Teaching Vibecoding: An Open Curriculum for AI-Assisted Software Development

Andy Luu
andyhluu23@gmail.com
AlgoArena
Virginia Beach, VA, USA

Eli Young
eliyoung4now@gmail.com
AlgoArena
Virginia Beach, VA, USA

Michael Wehar*
mwehar@haverford.edu
AlgoArena
Virginia Beach, VA, USA

Abstract

Vibecoding, building software by stating intent to an AI in natural language instead of hand-coding it, is spreading quickly through industry and entrepreneurship. In the 2025 Stack Overflow Developer Survey, 84% of developers reported using or planning to use AI coding tools, and Y Combinator, a startup accelerator, disclosed in early 2025 that a quarter of its most recent startup batch participants had codebases that were roughly 95% AI-generated. As the workforce moves in this direction, graduates need skills that conventional curricula, still built around writing code by hand, seldom teach directly, such as writing precise specifications, supervising AI coding agents, and verifying code that looks plausible but may be wrong or insecure. In a 2024 Cengage survey, 70% of recent graduates said training in generative AI should be integrated into coursework. We present *Vibecoding: Introduction to Applied AI*, an openly licensed curriculum, released early and still under active development. It comprises a textbook, lecture slides, a syllabus, and a hands-on lab for every chapter. The coursework runs from specification writing through building full-stack applications to critically evaluating AI-generated output. It is sequenced in this way so that students build competence with AI tools before confronting the shortcomings and limitations of such tools. We describe the curriculum's design and rationale while outlining our plan to pilot and evaluate it in the classroom.

CCS Concepts

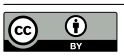
• **Social and professional topics** → **Computer science education**; **Software engineering education**; • **Software and its engineering** → *Designing software.*

Keywords

computing education, AI-assisted programming, curriculum design, vibecoding, prompt engineering

ACM Reference Format:
Andy Luu, Eli Young, and Michael Wehar. 2026. Teaching Vibecoding: An Open Curriculum for AI-Assisted Software Development. In *Proceedings of the 2nd ACM Virtual Global Computing Education Conference V.2 (SIGCSE Virtual 2026), November 12–15, 2026, Virtual Event, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3795860.3846244

*Also with Haverford College.



SIGCSE Virtual 2026, Virtual Event, USA

ACM ISBN 979-8-4007-2491-6/2026/11
https://doi.org/10.1145/3795860.3846244

1 Introduction

Vibecoding is building software by describing what you want in natural language and steering a large language model (LLM) toward a working result, rather than writing it traditionally [25]. The term was popularized in February 2025 by Andrej Karpathy [15]. Since then, AI coding tools have transitioned from primitive code generation by LLMs (like autocomplete) [7] to gigantic agentic systems that can iterate on entire software applications with autonomy. What is becoming increasingly important is knowing what to build, how to design systems, and how to ensure correctness, a shift that is already reshaping how software is built.

Using AI coding tools is now common amongst developers [26]; for instance, a quarter of Y Combinator's Winter 2025 startups reported codebases that were roughly 95% AI-generated [20], and products such as Cursor, Codex, and Claude Code let small teams ship applications with little to no handwritten code. By mid-2026, 90% of professional developers surveyed by JetBrains used AI coding agents at work weekly and 68% daily [3]. Computer science curricula, however, still focus primarily on designing algorithms and writing code manually [2, 17], even though 70% of recent graduates believe undergraduate courses should include basic training in generative AI to prepare students for the workplace [6]. The AI-native workflows that these curricula cover inadequately include properly specifying intent, evaluating or debugging AI-generated code, and supervising increasingly autonomous agents [22].

Prior work has largely characterized generative AI's effects on existing courses, students, and assessments [2, 12, 16, 18]. Vibecoding itself is now taught widely in university courses [4, 8, 27, 28], online short courses [1, 10, 13], and many community tutorials and informal guides [9]. However, up until this point, we have not found a course or tutorial that combines an openly licensed textbook, lecture slides, a syllabus, and graded labs into a single, classroom-ready curriculum. To help close this gap, we present *Vibecoding: Introduction to Applied AI*, an open curriculum that we are developing and have made freely available online as an early, in-progress release.[1] By encouraging learners to specify intent in

Vague prompt

"make a login page"

No framework named. No fields or validation. No error handling, and no definition of "done." The model has to guess all of it.

Specific prompt

"a Next.js login page with email + password, validate the email, show an error on failed sign-in, and redirect on success"

Far less is left to guess, though password storage still needs specifying.

Figure 1: A vague prompt versus a specific one (textbook Chapter 2).

[1] https://algoarena.net/textbook

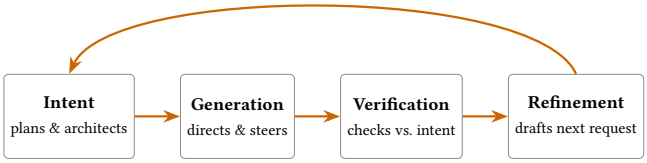


Figure 2: The four-stage vibecoding loop (Chapter 1). Verification also weighs security and personal-data handling.

their own language, the curriculum aims to lower a long-standing entry barrier for non-native English speakers [14, 23] and, because of how effective LLMs are at writing code, to give students without a computer science background a way in. The course outlines topics from the AI tool ecosystem and spec writing (Figure 1) to frontend and backend development, APIs, and agents, eventually arriving at where generated code can go wrong and how to catch it. Published studies show that, as noted above, these proficiencies are essential, yet often neglected [19, 21, 24, 29].

2 Audience and Licensing

Vibecoding: Introduction to Applied AI is openly licensed under CC BY-NC 4.0 so that instructors can adopt and adapt it freely. The course is designed for introductory undergraduate electives without programming prerequisites, and it contains optional chapters that are useful even for students without a programming background. It also complements traditional computer science courses, since students are still encouraged to review code and verify functionality.

2.1 Curriculum Structure

The textbook contains 14 core chapters, which we organize into four related units, and 17 optional chapters allowing instructors to customize pacing and depth (Table 1). Chapter 1 introduces the course's organizing framework: a foundational, four-stage loop that later chapters are based on (Figure 2). Students first build familiarity with AI coding tools across a range of applications (Chapters 3-8) before the course shifts to the critical evaluation of AI-generated output, including the hallucinations and "AI slop" often produced by these models (Chapters 9-10). Verification is the name of one of the four stages introduced in Chapter 1, and each core chapter's lab includes checkpoints where students test whether what they vibecoded behaves as intended. Security and responsible use are covered throughout, from keeping API keys safe and designing proper backend authentication to the final chapter's discussion of privacy, data security, prompt injection defense, copyright, and model bias. We sequence the course this way because we want students to take the risks of overreliance on these tools [5, 24] more seriously after firsthand experience than they would from mere warnings. Future pilots of the curriculum would test this hypothesis.

2.2 Labs and Assessment

Each chapter has a corresponding hands-on lab, with clear goals, step-by-step instructions, and common mistakes to avoid, that is governed by a grading rubric. For example, the Chapter 2 lab, "Spec-to-Ship," has students write a one-page product requirements document (PRD) and build strictly from it, following the course's approach of writing specs before generating [11, 30, 31]. Its rubric grades the final app against its original PRD, along with a log that keeps track of where unclear or unstated requirements led to undesired LLM assumptions. Other labs, like Chapter 5's "Prototype, Then Polish," have students describe a rough idea, however unstructured, prompt for and answer clarifying questions from the model that make the plan concrete, and only then generate code against that plan. As a midterm, students propose their final project in a PRD and complete a timed vibecoding test. The final project then turns that proposal into a complete application that is presented to the class. The instructor guide, slides, labs, and rubrics enable instructors with or without vibecoding experience to effectively teach the whole course or a selection of hand-picked topics.

Table 1: Organizing the core curriculum

Unit	Chapters	Core Competency
Foundations	1-2	Tool ecosystem and specification writing
Build Skills	3-8	AI-native IDEs; context; frontend; backend; authentication; APIs; agents
Critical Judgment	9-10	Spotting flawed AI output; debugging and evaluating AI systems
Extensions	11-14	Multimodal AI; product design; deployment; security and ethics

3 Conclusion

We have presented *Vibecoding: Introduction to Applied AI*, an open and in-progress curriculum that is freely available online. Two limitations apply: (1) it is still being written and refined, and (2) it has not yet been piloted in classrooms. As a result, we cannot substantially report on how well it generalizes across institutions, student populations, or skill levels. It is also subject to decaying relevance because the field is changing rapidly. This 2027 edition is meant to reflect the tool landscape as of late 2026 and is intended for courses beginning in the 2027-28 school year. We plan to pilot the curriculum in introductory undergraduate electives to investigate two research questions. **RQ1:** After completing the course, do students plan more before generating (writing specifications and having the model ask clarifying questions), and do they produce clearer, more correct results? **RQ2:** Does the course improve the ability of students to identify the characteristic ways that AI-generated code fails (e.g., hallucinated APIs, insecure patterns, and output that looks right but is wrong)? We plan to address both research questions with rubric-scored specifications, verification checklists, and program correctness results measured before and after the course. Instructors interested in piloting it are encouraged to contact us.

Acknowledgments

We thank the anonymous SIGCSE Virtual 2026 reviewers for their feedback. AI tools (e.g., Anthropic's Claude) assisted with drafting and editing this manuscript. The authors carefully reviewed and verified all of the presented content.

References

[1] Elliott Adams. 2026. TECH 42: Vibe Coding: Building Software in Conversation with AI. Online course, Stanford Continuing Studies, Fall 2026. https://continuingstudies.stanford.edu/courses/professional-and-personal-development/vibe-coding-building-software-in-conversation-with-ai/20261_TECH-42

[2] Brett A. Becker, Paul Denny, James Finnie-Ansley, Andrew Luxton-Reilly, James Prather, and Eddie Antonio Santos. 2023. Programming Is Hard - Or at Least It Used to Be: Educational Opportunities and Challenges of AI Code Generation. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2023)*. ACM, New York, NY, USA, 500–506. doi:10.1145/3545945.3569759

[3] Mikhail Bogdanov. 2026. AI Coding Agents: Adoption Trends. The JetBrains Blog. Survey of more than 15,000 professional developers, fielded May–July 2026. https://blog.jetbrains.com/research/2026/08/ai-coding-agent-adoption-2026/

[4] Karen Brennan. 2025. Vibe Coding (T564A). Course module, Harvard Graduate School of Education. https://news.harvard.edu/gazette/story/2026/04/vibe-coding-may-offer-insight-into-our-ai-future/

[5] Zana Buçinca, Maja Barbara Malaya, and Krzysztof Z. Gajos. 2021. To Trust or to Think: Cognitive Forcing Functions Can Reduce Overreliance on AI in AI-Assisted Decision-Making. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW1, Article 188 (2021). doi:10.1145/3449287

[6] Cengage Group. 2024. 2024 Graduate Employability Report: Career-Ready Education in the Age of AI. Cengage Group. https://cengage.widen.net/s/bmjxxjx9mm/cg-2024-employability-survey-report

[7] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, et al. 2021. Evaluating Large Language Models Trained on Code. arXiv:2107.03374. doi:10.48550/arXiv.2107.03374

[8] Cornell Tech. 2026. TECHIE 1121: Ethical Vibe Coding. Course, Summer 2026. Taught by H. Sandhaus and J. Segal. https://classes.cornell.edu/browse/roster/SU26/class/TECHIE/1121

[9] Datawhale. 2026. easy-vibe. GitHub repository, open tutorial curriculum for vibe coding. https://github.com/datawhalechina/easy-vibe

[10] DeepLearning.AI. 2025. Vibe Coding 101 with Replit. Online short course. https://www.deeplearning.ai/courses/vibe-coding-101-with-replit/

[11] Paul Denny, Juho Leinonen, James Prather, Andrew Luxton-Reilly, Thezyrie Amarouche, Brett A. Becker, and Brent N. Reeves. 2024. Prompt Problems: A New Programming Exercise for the Generative AI Era. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2024)*. ACM, New York, NY, USA, 296–302. doi:10.1145/3626252.3630909

[12] James Finnie-Ansley, Paul Denny, Brett A. Becker, Andrew Luxton-Reilly, and James Prather. 2022. The Robots Are Coming: Exploring the Implications of OpenAI Codex on Introductory Programming. In *Proceedings of the 24th Australasian Computing Education Conference (ACE '22)*. ACM, New York, NY, USA. doi:10.1145/3511861.3511863

[13] Google Cloud. 2026. Vibe Coding for Beginners: From Zero to App. Online course, Google Skills. Also offered on Coursera. https://www.skills.google/course_templates/1582

[14] Philip J. Guo. 2018. Non-Native English Speakers Learning Computer Programming: Barriers, Desires, and Design Opportunities. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, Article 396, 14 pages. doi:10.1145/3173574.3173970

[15] Andrej Karpathy. 2025. There's a new kind of coding I call "vibe coding". X (formerly Twitter). Post, 2 Feb. 2025. https://x.com/karpathy/status/1886192184808149383

[16] Majeed Kazemitabaar, Justin Chow, Carl Ka To Ma, Barbara J. Ericson, David Weintrop, and Tovi Grossman. 2023. Studying the Effect of AI Code Generators on Supporting Novice Learners in Introductory Programming. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23)*. ACM, New York, NY, USA. doi:10.1145/3544548.3580919

[17] Amruth N. Kumar and Rajendra K. Raj. 2024. Computer Science Curricula 2023 (CS2023): The Final Report. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024)*. ACM, New York, NY, USA. doi:10.1145/3626253.3633405

[18] Sam Lau and Philip J. Guo. 2023. From "Ban It Till We Understand It" to "Resistance is Futile": How University Programming Instructors Plan to Adapt as More Students Use AI Code Generation and Explanation Tools such as ChatGPT and GitHub Copilot. In *Proceedings of the 2023 ACM Conference on International Computing Education Research (ICER '23), Volume 1*. ACM, New York, NY, USA. doi:10.1145/3568813.3600138

[19] Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023. Is Your Code Generated by ChatGPT Really Correct? Rigorous Evaluation of Large Language Models for Code Generation. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*. arXiv:2305.01210. doi:10.48550/arXiv.2305.01210

[20] Ivan Mehta. 2025. A Quarter of Startups in YC's Current Cohort Have Codebases That Are Almost Entirely AI-Generated. TechCrunch. https://techcrunch.com/2025/03/06/a-quarter-of-startups-in-ycs-current-cohort-have-codebases-that-are-almost-entirely-ai-generated/

[21] Neil Perry, Megha Srivastava, Deepak Kumar, and Dan Boneh. 2023. Do Users Write More Insecure Code with AI Assistants?. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23)*. ACM, New York, NY, USA. doi:10.1145/3576915.3623157

[22] James Prather, Paul Denny, Juho Leinonen, Brett A. Becker, Ibrahim Albluwi, Michelle Craig, Hieke Keuning, Natalie Kiesler, Tobias Kohn, Andrew Luxton-Reilly, Stephen MacNeil, Andrew Petersen, Raymond Pettit, Brent N. Reeves, and Jaromir Savelka. 2023. The Robots Are Here: Navigating the Generative AI Revolution in Computing Education. In *Proceedings of the 2023 Working Group Reports on Innovation and Technology in Computer Science Education (ITiCSE-WGR '23)*. ACM, New York, NY, USA. doi:10.1145/3623762.3633499

[23] James Prather, Brent N. Reeves, Paul Denny, Juho Leinonen, Stephen MacNeil, Andrew Luxton-Reilly, João Orvalho, Amin Alipour, Ali Alfageeh, Thezyrie Amarouche, Bailey Kimmel, Jared Wright, Musa Blake, and Gweneth Barbre. 2025. Breaking the Programming Language Barrier: Multilingual Prompting to Empower Non-Native English Learners. In *Proceedings of the 27th Australasian Computing Education Conference (ACE 2025)*. ACM, New York, NY, USA, 74–84. doi:10.1145/3716640.3716649

[24] James Prather, Brent N. Reeves, Juho Leinonen, Stephen MacNeil, Arisoa S. Randrianasolo, Brett A. Becker, Bailey Kimmel, Jared Wright, and Ben Briggs. 2024. The Widening Gap: The Benefits and Harms of Generative AI for Novice Programmers. In *Proceedings of the 2024 ACM Conference on International Computing Education Research (ICER '24), Volume 1*. ACM, New York, NY, USA. doi:10.1145/3632620.3671116

[25] Advait Sarkar and Ian Drosos. 2025. Vibe Coding: Programming through Conversation with Artificial Intelligence. In *Proceedings of the 36th Annual Conference of the Psychology of Programming Interest Group (PPIG 2025)*. arXiv:2506.23253. doi:10.48550/arXiv.2506.23253

[26] Stack Overflow. 2025. 2025 Stack Overflow Developer Survey: AI. Stack Overflow. https://survey.stackoverflow.co/2025/ai

[27] Stanford University. 2025. CS146S: The Modern Software Developer. Course website. https://themodernsoftware.dev/

[28] University of Colorado System. 2025. Vibe Coding Fundamentals. Online course, Coursera. Taught by Scott Reed. https://www.coursera.org/learn/vibe-coding-fundamentals

[29] Priyan Vaithilingam, Tianyi Zhang, and Elena L. Glassman. 2022. Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems (CHI EA '22)*. ACM, New York, NY, USA. doi:10.1145/3491101.3519665

[30] Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C. Schmidt. 2023. A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT. arXiv:2302.11382. doi:10.48550/arXiv.2302.11382

[31] J.D. Zamfirescu-Pereira, Richmond Y. Wong, Bjoern Hartmann, and Qian Yang. 2023. Why Johnny Can't Prompt: How Non-AI Experts Try (and Fail) to Design LLM Prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23)*. ACM, New York, NY, USA. doi:10.1145/3544548.3581388