessed: Sep. 30, 2026. [Online]. Available: https://learn.microsoft.com/en-us/windows/win32/apiindex/windows-api-list

[27] Microsoft, "SetForegroundWindow function," Microsoft Learn. Accessed: Sep. 30, 2026. [Online]. Available: https://learn.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-setforegroundwindow

[28] Apple Inc., "AppleScript Language Guide," Apple Developer Documentation. Accessed: Sep. 25, 2026. [Online]. Available: https://developer.apple.com/library/archive/documentation/AppleScript/Conceptual/AppleScriptLangGuide/

[29] Apple Inc., "osascript," Mac OS X Developer Tools Manual. Accessed: Sep. 25, 2026. [Online]. Available: https://developer.apple.com/library/archive/documentation/Darwin/Reference/ManPages/man1/osascript.1.html

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

[30] Apple Inc., “Allow apps to control other apps on Mac,” macOS User Guide. Accessed: Sep. 25, 2026. [Online]. Available: https://support.apple.com/guide/mac-help/control-access-to-automation-on-mac-mh32356/mac

[31] Kent. Beck, *Extreme programming eXplained : embrace change*. Addison-Wesley, 2000.

[32] R. S. Pressman and B. R. Maxim, “Software Engineering A Practitioner’s Approach Ninth Edition,” 2020.

[33] I. Sommerville, “Software Engineering Tenth Edition,” 2016.

[34] G. J. Myers, C. Sandler, and T. Badgett, *The Art of Software Testing*. 2012.

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA