Autoethnography in the Pursuit of AI Engineering Upskilling

Ji-Jian Chin1, a) and Yvonne Hwei-Syn Kam1, 2, b)

1School of Engineering, Computing and Mathematics, University of Plymouth, PL48AA, United Kingdom   
2Centre for Cybersecurity and Quantum Computing, COE for Advanced Cloud, Faculty of Artificial Intelligence and Engineering, Multimedia University, 63100 Cyberjaya, Selangor, Malaysia

*b) Corresponding author: hskam@mmu.edu.my*

*a) ji-jian.chin@plymouth.ac.uk*

**Abstract.** Agentic AI tools such as code assistants and workflow agents are reshaping software development but pose steep learning curves, tool immaturity, and limited access to role-specific training. While platforms like GitHub Copilot Labs, Replit Ghostwriter, and DeepLearning.ai offer new upskilling paths, users often face quota limits, integration issues, and unclear workflows. This autoethnographic study documents a 100-hour, zero-cost upskilling effort by a computer science academic with minimal AI experence. Through hands-on use of tools like Replit, Cursor, and VS Code with Copilot, AI-powered marking assistants were developed. The study provides practical insights into tool capabilities, learning barriers, and workflow adaptation under real-world constraints. By capturing lived experiences, it offers a realistic roadmap for educators, developers, and time-constrained professionals entering Agentic AI development. This work fills a gap in current literature by showing what is realistically achievable within limited time and budget, highlighting both the potential and the practical limits of AI-assisted coding.

# introduction

As large-language models (LLMs) continue to evolve, the capacity for web applications to accommodate fuzzy logic and human-like inference improve, resulting in dynamic systems that can solve a myriad of new problems previously not feasible for direct logic systems to handle. Since the emergence of Agentic AI in late 2024, the field has rapidly evolved, introducing new tools and paradigms for application development.

The earliest reference to agentic workflows we found was Andrew Ng’s presentation at the Snowflake Developer Summit [1]. Since then, these systems have evolved into fully automated code generation services like Replit[2], no-code UI services like N8N[3] and more recently Google Agentic Development Kit[4], which automates code generation for applications. These represent potential disruptions to traditional software engineering.

As educators in higher education, we are compelled to understand these changes to better prepare students for the evolving workforce. The only way to adapt effectively is through hands-on exploration.

The motivation behind this study was to conduct a hands-on exploration of Agentic AI from a learner’s perspective, with the goal of better understanding the process, challenges, and pitfalls involved. Between March to May 2025 surrounding the Easter holiday period in the UK, 100 hours of the first author’s time was devoted to exploring resources from online tutorials, videos and Deeplearning.ai[5] to see what can be achieved at the end of the study. The intended beneficiaries of this study are:

1. Academics with limited time due to teaching, research, and administrative responsibilities, who need an efficient upskilling path.
2. Software companies looking to train new graduate hires/ employees to maximise the utility of these agents for software solution development.

The central question we explore is: *Given a finite amount of time, what can a software developer with no AI background realistically accomplish?* We believe this to be a timely exploration of the subject in the face of the massive disruption Agentic AI will bring to the software development world. To reflect real-world scenarios, we introduce three constraints to our study:

1. Requirements-driven approach: As academics ourselves, we focus on a practical use case: developing AI-powered marking assistants for academic use. This also leverages on our access to subject matter experts who can help verify the accuracy of results. The prototypes themselves are discussed in a separate paper; here, we focus on what can be achieved in 100 hours using a requirements-driven approach.
2. Limited time: a 100-hour time limit is imposed, reflecting the nature that most working professionals will have limited time to devote to upskilling endeavours. In our case, this was accelerated due to the 3-week Easter holiday where there were no classes, though other duties (research, administration, outreach) were still ongoing. A detailed breakdown of the time allocation within these 3 weeks is given in the Methodology section.
3. Zero cost: a strict financial constraint was imposed on this study. Only free-tier development tools were used, and LLM inference was run locally via Ollama[6]. This reflects the financial constraints of start-ups, small-medium enterprises and hobbyists who have limited funding.

Transitioning to an agentic AI paradigm proved more complex than expected. It is not as simple as prompting an AI for a solution and receiving perfect output. We detail these challenges and limitations throughout the paper as our findings. On a positive note, the study led to the successful development of five multimodal AI-powered marking assistant systems, built using the time, resources, and skills acquired during the upskilling process. These systems were tested, with results to be detailed in a separate paper.

The rest of our paper is structured as follows: in Section 2 we provide some background and related work on the value and history of autoethnographical work in computing, as well as agentic AI advancements. Section 3 outlines our methodology and the parameters of our study: a) Author skill background, b) technological stack required, c) activities and time measurements and d) online training resources. Section 4 will elaborate on tool limitations, challenges and insights. Section 5 presents a discussion and conclusion.

# Background and related work

We begin with surveying some previous work on autoethnographies. “Autoethnography is a research method that intertwines personal experiences with cultural and social contexts to understand the relationship between the self and others”[7]. While used mainly in human and social science studies, there are some previous works related to computing. For example, Cunningham and Jones provided their findings in the form of an autoethnography with regards to practice and education[8], Rapp used it in Human-Computer Interaction (HCI) [9], and Rucero examined smartphone abstinence through an autoethnographic lens [10]. As part of a preparatory measure in the development of agentic AI curriculum, we found autoethnography an appropriate method to explore the field and establish a pathway for colleagues and students.

Agentic AI systems, characterised by their autonomy, real-time adaptability and ability to solve multi-step problems, are attracting academic interest. They can automate tasks like literature reviews [11], or enhance performance compared to single-shot AI models [12]. In education, such systems can analyze student work, provide context-aware feedback, and adapt assessment strategies [13]. As these advancements become increasingly accessible, the education domain, which is rich with diverse and practical use case has emerged as one of our primary areas of development focus.

Another prime area of focus is software development. Existing software full stack development encompasses both front-end and back-end tasks, utilising advanced frameworks like React, Vue and Tailwind coupled with back-end technologies like Node.js, Python and Java Spring, and databases (SQL and/or NoSQL) [14]. Recent years have also seen advancements in full stack trending towards cloud-native environments with DevOps practices, utilising Docker, Kubernetes and CI/CD workflows [15],[16]. Combining both approaches to software development is non-trivial and involves a breadth of new technologies.

# Methodology

## Author skill background

The main author has background as a software developer before transitioning into cybersecurity, cloud technologies and more recently game technologies. Between the years 2008 and 2015, the main author has developed and deployed several web applications for Multimedia University using HTML/CSS/Javascript frontend, PHP MySQL backend on WAMP stack. The author had only limited exposure to Python, Docker and Git, mostly in a teaching context, so some upskilling was required.

## Technology Stack

The narrative in this section is from the lead author’s perspective as the software developer: Given our project goal, which was building LLM-powered marking assistants, I had to understand a full-stack AI development environment. The first step was to familiarize myself with the full stack environment for web applications. A decision was made to transfer to Python for better integration with AI coding tools, which resulted in additional learning time.

I describe my learning process in more detail below:

1. Frontend – regular Web5 HTML/CSS deployment, streamlit and bootstrap were considered but too time-consuming to go in-depth.
2. Backend – I attempted to use PostgreSQL for a more industrial-ready approach. This was difficult due to lack of administrator rights on the development machine, causing PostgreSQL to not install locally. A decision was made to use a Docker approach, so I have two containers running dpage/pgadmin(a phpMyAdmin equivalent frontend) and postgres on Docker Desktop. Fortunately, PostgreSQL’s management is quite similar to MariaDB, so with pgadmin’s interface the set up did not take too long. However, deploying on Docker required additional set up time to understand the connectivity within the Docker network. For subsequent projects, sqlite3 was used for simplicity, with migration to PostgreSQL an open possibility.
3. Docker Desktop – containers were built for the sqlite3 systems. An attempt was made to package the PostgreSQL set up, but unfortunately the YAML file configuration for the network was not possible to resolve, not even with AI assistance. We found the AI limitation here, as ChatGPT, Claude and Gemini all devolved into repeating the same answers, but the connection was still unable to be established. I decided to stop pursuing this course of action after 4 hours in.
4. GIT – while the main author had some experience with GIT but due to long lapses in practice, this skill was not retained. Fortunately, it was easy to relearn, and thus all projects have their GIT repositories set up, shown in table in Appendix.

Description of AI-specific components:

1. Agents – an LLM empowered by connected tools to take automated actions. Practically these are code class functions within Python that have access to utilities in a separate python file.
2. Tools – file readers, scrapers, visualisers etc. Within these tools, certain familiarity with the dependencies to be deployed is required:
   1. For Multimodal LLMs: PyMUPDF, OpenCV, RapidOCR, torch
   2. For Youtube API and frame analyzer: googleapiclient, youtube\_transcript\_api, nltk, threading, faiss, numpy, whisper, traceback, yt\_dlp
   3. For file readers (Java, Jupyter and PDF): werkzeug utilities, shutil, os utilities, uuid, rapid\_ocr, faiss, open\_clip, functools, pillow, torch
3. Retrieval Augmented Generation (RAG) using Vector Stores that index documents too large for LLM context: I tried set ups with chromadb and OpenAI-CLIP using models from nomic-embed-text and all-MiniLM-L6-v2 sentence transformer.
4. No-code workflow automation tool such as N8N: While building RAG-powered chatbots were easy, I found integrating the workflow with web pages in Flask to be challenging as the interfaces differ (fuzzy conversation input for LLMs vs precise control mechanics such as checkboxes and dropdown select boxes on Flask apps). I continue exploring better integration approaches.

## Activities

The lead author’s 100-hour study time frame was conducted from 15 March 2025 to 7 May 2025. The tracked time was rounded to the nearest half-hour. These activities are presented in Table 1.

## Training Resources

The primary learning resources used in this study were sourced from Deeplearning.ai[5] and YouTube tutorials. These were especially valuable during the early phases of the experiment to guide the direction of the implementations to come. A total of 20 courses were undertaken within the 100-hour time frame, but due to space limitations we are unable to list them here. We will provide them in the extended journal version. While not all courses were directly aligned with the project's objectives, most helped delineate the scope of the study. Even those that were only tangentially relevant at the time remain useful for future reference and enhancement.

**TABLE 1**. Time-Account of 100 hours spent during March-May 2025

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date** | **Time start** | **Time end** | **Hours** | **Topic** |
| 15-Mar | 4pm | 6pm | 2 | Deeplearning Introduction Courses |
| 17-Mar | 3pm | 4pm | 1 | Deeplearning CrewAI course |
| 18-Mar | 11.30am | 1pm | 1.5 | CrewAI Install |
| 21-Mar | 9am | 11am | 2 | CrewAI Ollama Setup |
| 21-Mar | 2pm | 3pm | 1 | CrewAI troubleshooting |
| 22-Mar | 1pm | 4pm | 3 | Deeplearning courses |
| 24-Mar | 11.30am | 4pm | 5.5 | RAG Pipeline Development (learning and trial) |
| 24-Mar | 7.30pm | 9pm | 1.5 | RAG Pipeline Development (continued) |
| 31-Mar | 11.30am | 3pm | 4.5 | RAG on Youtube analysis. |
| 01-Apr | 5pm | 6pm | 1 | RAG Pipeline Multimodal - built youtube analyser using Youtube API |
| 08-Apr | 10am | 3pm | 5 | Google Scholar Compiler using AI (partial success) |
| 08-Apr | 4pm | 7pm | 3 | Manus test run to compile academic CV (Unsuccessful) |
| 09-Apr | 9am | 11am | 2 | IPYNB RAG and Parser |
| 09-Apr | 12pm | 4pm | 5 | CrewAI with Frontend Integration (Unsuccessful) |
| 09-Apr | 8pm | 9pm | 1 | Langchain Chat on Deeplearning |
| 09-Apr | 9pm | 10pm | 1 | Lit review on Crew-AI, Langchain, LlamaIndex |
| 10-Apr | 1pm | 3pm | 2 | AutoGPT Exploration, Solving CrewAI Telemetry Issues |
| 10-Apr | 8pm | 10pm | 2 | CrewAI Filereader Tools troubleshooting |
| 11-Apr | 12pm | 2.30pm | 2.5 | Replit trial |
| 12-Apr | 11am | 12pm | 3 | Replit Ollama replication |
| 12-Apr | 2pm | 4pm | 2 | Replit Course on Deeplearning |
| 12-Apr | 4pm | 5pm | 1 | Explore Langflow, Datastax, FlowWise# |
| 12-Apr | 5pm | 8pm | 3 | n8n setup |
| 12-Apr | 8pm | 10pm | 2 | n8n trial with Ollama |
| 13-Apr | 10am | 11am | 1 | Roo Dev Ollama Setup |
| 14-Apr | 12pm | 1pm | 1 | Github Autopilot with VS Code Setup |
| 14-Apr | 4pm | 5pm | 1 | Postgres readup |
| 15-Apr | 11am | 12pm | 1 | Postgres Installation (localhost unsuccessful due to admin rights, docker successful) |
| 15-Apr | 2pm | 3pm | 1 | Postgres Integration with IPYNB Analyser |
| 15-Apr | 3pm | 4pm | 1 | Validation test with Lauren |
| 15-Apr | 5pm | 6pm | 1 | Flask documentation readup |
| 15-Apr | 8pm | 9pm | 1 | Explore GoogleAppsheet and MS PowerApps |
| 15-Apr | 9pm | 10pm | 1 | Langchain Academy Introduction course |
| 16-Apr | 3pm | 4pm | 1 | Set up github repositories for work done |
| 16-Apr | 4pm | 8pm | 4 | Docker build for ipynb analyser (Unsuccessful due to lack of understanding on docker networks) |
| 22-Apr | 1pm | 2pm | 1 | Agentic AI Code Structure study |
| 23-Apr | 11.30am | 1pm | 1.5 | Multimodal RAG set up |
| 23-Apr | 1pm | 2pm | 1 | Trial Google Agent Development Kit |
| 23-Apr | 3pm | 5.30pm | 2.5 | Built Youtube Analyzer from ytDLP and Whisper with frame analysis |
| 24-Apr | 12pm | 2pm | 2 | Architecture Planning for Java Project Analyser |
| 28-Apr | 12.30pm | 2pm | 1.5 | Built PDF Analyser with SQLITE |
| 28-Apr | 7pm | 8pm | 1 | COMP3000 System Design using Claude |
| 29-Apr | 10am | 11am | 1 | Complete GITHUB and Trello Analyser |
| 30 April | 9am | 11am | 2 | Packaging and deploying the PDF Analyser for tests. |
| 30 April | 4.30pm | 6pm | 1.5 | Modifications to Youtube Frame Analyser for github readiness |
| 30 April | 6pm | 8pm | 2 | Touch up Java Evaluator for testing |
| 1 May | 9.30am | 10am | 0.5 | Java Evaluator Tests |
| 1 May | 7pm | 10pm | 3 | Dockerise Java Analyser |
| 2 May | 11am | 2pm | 3 | Streamline pdfanalyser and Java eval packages, removing bloatware |
| 2 May | 4pm | 7pm | 3 | Finalise commits, generate class, sequence diagrams and push to GIT. |
| Total Hours | | | 100 |  |

# Tool limitations, Challenges and insights

This study explored several approaches to AI tool development. Table 2 groups tools by function, highlighting their limitations, technical hurdles, and key usage insights.

**TABLE 2**. Summary of AI assistants trialed and key findings

|  |  |  |  |
| --- | --- | --- | --- |
| **Tool Group** | **Tool Limitations** | **Technical Hurdles** | **Insights and Recommendations** |
| **Replit[2], Lovable [17] and Manus [18]**  Note: Lovable was not tested due to time constraints. | Free-tier restrictions limit utilization. Manus has limited credits, and once expired, access is removed. Replit allows 10 checkpoints; additional use incurs cost. Manus returned unrelated publications from Google Scholar using similar author names. | Elaborate system descriptions are needed; one-shot agents require comprehensive upfront design. Replit exhausted checkpoints quickly during debugging. Manus misidentified publications and required 10–15 minutes per failed attempt, using up all free credits. | Use these services for generating initial scaffolds and prototypes, then switch to other tools (listed below) for refinement. Transitioning from cloud to local environments requires manual setup. Quota restrictions hinder adjustments and iterations. Having to repeat edits missed by the AI often requires consuming additional credits. |
| **Cursor, Windsurf and Claude Desktop IDE** | Cursor’s free tier offers 2-weeks Pro trial, followed by 50 premium and 200 small model queries. Claude Desktop was more limited, so development was mainly on Cursor. Windsurf could not be tested due to admin rights limitations. | Cursor overrode programmer planned modular file structure and merged code into a monolithic app.py. Proposed edits were displayed only in the sidebar chat, not in the main code window, reducing developer oversight. Overwritten context memory sometimes ignored earlier prompts. | These IDEs integrate AI copilots in a familiar environment with folder hierarchy and terminals. They support LLM switching and Model Context Protocol (MCP), with indexed workspaces that provide agent context and reduce namespace conflicts, which is ideal for agentic workflows. However, developers must remain vigilant as AI code edits can bypass structural intent. |
| **VS Code with Github Copilot[19] and Roo Cline [20]** | GitHub Copilot allows 2,000 completions and 50 queries/month; the 50-query limit applies even when using local Ollama models. Roo Cline was tested with Codellama and Qwen:Coder, but response times (~1 min) were much slower than GPT-4o on Github copilot (~5 sec). | Slow response from Roo Cline limited its usefulness. Copilot was used to complete most assistants until the quota expired, after which development continued on Cursor. | VS Code with Copilot is likely the most cost-effective, offering free-tier access and compatibility with local LLMs. Recent updates enabled workspace indexing, putting it on par with Cursor but with more generous allocations. |
| **Notepad++ with Human Relay** | Same limitations as general LLMs: services switch to simpler models or stop responding when credits expire. Context window constraints limit code generation to small snippets. Free to use but time-consuming. | Tedious manual copy/paste between browser and IDE. Frequent namespace mismatches require refactoring. Development time greatly increased compared to automated tools. | Common fallback for developers unfamiliar with advanced tools. Initial phase of study used this method via ChatGPT, Claude, Gemini, Deepseek. Later transitioned to GitHub Copilot and Cursor to save time. Can yield better-designed applications but is not time-efficient. Depends on developer expertise. Lacks full AI-driven orchestration. |

# discussion

Most AI upskilling frameworks fall into structured MOOCs, corporate bootcamps, or informal self-study. Programs like DeepLearning.ai and Google’s AI Essentials offer curated paths but assume ample time and funding. In contrast, this study explores a time-limited, zero-cost upskilling path under academic constraints. Unlike prior work focused on structured learning programs, our autoethnographic approach captures what someone can accomplish in a short, practical, real-world timeframe and hands-on friction points. This makes it a practical, experience-based roadmap for educators and developers working within tight real-world limitations.

In our study, tools like Cursor and GitHub Copilot enabled rapid prototyping but struggled in full stack scenarios requiring modular architecture and long-term maintainability. For instance, Cursor replaced our modular layout with a single monolithic code, leading to structural and naming inconsistencies. While effective for quick iteration or scripting, “vibe coding” (an improvisational style of coding where developers rely on AI tools via natural language prompts to generate, refactor, and debug code with minimal upfront design) falls short for systems with structured and specific requirements unless balanced by traditional engineering practices. Our findings highlight the need for hybrid workflows that combine AI-driven spontaneity with disciplined design to ensure reliability and control in real-world development.

The main advantages of this study include the realistic constraints under which the study was conducted (limited time, zero cost), the diversity of tools explored, in a realistic application of developing five working prototypes within 100 hours. The autoethnographic approach also offers unique insights into lived experiences and constraints during rapid AI assisted development. However, there are some limitations. Some of the tools were still unstable or in early stages of development, and subject to rapid change given the fast-evolving nature of the domain. Furthermore, as common in autoethnographies, the results relied on subjective qualitative assessments from a single participant’s experience.

# CONCLUSION

This paper presented the outcomes of a time-limited, self-directed upskilling initiative focused on AI-enhanced software development using agentic workflows. Operating under zero-cost constraints, the study explored current tools, design practices, and deployment strategies, highlighting what can realistically be achieved within the short timeframe. The study is intended to provide readers with similar motivations an account of the salient concepts, tooling best practices and limitations and lessons learnt.

By the conclusion of the 100-hour experiment, five AI-powered marking assistants were developed, each at varying stages of completeness. The links to these marking assistants’ GIT repository is listed in the Appendix Table 3. All prototypes achieved core functional capability and reached a Minimum Viable Product (MVP) level before being pushed to GitHub. The AI-driven marking assistants developed during this study demonstrated clear utility in academic settings, such as in automating feedback, summarizing code, and supporting plagiarism checks. We intend to present the detailed methodology of developing these tools and the results in a separate publication focused on marking automation.

The upskilling path described in this study can be transferable to other applications such as academic CV parsing, literature review bots, course support chatbots, and RAG-based analyzers for documentation, all achievable with minimal cost. The findings can also inform the design of modular Agentic AI training curricula for wider educational use. Future work could include team-based studies involving participants with varied experience levels, the introduction of quantitative benchmarks such as time-to-deploy and agent accuracy, and the development of modular Agentic AI training materials to support curriculum design in higher education. It is the authors’ hope this study will assist in more effective educational approaches to tackle the disruption of AI in software development.

# References

1. Snowflake Developers, *Andrew Ng On AI Agentic Workflows And Their Potential For Driving AI Progress*, (Jun. 11, 2024). Accessed: May 06, 2025. [Online Video]. Available: https://www.youtube.com/watch?v=q1XFm21I-VQ
2. replit, *Pricing*. Accessed: Apr. 30, 2025. [Online]. Available: https://replit.com/pricing
3. n8n, *n8n Plans and Pricing - n8n.io*. Accessed: May 06, 2025. [Online]. Available: https://n8n.io/pricing/
4. Google, *Agent Development Kit*. Accessed: May 06, 2025. [Online]. Available: https://google.github.io/adk-docs
5. DeepLearning.AI, *Start or Advance Your Career in AI*. Accessed: May 07, 2025. [Online]. Available: https://www.deeplearning.ai/
6. Ollama, *Ollama*. Accessed: May 07, 2025. [Online]. Available: https://ollama.com
7. ScienceDirect, *Autoethnography - an overview | ScienceDirect Topics*. Accessed: May 06, 2025. [Online]. Available: https://www.sciencedirect.com/topics/social-sciences/autoethnography
8. S.J. Cunningham, and M. Jones, “Autoethnography: a tool for practice and education,” in *Proceedings of the 6th ACM SIGCHI New Zealand Chapter’s International Conference on Computer-Human Interaction: Making CHI Natural (CHINZ ’05)*, (Association for Computing Machinery, New York, NY, USA, 2005), pp. 1–8.
9. A. Rapp, “Autoethnography in Human-Computer Interaction: Theory and Practice,” in *New Directions in Third Wave Human-Computer Interaction: Volume 2 - Methodologies*, M. Filimowicz and V. Tzankova, Eds. (Springer International Publishing, Cham, 2018), pp. 25–42.
10. A. Lucero, “Living Without a Mobile Phone: An Autoethnography,” in *Proceedings of the 2018 Designing Interactive Systems Conference (DIS ’18)*, (Association for Computing Machinery, New York, NY, USA, 2018), pp. 765–776.
11. M. Gridach, J. Nanavati, K.Z.E. Abidine, L. Mendes, and C. Mack, “Agentic AI for Scientific Discovery: A Survey of Progress, Challenges, and Future Directions,” (Mar. 12, 2025), arXiv:2503.08979.
12. P. Sawant, “Agentic AI: A Quantitative Analysis of Performance and Applications,” (Feb. 20, 2025), Preprints: 2025021647.
13. E. Kasneci et al., “ChatGPT for good? On opportunities and challenges of large language models for education,” Learning and Individual Differences **103**, 102274 (2023).
14. Bitcot, *Top 10 Full Stack Development Trends for 2025 and Beyond*. Accessed: May 07, 2025. [Online]. Available: https://www.bitcot.com/top-full-stack-development-trends-and-industry-demands/
15. Nucamp, *What are the Fundamentals of Full Stack Web and Mobile Development?* Accessed: May 07, 2025. [Online]. Available: https://www.nucamp.co/blog/coding-bootcamp-full-stack-web-and-mobile-development-what-are-the-fundamentals-of-full-stack-web-and-mobile-development
16. H.C. Mann Jyoti, *Silicon Valley’s next act: bringing “vibe coding” to the world*, Business Insider. Accessed: May 07, 2025. [Online]. Available: https://www.businessinsider.com/vibe-coding-ai-silicon-valley-andrej-karpathy-2025-2
17. Lovable, *Lovable*. Accessed: May 07, 2025. [Online]. Available: https://lovable.dev/pricing
18. Openmanus, *Openmanus - AI Agent that bridges minds and actions*. Accessed: May 07, 2025. [Online]. Available: https://manus-ai.com/pricing
19. Visual Studio Code, *VS Code Agent Mode Just Changed Everything*, (Apr. 09, 2025). Accessed: May 07, 2025. [Online Video]. Available: https://www.youtube.com/watch?v=dutyOc\_cAEU
20. cline, *cline: Autonomous coding agent right in your IDE, capable of creating/editing files, executing commands, using the browser, and more with your permission every step of the way.* Accessed: May 07, 2025. [Online]. Available: https://github.com/cline/cline

# Appendix

**TABLE 3**. GITHUB Links to the AI assistants.

|  |  |
| --- | --- |
| **Assistant** | **Link** |
| Youtube Analyzer with API | https://github.com/kenobicjj/youtubeanalystAPI |
| Youtube Analyzer with frame dissect | https://github.com/kenobicjj/youtubeanalysisv1 |
| PDF Analyzer | https://github.com/kenobicjj/pdfanalyst |
| Java Project Analyzer | https://github.com/kenobicjj/java\_eval |
| Jupyter Notebook Analyzer | https://github.com/kenobicjj/courseworkreview |