

DOI: https://doi.org/10.48009/2_iis_2026_142

Into the Discussion of AI Impact on the Computer Programming Occupation – Potential Threats or Emerging Opportunities

Azad Ali, *University of Fairfax*, aali@ufairfax.edu

Fred Kohun, *Robert Morris University*, Kohun@rmu.edu

Shardul Pandya, *American National University*, Spandya@an.edu

Umesh Varma, *University of Fairfax*, ucvarma@ufairfax.edu

Abstract

This paper examines the effects of advances in artificial intelligence (AI) on the job market for computer programmers. It reflects whether this advancement represents an emerging threat to the programming job market or offers new opportunities to the same occupation. The paper presents statistics regarding predictions of the future of Information Technology occupations. It then explains how history repeated itself in different ways, noting that information technology professionals faced similar arguments about technological advances. It focuses on two major events when programming occupations face similar challenges. The paper then details work in computer programming and how artificial intelligence can simplify program development. Lastly, the paper presents a two-column table listing areas of the programming occupation at risk from the increasing use of artificial intelligence. The second column lists the expected emerging opportunities for computer programmers from the introduction of artificial intelligence.

Keywords: Programming occupation, artificial intelligence, programming, artificial intelligence

Introduction

The rapid advancement of artificial intelligence (AI) across education and the workforce has affected many occupations, some insignificantly and others significantly (Maslej et al., 202; Mohammed, 2025). Computer programming is among those occupations significantly affected by AI. Numerous studies predict that many computer programming jobs will be replaced by artificial intelligence, leading to AI's dominance and unemployment among computer programmers (Eloundou et al., 2024). At the same time, other studies suggest that AI will create more computer programming job opportunities rather than eliminate them, and will eventually lead to upskilling among talent prevalent in information technology (IT) occupations, particularly in computer programming jobs (Ellis, 2026).

Computer programming is one of the fields that researchers and analysts predict will be most significantly affected by advances in artificial intelligence (Hananto et al., 2025). Such predictions have raised concerns among students enrolled in Information Technology programs regarding the future of software development careers. Examining these perspectives in greater detail helps clarify both the reasoning

behind these predictions and their potential implications. The remainder of this study is divided into the following sections:

- Reviews employment statistics and projections for the Information Technology (IT) industry, with particular emphasis on the computer programming profession.
- Examines historical technological shifts within the IT industry, demonstrating how previous innovations displaced older technologies while creating new opportunities.
- Explains the computer programming profession and identifies the factors that make certain programming tasks more susceptible to automation.
- Introduces the fundamental concepts of artificial intelligence, providing the background necessary to understand its potential impact on programming-related occupations.
- Presents and discusses the paper's findings by comparing the factors that increase the susceptibility of programming tasks to AI-driven automation with the new opportunities AI is expected to create, highlighting the skills and competencies programmers may need to develop to remain competitive.

Predictions of Future IT Jobs

There is growing anxiety among students enrolled in IT programs and IT professionals that employment opportunities in the IT sector, particularly in computer programming, will decline as artificial intelligence advances. Many fear that fewer programming positions will be available, making employment in the profession increasingly difficult to obtain. This anxiety is fueled by published predictions and forecasts concerning the future of the computer programming occupation (Davenport & Paredes, 2018; Westby & Modestino, 2025). To better understand these concerns, this paper examines several predictions and forecasts and discusses their implications for the profession's future.

The Bureau of Labor Statistics (www.bls.gov) publishes data on its employment outlook for different jobs, organized by category and occupation. The BLS publishes annual tables detailing its job projections. These tables, known as the Occupational Outlook Handbook, gauge the market trends for various jobs. Thus, OOH serves as a job guide or a major employment reference for many seeking jobs in the referenced categories or just wanting to learn the health of certain occupations.

The OOH is divided into 25 major categories, which are then further divided into occupation groups. It publishes articles on job requirements and predictions of job availability for the next 10 years for each category. One of the 25 job categories is called “Computer and Information Technology Occupations”. Below are ten occupations from the “Computer and Information Technology” job category for which data for employment statistics is available:

- Computer and Information Research Scientists
- Computer Network Architects
- Computer Programmers
- Computer Support Specialists
- Computer Systems Analysts
- Database Administrators and Architects
- Information Security Analysts

- Network and Computer Systems Administrators
- Software Developers, Quality Assurance Analysts, and Testers
- Web Developers and Digital Designers

The forecasts OOH provides can significantly shape how people (students, job seekers, and employers) feel about the specific job under consideration. If the prediction is high and rising, students and employers feel good about the job, and program enrollment may increase. Conversely, if job numbers are declining, students enrolled in these fields will be less optimistic, and enrollment could decline.

The Predictions for Job Decline

To set the stage for identifying jobs expected to decline, we searched the OOH for data on IT occupations and found three occupations projected to experience slower job growth. These jobs are presented in Table 1 below. The notable column in this table is the fifth, which shows the predicted employment changes over 10 years, from 2024 to 2034. As shown in the table below, programming jobs are expected to decline by 6%. In other words, the number of programming jobs is expected to decline by 7,300 in 2034 from the 2024 peak. The two other occupations (computer support specialist and Network and computer systems architecture) are expected to decline, but not as sharply as shown for the computer programming occupation.

Table 1 – Occupations Expected to Decline (Source: <https://www.bls.gov/ooh/computer-and-information-technology/>)

The Occupation	2024 Median Pay	# jobs 2024	Employment change 2024-2034	Expected # Jobs 2034	Job outlook, 2024-2034
Computer programmers	\$98,670	121,200	-7,200	113,000	-6% decline
Computer support specialist	\$61,550	882,300	-24,200	858,900	-3% decline
Network and Computer Systems Architectures	\$96,800	331,500	-13,800	317,700	-4% decline

The numbers in the table above indicate a downward trend in demand for the three jobs listed. But this negative trend does not apply to all IT jobs; some IT jobs are expected to see positive trends.

Predictions of Positive Job Growth for IT Occupations

On the other hand, the increasing adoption of artificial intelligence is creating new career opportunities, making the occupations in Table 2 attractive alternatives for current and aspiring IT professionals. The notable column to show here is the fourth, which lists the job outlook for 2024-2034. The most noteworthy outlook here is about the occupation “Computer and Information Research Scientists”. This may represent a direct impact of AI and the increased use of data analytic and this job category is often referred to as “Data Scientists”.

Table 2 – IT Occupations Expected to Expand (Source: <https://www.bls.gov/ooh/computer-and-information-technology/>)

The occupation	2024 pay	#jobs 2024	Employment Change, 2024–34	Total expect 2034 jobs	Job Outlook, 2024–34
Computer and Information Research Scientists	\$140,910	40,300	7,900	48,200	+20% higher than average
Computer network architect	\$130,390	179,200	21,400	200,060	12% (Much faster than average)
Computer Systems Analyst	\$103,790	521,100	45,500	566,600	9% (Much faster than average)
Database Administrators and Architects	\$123,100	144,900	5,300	150,200	4% (As fast as average)
Information Security Analysts	\$124,910	182,800	52,100	234,900	29% (Much faster than average)
Software Developers, Quality Assurance Analysts, and Testers	\$131,450	1,895,500	287,900	2,183,400	15% (Much faster than average)

A Closer Look at the Data for the Programming Occupation

Reading the two tables above reveals that only a few occupations will be negatively affected by changes in the technology market. The computer programming occupation is expected to decline by 6% between 2024 and 2034. Since this occupation is most affected over the next 10 years, we decided to dig deeper into the OOH data for computer programming to provide a clearer picture and present other relevant information. Table 3 below shows more detailed information and quick facts about the computer programming occupation.

Table 3 – A Closer Look at Programming Occupation from BLS (Source: <https://www.bls.gov/ooh/computer-and-information-technology/computer-programmers.htm>)

Quick Facts: Computer Programmers	
2024 Median Pay	\$98,670 per year \$47.44 per hour
Typical Entry-Level Education	Bachelor's degree
Work Experience in a Related Occupation	None
On-the-job Training	None
Number of Jobs, 2024	121,200
Job Outlook, (2024–2034)	-6% (Decline)
Employment Change, 2024–34	-7,200

A closer look at Table 3 reveals worrisome data. The number of available jobs for this occupation is expected to decline by 7,200, representing 6% of the jobs currently occupied by programmers. So, instead of facing job growth like the other IT occupations, opening more jobs, the programming occupation will face a significant decline.

The preceding discussion suggests growing concern about the declining number of employment opportunities in the computer programming profession and the potential for future job displacement as artificial intelligence continues to evolve.

However, this change in demand is not isolated from other technological changes that took place in the past. History shows that IT faced similar technological shifts; a brief examination of prior technological changes may help understand the pattern of their effects on the job market.

The Theory of History Repeats Itself

The theory of History repeats itself, or “historical parallel” refers to a set of similar events that are repeated at different times in history (Trew & Wilson, 2012). They could refer to the pattern of weather changes and their effect. We often hear the word "El Niño " repeated to refer to a pattern of weather changes that took place in the past and is being repeated in a similar fashion that happened before. It can also refer to battles between neighboring countries that recur throughout history. They can be used simply to describe history or to predict the future based on similar past events. In the case of Information technology, it could refer to recurring patterns of technological change (Liang et al., 2009).

Regarding patterns of change in the Information Technology field, educators and practitioners are familiar with rapid technological shifts that eliminate some jobs while creating others requiring new, related skills (Trivedi et al., 2023). These changes often build on prior experience rather than making it irrelevant. For example, punch cards were a primary method of data entry from roughly the 1890s through the 1960s, and operating them required specialized skills (IBM, n.d.). In the 1970s, keyboards began replacing punch cards as the dominant data-entry tool. Over time, keyboards evolved into a broader range of input devices and data-entry roles. Workers who had experience with punch-card systems were therefore able to adapt by applying their existing knowledge to newer technologies and workplace demands.

Similarly, throughout the 1970s, learning computer skills was synonymous with learning about mainframes. Knowledge and experience working on different parts of the mainframe were once required to enter the field of computer jobs (Khandelwal, 2025).

The list goes on, with different technologies and cycles of technological updates that remove certain required skills from the job market while introducing a new set. The cycle of technological change and its subsequent effect on the job market can be illustrated by two major events in history: first, the events that occurred after the year 2000 (termed Y2K), and second, the rise of web development.

The Y2K Era

The growth of programming languages provides another example of repeated technological change and its effect on the job market. The turn of the century and the Y2K issue illustrate how rapidly changing technology can reshape workforce expectations. At that time, three programming languages were especially prominent: BASIC in education, COBOL in business applications, and Fortran in scientific applications. As a result, entering the programming profession often required skill in one of these languages. Many observers expected Y2K to disrupt this pattern and feared that programmers specializing in these languages would lose job opportunities after the turn of the century.

The events that followed Y2K were different from many of the early predictions. Some legacy hardware and software positions did disappear from the market, but new technology-related jobs also emerged. Many technology professionals responded by retraining and reskilling to work with newer systems, allowing them to continue working in the field. Stuart and Cooney (2008) describe this type of skill transition as “side skill” development. A similar cycle may occur as AI continues to evolve and expand across different fields.

The Era of Web Design and WYSIWYG

A more relevant example is the development of web pages and the HTML code used to create them. As it is known in web development, writing code to display a web page requires learning HTML syntax. This can be extended to learning other tools, such as JavaScript and cascading style sheets, if additional features are needed for the web page. This was changed when the emerging web page technology introduced the “What You See Is What You Get” (WYSIWYG) model. This latest technology model does not require learning syntax or programming rules to create web pages; instead, simple drag-and-drop can generate many lines of HTML code. Thus, users can create web pages without learning about web programming (Ali & Kohun, 2025).

That expectation was only true to a small extent. Some jobs were lost, but many others were created to fill the market demand. Technological advances created more jobs that favor those skilled in learning the

three languages: BASIC, COBOL, and Fortran. Then came a wave of new programming languages and applications that opened innovative avenues for people with the same application. In terms of the job market, despite losing some programming jobs on older platforms, the development of front-end, back-end, security, and other platforms kept the programming job market alive and growing (Ali & Kohun, 2025).

If we apply what was noted above to the new forthcoming changes of the AI technology and the jobs that will be created from AI, three points can be drawn from this conversation:

- There will be programming jobs that are going to be replaced by AI
- There will be other opportunities that will emerge from the use of AI that will keep programming jobs alive
- Programming skills will remain to have advantages for some newly created jobs by AI. However, to maintain these advantages, programmers may need to improve their skills (upskill) to remain employed in the technology job market.

Programming Occupation – How it is Performed

At its simplest, computer programming means giving computer instructions to complete specific tasks. These instructions must follow the rules and guidelines, or syntax, of a programming language. Although syntax rules can be numerous, they can be learned through practice and, in some cases, supported or partially automated by development tools. Many students initially struggle with syntax, which can lead to errors. As the tasks a program must perform become more complex, the program's structure and components also become more difficult to manage (Bhattacharjee, 2024).

The Bureau of Labor Statistics describes computer programmers as professionals who write and test computer code to develop software. While this definition is useful, it simplifies the broader work involved in programming. In practice, writing and testing code requires knowledge of programming logic, syntax rules, development processes, testing procedures, and software requirements.

Programming tasks can range from simple exercises to complex applications that include modules, conditions, branches, files, and multiple processing rules. In academic settings, students often begin with simple examples, such as printing “Hello World,” converting Celsius to Fahrenheit, or converting kilometers to miles. These exercises help introduce basic concepts, but they remain far removed from the complexity of real-world software development (Mselle & Kondo, 2014).

Programming can also be classified in various ways, such as batch versus interactive or online versus offline. Batch programming typically accesses data from files and follows a predefined sequence of steps to produce an expected output. Interactive programming is less predictable because the program's execution depends on user input and other conditions that may not be known in advance (Stewart, 1999).

Another important concept is boilerplate code, which refers to reusable code segments that can be applied across multiple programs. For example, a program segment that calculates the difference between two dates may be written once and then reused in several financial applications. Because boilerplate code is

often smaller, more standardized, and less complex than full applications, it typically requires less effort to create and adapt (Nam et al., 2019).

In real-world applications, programmers face many factors that increase software complexity. In the past, complexity was often measured by the number of lines of source code. Over time, this measure became less sufficient because modern programs may include multiple variables, interfaces, files, systems, dependencies, and other conditions (Ghazarian, 2015). Poorly organized code is sometimes described as “spaghetti programming,” a term for code that is difficult to understand, maintain, provide updates or modify (Abbes et al., 2011). Complexity increases further when several programmers work on the same program, especially when each programmer uses different coding styles, file structures, or maintenance practices independently from others.

The Basics of Artificial Intelligence

Although the discussion so far has addressed artificial intelligence broadly, Generative Artificial Intelligence (GenAI) is especially relevant to the programming occupation. GenAI represents a significant new frontier in AI because of its potential to improve productivity, transform communication, and reshape workflows. What began as a single model has expanded into many specialized GenAI models designed for different purposes (Joshi, 2025).

GenAI has already had a substantial impact on computer programming by simplifying code writing. Instead of manually writing code while strictly following a programming language’s syntax and logic, a user can describe a task in ordinary written or spoken language (Anand, 2021). For example, a user might ask GenAI to “Write a Java program that allows the entry of five exam scores, calculates the average, identifies the minimum and maximum scores, and prints the results.” GenAI can then generate code that performs these tasks without requiring the user to know the programming language’s syntax or logic in detail.

This simple example shows how GenAI can support code generation even when the user has limited knowledge of the programming language. However, as programming tasks become more complex, users benefit from understanding both the basic concepts behind GenAI and the process of prompt engineering.

Prompt engineering is the practice of carefully designing the instructions, questions, examples, and constraints fed into a GenAI system to produce more accurate, relevant, and useful outputs (IBM n.d.-b). In programming contexts, this may include specifying the programming language, defining the required inputs and outputs, identifying error-handling needs, or asking the system to revise and improve earlier responses (Brynjolfsson & Raymond, 2025). Therefore, prompt engineering is important because the quality of the instructions given to GenAI can strongly influence the quality, accuracy, and usefulness of the generated code.

A prompt in AI is the text an individual enters to communicate with a generative AI model. The purpose of these prompts is to get answers to questions, complete specific tasks, and fulfill similar requests (Solly, 2024). The user provides a question or request, and GenAI generates a response by drawing on a massive database (Walter, 2024). While this process may seem simple, the text entered into the prompt

significantly influences the relevance and accuracy of the output. Therefore, effective prompt engineering is a crucial skill for obtaining high-quality results from GenAI (Chen et al., 2025).

The popular adage GIGO, or “garbage in, garbage out,” applies directly to GenAI. If the prompts are incomplete or incomprehensible to AI, the resulting output from GenAI will likely be of poor quality or not directly relevant to the prompt. However, if the prompts are clear and follow the correct protocols, the likelihood of receiving consistent, relevant answers from GenAI is significantly higher (Hutson & Cotroneo, 2023).

The key takeaway is that prompts entered into GenAI must be carefully crafted, or “engineered,” to increase the likelihood of receiving a direct, relevant response. This process of meticulously designing queries to enter GenAI is known as prompt engineering (Chen et al., 2025). It is important to clarify that prompt engineering is not always a single question-and-answer exchange. If the initial response is not satisfactory, a user can ask follow-up questions. GenAI typically retains information from earlier prompts and builds upon it. This process of entering additional questions and related data is known as an “iterative prompt engineering” (Wang et al., 2024).

Another relevant point is that through user-GenAI collaboration, both sides can learn from each other. GenAI can learn from the user through the multiple inquiries and prompts they enter. The same thing happens to users when they learn from GenAI by observing the different responses to the prompts they enter (Sowa & Przegalinska, 2025).

Discussion and Conclusion

This section synthesizes the main points discussed in the paper and presents them in the two-column table developed for this study. It first examines the programming tasks and job roles most likely to be affected by the increased use of GenAI. It then discusses the career opportunities that may arise as GenAI becomes more widely adopted in computer programming.

Potential Threats to the Programming Occupation

Based on the preceding discussion, the following two broad principles help identify programming tasks that are more likely to slow down or be automated as GenAI becomes more widely used:

- Tasks that are simple, routine, and low in programming complexity
- Tasks that are repetitive and require little or no user or programmer intervention

Using these principles, this paper identifies five programming-related areas that may be especially vulnerable to automation through GenAI: entry-level programming positions, routine or boilerplate coding tasks, batch programming, basic program testing, and basic front-end web development. Each area is discussed below.

Entry-level programming roles often involve tasks with relatively low complexity. Because these assignments may focus on basic syntax, simple logic, or narrowly defined coding requirements, they are more likely to be supported or automated by GenAI tools than tasks requiring deeper design, analysis, or problem-solving expertise.

Boilerplate tasks involve creating reusable code segments that can be applied in multiple places with little or no modification. Because these code segments are typically standardized, have limited scope, and are less dependent on complex decision-making, they are well-suited for automation. GenAI can generate or adapt this type of code quickly when the requirements are clearly stated.

Batch programming generally follows a predefined sequence of steps to perform repetitive tasks on a regular schedule, such as daily, weekly, or monthly. Because these programs often begin at a defined starting point, process data according to established rules, and end at a predictable output, they usually require little real-time user interaction. For these reasons, some batch programming tasks may be increasingly handled by AI-supported tools.

Program Testing & Debugging: Program development often includes testing to ensure the software functions correctly and meets its specifications. Basic testing may involve limited data inputs, expected outputs, and repeated comparisons between actual and expected results. When test cases are predictable and repetitive, GenAI and related automation tools can help generate test cases, identify discrepancies, and support repeated testing cycles. As a result, basic testing tasks may be reduced or reshaped by automation.

Front-end web development focuses on the part of a website or application that users see and interact with. In the past, creating web pages required manually writing HTML and testing the page before deployment. WYSIWYG tools simplified this process by allowing users to create pages visually, and GenAI has further reduced the need to write basic front-end code manually. Therefore, simple front-end web development tasks may increasingly be delegated to GenAI tools rather than assigned to programmers (Jadhav & Gholve, 2024).

New Career Opportunities from GenAI

As GenAI becomes more widely integrated into software development, programmers may be able to transition into emerging roles that build on their existing technical knowledge. These opportunities generally require upskilling in areas such as data analysis, machine learning, prompt engineering, AI governance, and human-AI collaboration. Five career paths are especially relevant: data science, AI prompt engineering, machine learning, AI ethics, and vibe coding.

Data Science: Data science continues to expand as organizations collect larger volumes of data and seek to extract meaningful value from it. Although job requirements vary, programming knowledge remains central to this field because data scientists often write scripts, clean and transform data, build models, and interpret results (Mohammed, 2025). The Bureau of Labor Statistics projects strong growth for this occupation, suggesting that data science may offer a promising pathway for programmers who strengthen their skills in analytics, statistics, and data-driven decision-making.

AI Prompt Engineering: Prompt engineering is becoming increasingly important as programmers use GenAI tools to generate, test, explain, and refine code. Effective prompt engineers understand how to translate technical requirements into clear instructions, evaluate AI-generated outputs, and iterate until the result meets project goals. Programmers are well-positioned for this work because their knowledge of logic, syntax, testing, and software requirements helps them guide GenAI systems more effectively.

Machine Learning: Machine learning practitioners develop systems that identify patterns in data and improve performance over time without requiring every rule to be explicitly programmed. This work often involves selecting algorithms, preparing datasets, training models, evaluating results, and refining performance. Programmers who build expertise in algorithms, data structures, statistics, and model development can apply their existing coding skills to this growing area.

AI Ethics and Governance: As GenAI becomes more widespread, organizations need professionals to evaluate whether AI systems are accurate, fair, transparent, and responsible. AI ethics specialists help identify bias, protect privacy, assess risks, and promote responsible use of AI-generated content and code. Programmers can contribute to this field by combining technical knowledge with an understanding of ethical, legal, and social implications.

Vibe Coding: Vibe coding is the practice of directing GenAI tools to create, revise, test, and improve software using natural-language instructions. Instead of writing every line of code manually, the programmer guides the AI system, reviews its output, identifies errors, and requests refinements through repeated iterations (Meske et al., 2025; Ray, 2025). This approach still benefits from programming expertise, as users must assess whether the generated code is correct, efficient, secure, and aligned with the intended requirements.

Conclusion

Drawing on the discussion above, the paper concludes that GenAI is likely to reshape rather than simply eliminate the computer programming occupation. Some programming tasks—especially routine, repetitive, and low-complexity work—may be slowed down or automated. At the same time, GenAI is expected to create new opportunities for programmers who expand their skills (upskill) in data science, prompt engineering, machine learning, AI ethics and governance, and AI-assisted software development. We thought it would be best to present this summary in a two-column table for easier reference.

Table 4 summarizes these areas of risk and opportunity. Although the table has two columns and five rows, but they are not presented to show comparison between the two columns. Instead, they are presented. Instead, we introduced them to show five areas of programming more likely to be automated. At the same time, the second column shows potential jobs that programmers can “upskill” to more than people from other occupations.

Table 4 – Programming Tasks at Risk and Emerging AI-Related Career Opportunities

Programming tasks most likely to slow down or be automated	Emerging career opportunities for programmers
Entry-level programming tasks	Data science
Routine coding and boilerplate tasks	AI prompt engineering
Repetitive batch programming tasks	Machine learning
Basic program testing tasks	AI ethics and governance
Basic front-end web development tasks	Vibe coding and AI-assisted software development

References

- Abbes, M., Khomh, F., Gueheneuc, Y. G., & Antoniol, G. (2011, March). An empirical study of the impact of two antipatterns, blob and spaghetti code, on program comprehension. In *2011 15Th european conference on software maintenance and reengineering* (pp. 181-190). IEEE. <https://doi.org/10.1109/csmr.2011.24>
- Ali, A., & Kohun, F. (2025). A snapshot at the future of teaching computer programming courses from the lenses of artificial intelligence-A comparison with teaching web design courses. *Issues in Information Systems*, 26(3).
- Anand, R. (2021). The Impact of Artificial Intelligence on Program Management Jobs Worldwide: Challenges, Opportunities, and Implications. *Opportunities, and Implications (May 02, 2021)*.
- Bhattacharjee, S. (2024). What is the software complexity? Know the Challenges and solutions. <https://vfunction.com/blog/software-complexity/>
- Brynjolfsson, E., Li, D., & Raymond, L. (2025). Generative AI at work. The Quarterly Journal of Economics, 140(2), 889–942. <https://doi.org/10.1093/qje/qjac044>
- Chen, C., Hu, W., & Wei, X. (2025). From anxiety to action: exploring the impact of artificial intelligence anxiety and artificial intelligence self-efficacy on motivated learning of undergraduate students. *Interactive Learning Environments*, 33(4), 3162-3177. <https://doi.org/10.1080/10494820.2024.2440877>
- Davenport, T., & Paredes, M. (2018). Can We Predict What Jobs AI Will Take?. *World*, 75, 133-000.
- Ellis, L. (2026, May 19). These Companies Say AI Is Reviving Entry-Level Jobs, Not Killing Them. *Wall street journals*. https://www.wsj.com/lifestyle/careers/ai-entry-level-jobs-graduates-595cee28?mod=hp_lead_pos2
- Eloundou, T., Manning, S., Mishkin, P., & Rock, D. (2024). GPTs are GPTs: Labor market impact potential of LLMs. *Science*, 384(6702), 1306–1308. <https://doi.org/10.1126/science.adj0998>
- Ghazarian, A. (2015, May). A theory of software complexity. In *2015 IEEE/ACM 4th SEMAT Workshop on a General Theory of Software Engineering* (pp. 29-32). IEEE. <https://doi.org/10.1109/GTSE.2015.11>

- Hananto, A., Hasibuan, N. S., Olivia, S., Tukino, T., & Novalia, E. (2025). The Impact of Artificial Intelligence on Programmer Jobs: Threat or Opportunity?. *JUSIFO (Jurnal Sistem Informasi)*, 11(1), 11-20.
- Hutson, J., & Cotroneo, P. (2023). Generative AI tools in art education: Exploring prompt engineering and iterative processes for enhanced creativity. *Metaverse*, 4(1).
<https://doi.org/10.54517/m.v4i1.2164>
- IBM. (n.d.). *Punch Card Data Processing – Hursley Museum*. <https://ibmhursleymuseum.org/punch-card-data-processing/>. <https://ibmhursleymuseum.org/punch-card-data-processing/>
- IBM. (n.d.-b). *What is prompt engineering? | IBM*. <https://www.ibm.com/think/topics/prompt-engineering>. <https://www.ibm.com/think/topics/prompt-engineering>
- Jadhav, S. S., & Gholve, S. S. (2024). The impact of AI on web development. *International Journal of Scientific Research in Modern Science and Technology*, 3(8), 07-12.
<https://doi.org/10.59828/ijrmst.v3i8.240>
- Joshi, S. (2025, April 22). Introduction to Generative AI: Its Impact on Jobs, Education, Work and Policy Making. SSRN <https://doi.org/10.2139/ssrn.5206473>
- Khandelwal, A. P. (2025). AI-Driven Mainframe Modernization: Unlocking Legacy Data for Cloud Analytics. *Journal Of Engineering And Computer Sciences*, 4(6), 60-67.
- Liang, B., Sepehrnoori, K., & Delshad, M. (2009). An automatic history matching module with distributed and parallel computing. *Petroleum Science and Technology*, 27(10), 1092-1108.
- Maslej, N., Fattorini, L., Perrault, R., Gil, Y., Parli, V., Kariuki, N., Capstick, E., Reuel, A., Brynjolfsson, E., Etchemendy, J., Ligett, K., Lyons, T., Manyika, J., Niebles, J. C., Shoham, Y., Wald, R., & Walsh, T. (2025). *The AI Index 2025 annual report*. Stanford University, Institute for Human-Centered AI. <https://hai.stanford.edu/research/ai-index-report>
- Meske, C., Hermanns, T., Von der Weiden, E., Loser, K. U., & Berger, T. (2025). Vibe coding as a reconfiguration of intent mediation in software development: Definition, implications, and research agenda. *IEEE Access*, 13, 213242-213259. <https://doi.org/10.2139/ssrn.5378201>
- Mohammed, S. (2025). Will AI replace statistical programming jobs. *Journal of Management Research and Analysis*; 12(3), 185-189. <https://doi.org/10.18231/j.jmra.38192.1758705703>

- Mselle, L. J., & Kondo, T. S. (2014). Against the “Hello World”. *International Journal of Computer Applications*, 95(26), 17-22.
- Nam, D., Horvath, A., Macvean, A., Myers, B., & Vasilescu, B. (2019, November). Marble: Mining for boilerplate code to identify API usability problems. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)* (pp. 615-627). IEEE. <https://doi.org/10.1109/ase.2019.00063>
- Ray, P. P. (2025). A review on vibe coding: Fundamentals, state-of-the-art, challenges and future directions. *Authorea Preprints*. <https://doi.org/10.36227/techrxiv.174681482.27435614/v1>
- Solly, N. (2024, April 3). *Prompt Engineering: The Key to Enhancing AI's Contextual Understanding* <https://mohara.co/insights/prompt-engineering-the-key-to-enhancing-ais-contextual-understanding/>
- Sowa, K., & Przegalinska, A. (2025). From expert systems to generative artificial experts: A new concept for human-ai collaboration in knowledge work. *Journal of Artificial Intelligence Research*, 82, 2101-2124. <https://doi.org/10.1613/jair.1.17175>.
- Stewart, T. J. (1999). Concepts of interactive programming. In *Multicriteria Decision Making: Advances in MCDM Models, Algorithms, Theory, and Applications* (pp. 277-304). Boston, MA: Springer US.
- Stuart, M., & Cooney, R. (2008). training and the limits of supply-side skill development. *Industrial Relations Journal*, 39(5), 346-353.
- Trew, A., & Wilson, G. (Eds.). (2012). *Past, present, parallel: a survey of available parallel computer systems*. Springer Science & Business Media.
- Trivedi, A., Kaur, E. K., Choudhary, C., & Barnwal, P. (2023, March). Should AI technologies replace the human jobs?. In *2023 2nd International Conference for Innovation in Technology (INOCON)* (pp. 1-6). IEEE. <https://doi.org/10.24321/2456.429x.202504>
- Walter, Y. (2024). Embracing the future of artificial intelligence in the classroom: The relevance of AI literacy, prompt engineering, and critical thinking in modern education. *International Journal of Educational Technology in Higher Education*, 21(1), 15. <https://doi.org/10.1186/s41239-024-00448-3>

- Wang, L., Song, R., Guo, W., & Yang, H. (2024). Exploring prompt pattern for generative artificial intelligence in automatic question generation. Interactive Learning Environments. *Advance online publication*. <https://doi.org/10.1080/10494820.2024.2412082>
- Westby, A., & Modestino, A. S. (2025). The impact of generative AI on job opportunities for junior software developers (Working paper). Boston University. https://aliciasassermodestino.com/wp-content/uploads/2025/06/Impact_of_GenAI_on_SWEs_061625.pdf.