Engineering Skills Model (ESM)

Technical Report Version 2.0

Table of Contents

Background

Codility believes that the future of effective and transformational hiring and talent management is skills-based, fair, and transparent. Its mission is to become the leading platform for assessing and advancing technical skills in an AI-driven world.

To advance this mission, Codility helps many organizations systematically identify the skills required for success in each role. A critical first step in building talent management practices is to define the job-required skills or competencies. Identifying job-relevant skills is a prerequisite for valid and defensible talent practices, as outlined in the *Uniform Guidelines on Employee Selection Procedures* (Equal Employment Opportunity Commission et al., 1978) and in the *Principles for the Validation and Use of Personnel Selection Procedures* (Society for Industrial and Organizational Psychology, 2018), adopted as policy by the American Psychological Association (APA).

Yet traditional engineering skill taxonomies often fall short. They tend to fragment work into hundreds of highly granular skills. Most commonly, programming languages and frameworks such as Python, JavaScript, React, or Django are treating each as a discrete capability. This creates two problems:

1. **The overwhelming volume of entries makes these taxonomies unwieldy for hiring, training, or benchmarking.**

2. **They miss the bigger picture. By reducing engineering to lists of tools and syntax, they fail to capture the underlying competencies that actually drive performance.**

Codility's Assessment Science team set out to address this gap. While maintaining and updating coverage of specific programming languages and frameworks remains important (see Appendices A and B), the team wanted to design a skill model centered on higher-order, role-relevant competencies. Rather than asking whether an engineer knows Python or React, the model could evaluate how they apply knowledge to real-world challenges. In this way, it would reflect the reality of engineering work: success depends not on memorizing tools but on solving problems, architecting systems, and building reliable software.

Building on nearly five years of research, Codility's Assessment Science team of tenured industrial-organizational (I-O) psychologists has developed a conceptual framework that describes the skills required for success in technical roles. **Codility's Engineering Skills Model (ESM)** provides a holistic view of the important skills across an array of technical job families and can be used for effective talent assessment and development. This technical report documents the development, methodology, and evolution of the ESM, tracing how the model has advanced over the past several years.

The ESM functions as both a competency model for customers and a roadmap that informs the Codility content team in designing, updating, and prioritizing task creation over time. That said, not every competency in the ESM has a one-to-one task available. While every task on the platform has skills that are represented within the ESM, the model is intentionally forward-looking and extends beyond today's content coverage.

Model Development and Evolution

Codility's Engineering Skills Model (ESM) was developed through a multi-year, evidence-based process grounded in best practices for job analysis and competency modeling. Codility's Assessment Science team conducted literature reviews, expert interviews, and large-scale analyses of technical job data to identify recurring patterns in the skills required for success. The model was refined iteratively through validation studies, input from subject-matter experts, and application across diverse technical roles. It continues to evolve to reflect the shifting demands and emerging practices of the software engineering landscape.

Initial Development of ESM 1.0

The first version of the Engineering Skills Model (ESM 1.0) was created to provide a structured, research-backed foundation for assessing technical talent. Though intentionally broad, this version served as a critical starting point, enabling organizations to align hiring and development practices with job-relevant requirements. The process used to develop the categories and their contents is summarized below.

Technical Skills

Codility's Assessment Science team examined the language-agnostic skills required across job levels for six technical job families to build the technical skills section of the model. These job families included backend developers, frontend developers, data scientists, quality assurance (QA) engineers, mobile developers, and developer operations (DevOps) professionals.

To ensure coverage across all job families, the Assessment Science team reviewed more than 30 software engineering skill and competency frameworks from public and private organizations. Examples include the IEEE Computer Society's Software Engineering Competency Model (SWECOM) (Fairley, 2014), the Software Engineering Body of Knowledge (SWEBOK) (Bourque & Fairley, 2014), the SEI Software Assurance Competency Model (Hilburn et al., 2013), and the Inclusive Engineering Framework (Bonfield, n.d.). The team also examined professional training curricula, academic programs, and published research. A full list of sources is available upon request.

The job information was reviewed and consolidated into a streamlined skills taxonomy. The team then engaged over 25 internal subject matter experts (SMEs) and 22 external SMEs to refine the taxonomy and align it with the six job families described earlier. In job-family-specific focus groups, SMEs provided quantitative ratings of each skill's importance and whether it was required at entry for various engineering roles. Based on these ratings, the team finalized the initial list of technical skills.

Intrapersonal and Collaboration Skills

In addition, the Assessment Science team reviewed existing job information and skill-related resources (e.g., systems and software engineering practice overviews, training curricula, competency models, and research articles) to identify cross-functional, **non-technical skills** required for success in technical roles. These soft skills are foundational to engineering careers, as they enable the effective application of technical expertise to job-related behaviors.

The team identified an initial set of 16 non-technical skills, grouped into two categories: **intrapersonal skills**, which enable effective functioning in a professional environment, and **collaboration skills**, which support working effectively with others. A panel of seven Codility engineering managers representing diverse specialties reviewed the list and provided feedback on the accuracy and clarity of each skill's name and definition. They also rated each skill as essential or not for effective performance across representative software and systems engineering roles and added comments. Skills judged essential by at least two-thirds of the managers were retained.

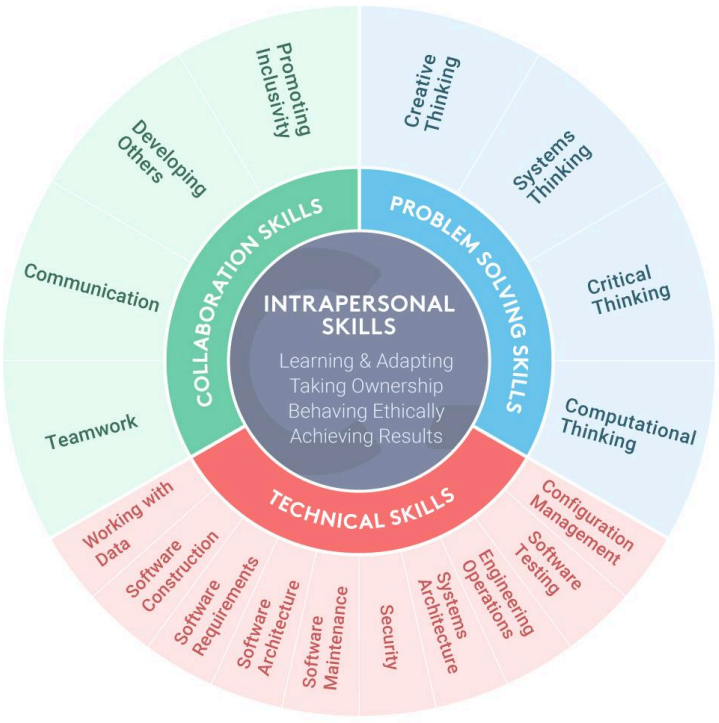
Problem-Solving Skills

Upon reviewing and consolidating the job information and skill frameworks gathered (e.g., MITRE's Systems Engineering Competency Model [Metzger & Bender, 2007]; INCOSE Systems Engineering Competency Framework [Presland et al., 2018]), the team determined that a **problem-solving skills** category was necessary. Six critical skills were initially identified: computational thinking, systems thinking, creative thinking, continual learning, design thinking, and product thinking. Because the definitions of design thinking and product thinking overlapped considerably, these were later combined into a single "design/product thinking" category.

Internal and External SME Sampling

The Assessment Science team convened 11 demographically diverse, experienced internal and external SMEs to review the preliminary model, including its skill and subskill names, definitions, and categories. The sampling plan ensured a well-rounded representation of expertise within the target domain. Internal participants included engineering managers and individual contributors from several job families (e.g., frontend, backend, DevOps, data science), as well as Codility leaders such as the founder, CTO, and two technical board members. To provide a broader industry perspective, the team also invited two external CTOs from midmarket and enterprise technology organizations with established engineering teams to join the expert panel.

SMEs were reminded that ESM skill and subskill names, definitions, and category labels should be broad enough to apply across roles and seniority levels, yet flexible enough to reflect the skill profile needed for a specific role, team, or project. Meeting these criteria ensures that the ESM can support diverse company strategies, projects, and talent needs by providing a holistic view of an engineer's skills for a defined role.



SME Feedback Review and Consolidation

Codility's Assessment Science team iteratively reviewed SME feedback from the preliminary model to develop the final model (ESM 1.0). Written and verbal feedback was compiled into proposed changes to skill names, definitions, and the placement of skills and subskills. Through a series of discussions, the team evaluated these proposals and accepted, modified, or rejected them once consensus was reached. Key

changes included clarifying vague definitions, consolidating overlapping skills, reassigning subskills, and adding skills that SMEs collectively agreed were essential.

After generating a final version of the model (ESM 1.0), the team re-engaged SMEs (including engineering leaders and experts) to confirm these model changes. Thus, the foundational version of this model encapsulated SME feedback, job information drawn from competency models, training curricula, and professional standards, review by experienced I-O psychologists, a second and third round of SME feedback and confirmation, and a post-hoc comparison with the beta V4 version of IEEE's SWEBOK.

ESM 1.0 employed a distinctive wheel-shaped design. The design highlighted the equal importance of technical, collaboration, and problem-solving skills while emphasizing the central role of intrapersonal skills. This presentation reinforced the growing need for engineers to develop well-rounded skill sets.

AI-Assisted Work Analysis (ESM 1.1)

A key design requirement was that the ESM include skills essential for software engineering roles both **now and in the future**. In Q4 2023, Codility's Assessment Science team conducted additional research to identify skills particularly important for work with generative AI. The team reviewed academic and best-practice literature on generative AI and prompt engineering and consulted with several thought leaders in the field.

To expand this research, the team gathered resources on AI-assisted software engineering, incorporated relevant work activities from O*Net job titles, and engaged early adopters of AI coding tools (e.g., Copilot, ChatGPT). These SMEs either used AI tools regularly in their workflows or served as engineering leaders determining how their teams would adopt them. All SMEs were experienced engineers or engineering leaders representing several job families, including frontend, backend, DevOps, infrastructure, and full stack engineering. In addition to interviewing SMEs about their personal experiences working with AI systems, the team also reviewed sample ChatGPT transcripts and observed SMEs working directly with AI tools.

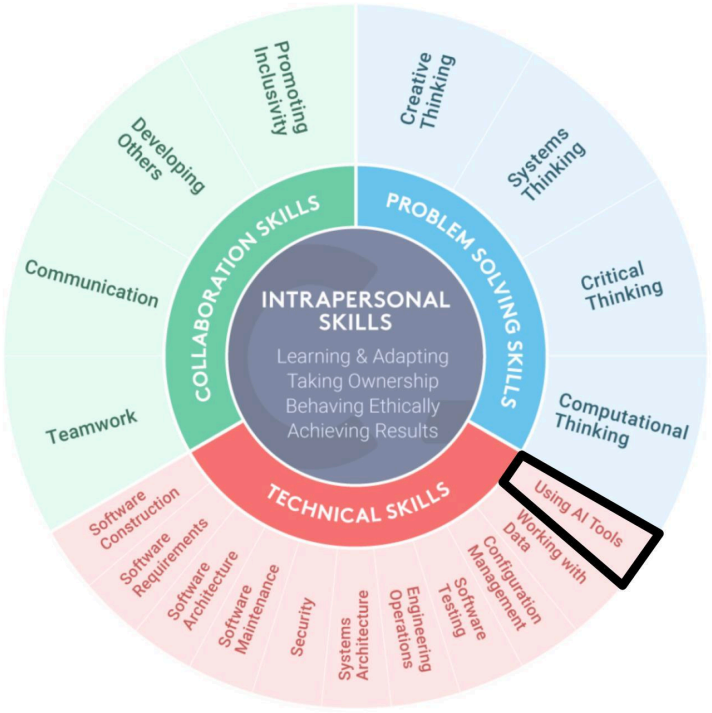
After synthesizing SME interview and job observation feedback, the team generated 23 unique work activity statements that described how engineers relied on AI systems in practice. To check the general relevance of these activities, the team compared them against baseline work activities from O*Net, which led to the removal of two statements and a final set of 21. From this set, several new skills emerged that had not been captured in the original ESM. Together, these skills described different ways engineers needed to work with or around AI systems, from writing prompts and checking outputs to managing data pipelines and tailoring large models for specific applications. These included:

- **Prompt Writing:** Crafting clear, concise, and effective prompts or questions to elicit specific responses or outputs from an AI system or large language model.
- **AI Proofreading:** Reviewing and editing in-line text or code output generated by AI systems.
- AI Output Validation: Evaluating the appropriateness and reliability of outputs produced by AI systems using tests and other methods (e.g., retrieval-augmented generation).
- **AI Models:** Applying knowledge of AI models, their functionalities, limitations, and potential impacts on engineering systems and processes.

- **Large Language Model (LLM) Frameworks:** Understanding and applying LLM architecture, training processes, and use cases; tailoring LLMs for specific applications based on consideration of the models' computational and data requirements.
- **Machine Learning Operations:** Applying knowledge of various stages of machine learning project development, including data preparation, model training, deployment, monitoring, and maintenance.
- **Large-Scale Data Analysis:** Analyzing, interpreting, and gleaning insights from large datasets using data analytics tools and methodologies, statistical analysis, and data visualization techniques.

To strengthen these findings, the team brought in another group of six Codility engineers who had at least three months of experience working with AI tools in their daily workflows. This second group represented a range of engineering roles and was asked to confirm the linkages between activities and skills. The rule was that a work activity would only be tied to a skill if at least two-thirds of the SMEs agreed on the connection (see Appendices C and D). The review showed that three skills stood out as most central to AI-assisted engineering work:

- Prompt Writing
- AI Proofreading
- AI Output Validation



At the same time, several traditional ESM skills, such as software construction, software maintenance, and computational thinking, also showed strong connections, reinforcing their continuing importance.

From this review, the team concluded that many of the skills required for generative AI and future engineering roles were already represented in the ESM. However, they determined it was necessary to add **Using AI Tools** as a new technical skill, reflecting its growing importance among software engineers. In addition, they identified one problem-solving skill as critical for effective use of generative AI tools in technical roles: computational thinking. For a full discussion of computational thinking and its importance to software engineering, see Wing (2006).

Quantitative Survey (ESM 1.2)

In line with psychometric best practices, multiple sources of evidence, both qualitative and quantitative, are needed to establish validity (AERA, APA, & NCME, 2014; SIOP, 2018). Incorporating quantitative methods also reduces the risk of bias inherent in expert judgment and strengthens the defensibility of personnel decisions (Cizek, 2020). Building on this foundation, in the summer of 2024 Codility's Assessment Science team conducted a quantitative survey to capture criticality and required-upon-entry ratings for ESM skills across six technical job families. The survey aimed to confirm that each skill was critical and required upon entry for at least one job family, validate the distinction between **standard and specialized skills**, and examine the relevance of newly added skills across job families and their degree of importance.

Data were collected by inviting software developers visiting Codility's developer training website or assessment platform to complete the survey in exchange for entry into a $100 gift drawing. Instead of using a Likert-style importance scale, respondents indicated whether each skill was critical, required upon entry, or both, using yes/no categories. This format reduced response time and cognitive load while also applying a more stringent standard than traditional job analysis importance ratings. Tables 1, 2, and 3 show the respondent frequencies by experience, seniority, and job family. The respondent sample represents 24 different countries with India, the US, and Brazil being the most frequent.

Table 1

Survey Respondents by Years of Experience

Years of Experience	n	%
Less than a year	6	5.8
1-4 years	45	44.7
5-9 years	26	25.2
10-14 years	16	15.5
15+ years	10	9.7

Table 2

Survey Respondents by Seniority

Seniority	n	%
Entry-level (0-2 years of experience)	11	10.7
Mid-level (3-5 years of experience)	27	26.2
Senior (5+ years of experience)	50	48.5
Manager or supervisor	12	11.7
Director or above	2	1.9

Table 3

Survey Respondents by Job Family

Job Family	n	%
Backend	32	31.1
Full stack	32	31.1
Other: open-ended	9	8.7
Data science	7	6.8
DevOps	7	6.8
Frontend	7	6.8
QA	6	5.8
Security	2	1.9
Mobile	1	1.0

Appendix E presents the survey results. With the exception of the newer AI skills, all skills were rated as critical by at least 25% of respondents across job families. These results quantitatively confirmed the relevance of these skills to technical job roles. They also showed that, although still emerging, the newly added AI skills were considered critical for at least some job families. The ESM was updated with these quantitative results and released as version 1.2.

Addition of AI Skills (ESM 2.0)

One of the most significant recent changes to the ESM was driven by the rapidly evolving technological landscape of AI. While the ESM included Using AI Tools as a technical skill, the broader implications of AI adoption within engineering extended beyond tool use. For engineering roles, success increasingly depended on a deep understanding of AI technologies and their underlying mechanics. Mastery of these skills enabled technical employees not only to incorporate AI capabilities into existing systems but also to design innovative solutions that leveraged AI to create business value.

Additionally, the team recognized that AI-related skills were no longer limited to software engineers. Non-engineering employees (i.e., the general workforce) increasingly relied on AI tools to achieve success, making effective collaboration between engineers and the broader workforce an essential focus.

In Q1 2025, Codility's Assessment Science team conducted a literature review on AI-Readiness and AI Literacy to explore the skillsets necessary for success across both the traditional technical workforce and the non-technical general workforce. Research consistently showed that AI literacy was a multidimensional construct encompassing knowledge, application, evaluation, and ethical considerations. Long and Magerko (2020) defined AI literacy as a set of competencies enabling individuals to critically evaluate AI technologies, collaborate effectively with AI, and use AI as a tool in daily and professional contexts. Similarly, Ng et al. (2021) proposed four foundational aspects—knowing and understanding AI, applying AI, evaluating and creating AI, and addressing ethical issues—that were widely adopted in subsequent frameworks.

Recent scale development further reinforced this multidimensionality. Laupichler et al. (2023) validated the Scale for the Assessment of Non-Experts' AI Literacy (SNAIL), identifying three core factors: technical understanding, critical appraisal, and practical application. Soto-Sanfiel et al. (2024) expanded this perspective with the SAIL4ALL scale, which organized competencies into four domains: What is AI?, What can AI do?, How does AI work?, and How should AI be used? Lintner's (2024) systematic review of 16 AI literacy scales similarly concluded that consistent competencies emerged across diverse populations—technical knowledge, practical application, critical evaluation, and ethics—as the backbone of AI readiness.

To adapt these findings for organizational use, Codility supplemented the literature with interviews and focus groups involving internal and external technology leaders, collectively representing more than 120 years of experience. The proposed skills were refined through multiple review and revision cycles to ensure both validity and relevance. This research led to the enhancement of the ESM framework through the addition of new **AI-Readiness Skills**, structured into four distinct skill categories:

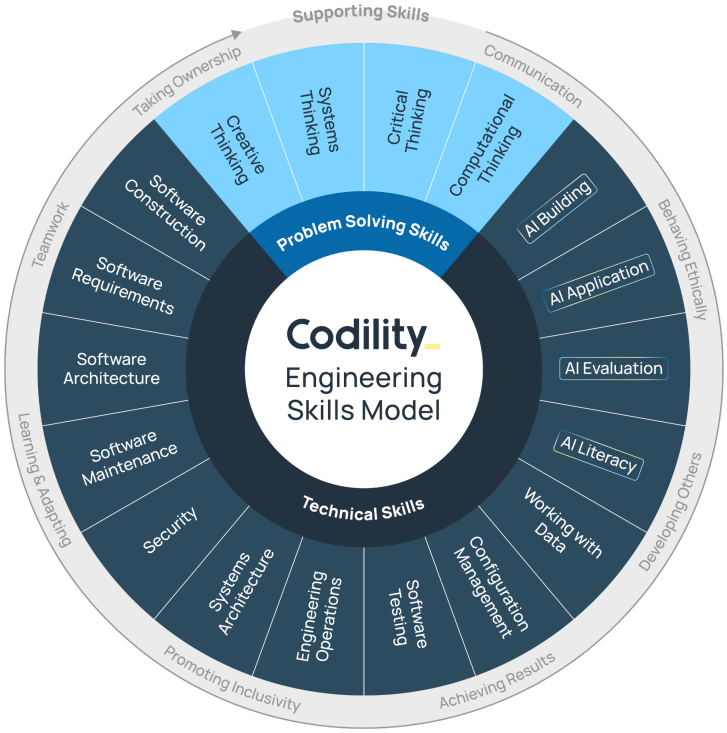
- **AI Literacy:** Understanding core concepts of artificial intelligence, including its capabilities, limitations, ethical considerations, and common applications.
- **AI Evaluation:** Assessing the accuracy, reliability, and appropriateness of AI systems and outputs, including bias detection and validation against objectives and ethical standards.
- **AI Application:** Utilizing artificial intelligence in practical, hands-on scenarios. This includes skills such as creating AI-driven workflows, writing effective prompts for AI tools, and integrating AI solutions into existing systems.
- **AI Building:** Designs systems that leverage machine learning and artificial intelligence to solve specific problems or enhance processes.

Upon finalizing the skill categories and subskills, SMEs were also asked to indicate whether each skill was important exclusively for engineering roles or for both engineering and non-technical employees. This step ensured that the framework captured not only technical expertise but also cross-functional competencies relevant to the broader workforce. For example, technical roles might require advanced skills such as "Model Maintenance," while general workforce roles benefit from accessible skills like "AI Output Proofreading." The results of this exercise are presented in Appendix F.

Beyond the new skill categories, new subskills such as "Data Privacy" were also integrated into older skill categories (e.g., Working with Data) to address emerging priorities.

Current Model (ESM 2.0)

The ESM offers a broad and easily adaptable approach to assessing skills across various engineering roles, industries, and organizations. Its simplicity and versatility make it a valuable tool for creating skill-based frameworks for engineering teams, allowing them to adjust skill targets as technology evolves. A complete list of skills, including definitions and subskills, is provided in Appendix G.



The current version of the ESM (2.0) is presented in a wheel-shaped design that highlights the balanced integration of skill domains. At the center, the model anchors Technical Skills, reflecting the core of engineering work. At the top are Problem-Solving Skills, which cut across domains and support innovation and adaptability. The outer layer illustrates Supporting Skills that enable engineers to apply technical expertise effectively in organizational contexts. The Assessment Science team determined that interpersonal and collaboration skills were best represented as supporting skills, since they are less directly measurable in a screening assessment and are more appropriately evaluated through technical interviews and other interactive methods. In contrast, technical and problem-solving skills were placed at the center because they are not only foundational to engineering success but also align with Codility's platform strengths, where these domains can be reliably assessed at scale.

Codility's Assessment Science team designed the ESM to be applicable to most technical roles (and even some non-technical roles), easily adjustable as target jobs and technologies change, and flexible to a range of proficiency levels and skill sets. Still, the team continually monitors the ESM by researching and gathering feedback to explore how changes in the field might signal changes in the skills required for effective practice. Iteration is vital for any product, software, or system to stay relevant, and this will be a primary focus of Codility's research moving forward.

References

American Educational Research Association, American Psychological Association, & National Council on Measurement in Education. (2014). *Standards for educational and psychological testing.* American Educational Research Association.

Bonfield, D. (n.d.). *Inclusive engineering framework.* Inclusive Engineering. Retrieved September 22, 2023, from http://www.inceng.org/inclusive-engineering-framework.html

Bourque, P., & Fairley, R. E. (2014). *Guide to the software engineering body of knowledge (SWEBOK®): Version 3.0.* IEEE Computer Society Press.

Cizek, G. J. (2020). *Validity: An integrated approach to test score meaning and use.* Routledge.

Equal Employment Opportunity Commission, Civil Service Commission, Department of Labor, & Department of Justice. (1978). *Uniform guidelines on employee selection procedures. Federal Register, 43*(166), 38290–38315.

Fairley, R. E. (2014). *A software engineering competency model (SWECOM).* IEEE Computer Society Press.

Hilburn, T., Ardis, M., Johnson, G., Kornecki, A., & Mead, N. (2013, March 11). *Software assurance competency model (Technical Note CMU/SEI-2013-TN-004).* Software Engineering Institute. https://doi.org/10.1184/R1/6583991.v1

Laupichler, M. C., Aster, A., Haverkamp, N., & Raupach, T. (2023). Development of the "Scale for the assessment of non-experts' AI literacy" – An exploratory factor analysis. *Computers in Human Behavior Reports, 12,* 100338. https://doi.org/10.1016/j.chbr.2023.100338

Lintner, T. (2024). A systematic review of AI literacy scales. *npj Science of Learning, 9,* 50. https://doi.org/10.1038/s41539-024-00264-4

Long, D., & Magerko, B. (2020). What is AI literacy? Competencies and design considerations. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (pp. 1–16). Association for Computing Machinery. https://doi.org/10.1145/3313831.3376727

Metzger, L. S., & Bender, L. R. (2007). *MITRE systems engineering (SE) competency model* (Version 1.13E). The MITRE Corporation.

National Center for O*NET Development. (n.d.). Browse by cross-functional skills. O*NET OnLine. Retrieved September 22, 2023, from https://www.onetonline.org/find/descriptor/browse/2.B

Ng, D. T. K., Leung, J. K. L., Chu, K. W. S., & Qiao, M. S. (2021). AI literacy: Definition, teaching, evaluation and ethical issues. *Proceedings of the Association for Information Science and Technology, 58*(1), 504–509. https://doi.org/10.1002/pra2.487

Presland, I., Beasley, R., Gelosh, D., Heisey, M., & Zipes, L. (2018). *INCOSE systems engineering competency framework* (INCOSE Technical Product Reference: INCOSE-TP-2018-002-01.0). International Council on Systems Engineering (INCOSE).

Society for Industrial and Organizational Psychology. (2018). *Principles for the validation and use of personnel selection procedures* (5th ed.). American Psychological Association.

Soto-Sanfiel, M. T., Angulo-Brunet, A., & Lutz, C. (2024). *The Scale of Artificial Intelligence Literacy for all (SAIL4ALL): A tool for assessing knowledge on artificial intelligence in all adult populations and settings.* [Preprint]. https://doi.org/10.31234/osf.io/x94s7

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35.

Appendices

Appendix A: Programming Languages Covered

Programming Languages Covered

C	C#	C++	COBOL
Dart	Elixir	Go	Haskell
Java	JavaScript	Kotlin	Lua
Objective-C	Pascal	Perl	PHP
Python	R	Ruby	Rust
Scala	Solidity	SQL	Swift
TypeScript	Visual Basic		

Note: This table lists programming languages at a general level. If you need detailed information on supported versions, compilers, runtimes, and databases, please refer to Codility's support article: What technologies does Codility support?

Appendix B: Technologies Covered

Technologies Covered			
.NET	Android	Angular	Ansible
Apache Camel	Apache Spark	ASP.NET Core	AWS
AWS Lambda	Azure	Bash	Blockchain
Chef	CSS	Cucumber	DB2
Django	Docker	DynamoDB	Elasticsearch
Entity Framework	Excel	Express.js	FastAPI
Flask	Flutter	Git	GitLab CI
Google Cloud Platform	Google Slides	GraphQL	Hibernate
HTML	iOS	Java Streams	Jenkins
Jest	jQuery	JSON	JUnit
Kubernetes	Laravel	Linux	Mocha
MongoDB	Mulesoft	MySQL	Node.js
NoSQL	Open API	Pandas	Powerpoint
PowerShell	Puppeteer	PySpark	PyTest
React	React Native	Redux	Regular Expressions
Robot	RSpec	Ruby on Rails	Salesforce
Selenium	Serverless Framework	Shell	Spring
Spring Boot	Symfony	Tableau	Terraform
Vue.js	XML	xUnit	YAML

Note: This table lists frameworks and libraries at a general level. If you need detailed information on supported versions, compilers, runtimes, and databases, please refer to Codility's support article: What technologies does Codility support?

Appendix C: Preliminary Work Activity Statements and Skill Linkages

Work Activity Label	Work Activity	Software Construction	Software Maintenance	Prompt Writing	Software architecture	AI proofreading	Systems Architecture	AI output validation	Software Testing	AI models	Software requirements	Large Language Model Frameworks	Engineering Operations	Machine Learning Operations	Systems Thinking
Coding Basic Functions	Write prompts that generate code for well-defined tasks (e.g., function that computes the determinant of a matrix, rotates a vector, makes an AWS query).	X		X		X									
Suggesting Code	Choose or modify automated code suggestions in development environments (e.g., suggesting functions and parameters relevant to the task).	X													
Generating Boilerplate Code	Write prompts or accept suggestions to generate boilerplate code, particularly in situations where the code structure is standard and repetitive.	X		X		X									
Streamlining Tests	Write prompts to generate tests for existing functions.	X	X	X				X	X						
Writing Documentation	Write prompts to generate code documentation and commentary.		X	X		X									
Generating Complex Data Processing Code	Write prompts to generate code to handle complex data processing tasks (e.g., fuzzy matching).	X		X				X							
Extracting and Formatting Data	Write prompts to extract and format data (e.g., non-standard formats such as PDFs containing tables with non-natural language symbols).	X		X											
Detecting and Explaining Errors	Ask questions to identify and understand errors in complex coding syntax, particularly in niche languages or specific tools (e.g., makefiles).	X	X			X									
Debugging and Making Quick Fixes	Ask questions or write prompts to debug and generate fixes for simpler coding problems (i.e., enhancing the feedback loop in the development process).		X	X				X							
Infrastructure Creation and Debugging Support	Write prompts to generate simple infrastructure-related code and debug, particularly in cloud computing environments.			X		X		X					X		
Searching and Adapting Documentation	Ask questions to search documentation and write prompts to adapt documentation for specific coding needs (e.g., creating policies in Terraform or understanding specific software behaviors).		X			X									
Learning a Codebase	Ask questions to learn new tools and technologies without a deep dive into underlying systems.														
Learning New Concepts	Ask questions to learn new programming concepts or technologies, particularly for more complex or abstract ideas.														

Work Activity Label	Work Activity	Software Construction	Software Maintenance	Prompt Writing	Software architecture	AI proofreading	Systems Architecture	AI output validation	Software Testing	AI models	Software requirements	Large Language Model Frameworks	Engineering Operations	Machine Learning Operations	Systems Thinking
Communicating and Collaborating	Write prompts to draft announcements, generate training content, and explain technical concepts in a more accessible manner.			X											
Brainstorming and Project Planning Assistance	Ask questions and write prompts to brainstorm ideas for breaking down problems or conceptualizing projects.			X											X
Evaluating AI output	Critically assess AI-generated outputs for accuracy, relevance, and quality, comparing them against established benchmarks or criteria.					X									
Validating AI output	Confirm the reliability and consistency of AI outputs across various scenarios to ensure they meet required standards and are fit for their intended purpose.							X							
Understanding AI models	Analyze and comprehend the capabilities and applications of AI models to effectively choose and manage AI systems that best achieve work goals.									X					
Developing with Large Language Model	Develop and maintain services leveraging large language models, with a focus on user experience and product features.											X			
Implementing Machine Learning Models	Identify strategic opportunities for implementing machine learning models to enhance product experiences.													X	
Working with Large-Scale Datasets	Engage in the analysis of large-scale datasets utilizing statistical and machine learning tools.														

Preliminary Work Activity Statements and Skill Linkages (Continued)

Work Activity Label	Work Activity	Large-Scale Data Analysis	Working with Data	Configuration Management	Computational Thinking	Security	Critical Thinking	Creative Thinking	Communication	Developing Others	Promoting Inclusivity	Teamwork	Learning and Adapting	Taking Ownership	Behaving Ethically	Achieving Results
Coding Basic Functions	Write prompts that generate code for well-defined tasks (e.g., function that computes the determinant of a matrix, rotates a vector, makes an AWS query).															
Suggesting Code	Choose or modify automated code suggestions in development environments (e.g., suggesting functions and parameters relevant to the task).															
Generating Boilerplate Code	Write prompts or accept suggestions to generate boilerplate code, particularly in situations where the code structure is standard and repetitive.															
Streamlining Tests	Write prompts to generate tests for existing functions.															
Writing Documentation	Write prompts to generate code documentation and commentary.								X							
Generating Complex Data Processing Code	Write prompts to generate code to handle complex data processing tasks (e.g., fuzzy matching).		X		X											
Extracting and Formatting Data	Write prompts to extract and format data (e.g., non-standard formats such as PDFs containing tables with non-natural language symbols).		X		X											
Detecting and Explaining Errors	Ask questions to identify and understand errors in complex coding syntax, particularly in niche languages or specific tools (e.g., makefiles).															
Debugging and Making Quick Fixes	Ask questions or write prompts to debug and generate fixes for simpler coding problems (i.e., enhancing the feedback loop in the development process).															
Infrastructure Creation and Debugging Support	Write prompts to generate simple infrastructure-related code and debug, particularly in cloud computing environments.															
Searching and Adapting Documentation	Ask questions to search documentation and write prompts to adapt documentation for specific coding needs (e.g., creating policies in Terraform or understanding specific software behaviors).															
Learning a Codebase	Ask questions to learn new tools and technologies without a deep dive into underlying systems.												X			

Work Activity Label	Work Activity	Large-Scale Data Analysis	Working with Data	Configuration Management	Computational Thinking	Security	Critical Thinking	Creative Thinking	Communication	Developing Others	Promoting Inclusivity	Teamwork	Learning and Adapting	Taking Ownership	Behaving Ethically	Achieving Results
Learning New Concepts	Ask questions to learn new programming concepts or technologies, particularly for more complex or abstract ideas.												X			
Communicating and Collaborating	Write prompts to draft announcements, generate training content, and explain technical concepts in a more accessible manner.								X	X						
Brainstorming and Project Planning Assistance	Ask questions and write prompts to brainstorm ideas for breaking down problems or conceptualizing projects.				X			X								
Evaluating AI output	Critically assess AI-generated outputs for accuracy, relevance, and quality, comparing them against established benchmarks or criteria.															
Validating AI output	Confirm the reliability and consistency of AI outputs across various scenarios to ensure they meet required standards and are fit for their intended purpose.															
Understanding AI models	Analyze and comprehend the capabilities and applications of AI models to effectively choose and manage AI systems that best achieve work goals.															
Developing with Large Language Model	Develop and maintain services leveraging large language models, with a focus on user experience and product features.															
Implementing Machine Learning Models	Identify strategic opportunities for implementing machine learning models to enhance product experiences.															
Working with Large-Scale Datasets	Engage in the analysis of large-scale datasets utilizing statistical and machine learning tools.	X														

Appendix D: Skill Labels, Definitions, and Number of SME Linked Work Activities

Skill Label	Skill Definition	# Activities linked (4 or more SMEs)
Prompt Writing	Craft clear, concise, and effective prompts or questions designed to elicit specific responses or outputs from an AI system or large language model.	10
AI proofreading	Reviewing and refining the output generated by artificial intelligence systems, particularly in text form.	10
AI output validation	Assessing and confirming the appropriateness and reliability of outputs produced by AI systems.	9
Software Construction	Developing software by implementing user interfaces, managing errors, asynchronous tasks, and memory, utilizing object-oriented principles, algorithms, and APIs, while supporting cross-platform compatibility, scripting, concurrency, event-driven and functional programming, recursion, linear and low-level programming, and ensuring user accessibility.	8
Software Maintenance	Continuously managing and improving software code to sustain its functionality, performance, and reliability.	5
Computational Thinking	Formulating and solving problems using strategies common to computer science and programming. Breaking down complex problems into smaller, more manageable components and expressing solutions in a language that a computer can execute.	3
Working with Data	Examining and interpreting data through querying, visualization, reporting, and statistical/ML modeling to gain insights and make informed decisions. Includes data preparation, sourcing, exploration, and ongoing model tuning and maintenance.	2
Critical Thinking	Evaluating information and arguments through reasoned analysis to make well-informed decisions, solve problems, and facilitate learning. It involves skepticism, logical reasoning, and the ability to question assumptions and biases.	2
Creative Thinking	Generating original ideas, finding unique solutions to problems, and combining existing ideas in new ways.	2
Communication	Conveying information, ideas, thoughts, or feelings effectively and clearly to others through various means, such as verbal, written, non-verbal, or digital channels. Listening actively, expressing oneself coherently, adapting communication style to different audiences, and comprehending and interpreting information from others accurately.	2
Learning and Adapting	Responding to volatility, uncertainty, complexity, and ambiguity (VUCA) with curiosity and flexibility. Seeking new ideas and approaches by applying learning to work and discarding ineffective methods. Adapting plans, priorities, and goals in response to difficult conditions, tight deadlines, obstacles, or setbacks.	2
Software architecture	Understands fundamental software structures such as design patterns, microservices, object-oriented design, and code structure.	1
Software Testing	Applying software testing techniques, levels, and automation to ensure software functionality, reliability, and quality.	1
AI models	Awareness of different AI models, their functionalities, limitations, and potential impacts on engineering systems and processes.	1
Large Language Model Frameworks	Understands large language model architecture, training processes, use cases, and how to tailor them for specific applications, as well as understands the computational and data requirements of these models.	1
Engineering Operations	Planning, delivering, and controlling operations to streamline software development. Applying automation, using continuous integration and delivery (CI/CD), and applying deployment strategies and infrastructure as code paradigms, while adhering to security and observability practices.	1
Machine Learning Operations	Applies knowledge of various stages of machine learning project development, including data preparation, model training, deployment, monitoring, and maintenance.	1
Systems Thinking	Understanding how different parts of a system interrelate and how changes in one part of the system affect the whole. Taking into account the user's perspective and the broader context in which the system and its components will function together to fulfill its intended objectives and satisfy user requirements.	1

Skill Label	Skill Definition	# Activities linked (4 or more SMEs)
Large-Scale Data Analysis	Analyzing, interpreting, and gleaning insights from extensive datasets; requires proficiency in data analytics tools and methodologies, statistical analysis, and data visualization techniques	1
Configuration Management	Understanding and implementing processes and tools to track, control, and document individual components and assets within IT systems and software to ensure consistency, traceability, and reproducibility throughout their lifecycle (e.g., setting up build systems, makefiles, versioning software components, maintaining a comprehensive record of system configurations).	1
Developing Others	Developing, guiding, teaching, coaching, and mentoring others toward achieving personal and organizational goals. Delivering effective feedback on an ongoing basis to improve performance.	1
Systems Architecture	Determining system configurations, monitoring systems, employing architectural principles, and analyzing/debugging systems to ensure robust and efficient software systems.	0
Software requirements	Collects and communicates functional and non-functional requirements for a software product and translates them into technical designs, while considering user and business perspectives and the technical context .	0
Security	Safeguarding software and systems through adherence to secure coding standards, implementation of cryptographic techniques, and ensuring robust infrastructure security measures to protect against potential threats and vulnerabilities.	0
Promoting Inclusivity	Respecting, appreciating, and encouraging contributions and participation from all individuals, including those with diverse backgrounds and cultures. Fostering an inclusive mindset that evaluates perspectives, practices, and products from different cultural perspectives.	0
Teamwork	Working effectively with others on shared tasks (e.g., co-creating, pair programming). Resolving conflicts through facilitation and reconciliation. Actively participating in team discussions, assisting other members in addressing problems, and emphasizing the team's collective interests. Readily sharing information and knowledge with the team. Acknowledging and advocating for others' contributions.	0
Taking Ownership	Taking responsibility for one's decisions and behaviors, holding oneself accountable for meeting work objectives within set timeframes, and taking action without waiting for direction.	0
Behaving Ethically	Exhibiting integrity, trustworthiness, honesty, and fairness in one's decisions and actions. Demonstrating global, social, intellectual, and technological responsibility.	0
Achieving Results	Setting goals for personal and group accomplishment, monitoring progress toward goal attainment, and working to meet or exceed goals. Persisting to accomplish tasks. Being outcome-driven versus output-driven.	0

Appendix E: Survey Results for Criticality and Required Upon Hire (RUH)

	Backend		Data science		DevOps		Frontend		Full stack		QA		Security		Other		Mobile		Total	
	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH
Design patterns	0.75	0.44	0.57	0.29	0.71	0.29	1.00	0.43	0.78	0.31	0.67	0.33	1.00	0.00	0.78	0.11	1.00	0.00	0.77	0.33
Software Architecture	0.78	0.38	0.57	0.14	0.71	0.14	0.86	0.43	0.75	0.25	0.83	0.17	1.00	0.00	0.78	0.22	1.00	0.00	0.77	0.27
Basic Programming	0.66	0.53	0.57	0.29	0.71	0.14	1.00	0.57	0.81	0.38	0.83	0.17	1.00	0.00	0.78	0.11	1.00	1.00	0.76	0.38
Error Exception handling	0.72	0.41	0.86	0.14	0.57	0.29	0.86	0.43	0.78	0.28	0.83	0.17	1.00	0.00	0.56	0.22	1.00	0.00	0.75	0.30
Object oriented Programming	0.72	0.50	0.29	0.29	0.57	0.29	0.86	0.43	0.88	0.28	0.83	0.17	1.00	0.00	0.56	0.11	1.00	1.00	0.74	0.34
Data structures	0.72	0.44	0.71	0.14	0.57	0.29	0.71	0.43	0.78	0.38	0.83	0.17	0.50	0.50	0.67	0.33	1.00	0.00	0.73	0.36
Asynchronous operations	0.78	0.28	0.43	0.29	0.43	0.43	1.00	0.57	0.84	0.31	0.33	0.67	0.50	0.50	0.44	0.11	1.00	1.00	0.71	0.34
Debugging	0.75	0.50	0.71	0.29	0.57	0.14	0.71	0.43	0.66	0.34	1.00	0.00	1.00	0.00	0.56	0.33	1.00	1.00	0.71	0.36
Teamwork	0.72	0.41	0.71	0.14	0.57	0.00	0.86	0.43	0.69	0.25	0.83	0.00	0.50	0.00	0.67	0.22	1.00	1.00	0.71	0.27
Code readability	0.78	0.38	0.57	0.14	0.57	0.00	0.71	0.14	0.72	0.25	0.67	0.33	1.00	0.00	0.44	0.33	0.00	0.00	0.69	0.26
Software requirements	0.69	0.34	0.57	0.14	0.43	0.29	0.86	0.43	0.72	0.28	0.83	0.17	1.00	0.00	0.56	0.22	1.00	1.00	0.69	0.29
Code optimization	0.75	0.25	0.57	0.29	0.57	0.29	0.57	0.14	0.66	0.28	0.67	0.17	1.00	0.00	0.56	0.33	1.00	1.00	0.67	0.26
Secure Coding Standards	0.69	0.28	0.57	0.14	0.57	0.29	0.71	0.43	0.69	0.31	0.67	0.33	1.00	0.00	0.56	0.44	0.00	0.00	0.66	0.30

	Backend		Data science		DevOps		Frontend		Full stack		QA		Security		Other		Mobile		Total	
	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH
Communication	0.66	0.47	0.71	0.14	0.71	0.14	0.71	0.29	0.59	0.28	0.83	0.00	0.50	0.00	0.56	0.33	1.00	1.00	0.65	0.31
Leveraging APIs	0.75	0.44	0.57	0.29	0.71	0.14	0.86	0.29	0.56	0.34	0.50	0.33	0.50	0.50	0.56	0.33	1.00	0.00	0.65	0.35
Behaving Ethically	0.59	0.50	0.71	0.00	0.71	0.14	0.86	0.29	0.59	0.28	0.83	0.00	0.50	0.00	0.56	0.22	1.00	1.00	0.64	0.30
Concurrency	0.69	0.28	0.43	0.29	0.71	0.14	0.57	0.43	0.66	0.47	0.50	0.50	1.00	0.00	0.56	0.00	1.00	0.00	0.64	0.32
Taking Ownership	0.59	0.44	0.71	0.00	0.71	0.14	0.86	0.43	0.56	0.41	0.83	0.17	0.50	0.00	0.67	0.22	1.00	1.00	0.64	0.34
Achieving Results	0.44	0.41	0.71	0.00	0.71	0.14	0.86	0.57	0.72	0.34	0.83	0.17	0.00	0.00	0.67	0.22	1.00	0.00	0.63	0.31
Learning and Adapting	0.56	0.41	0.57	0.14	0.71	0.14	0.86	0.29	0.53	0.34	0.83	0.00	0.50	0.00	0.67	0.11	1.00	0.00	0.61	0.28
System Design	0.66	0.28	0.43	0.29	0.57	0.29	0.57	0.57	0.66	0.28	0.33	0.67	0.50	0.50	0.67	0.11	1.00	0.00	0.61	0.31
Engineering Operations	0.69	0.31	0.43	0.43	0.57	0.29	0.57	0.43	0.56	0.34	0.67	0.33	0.50	0.50	0.56	0.56	1.00	0.00	0.60	0.36
Release Management	0.72	0.34	0.29	0.57	0.71	0.14	0.71	0.43	0.47	0.38	0.50	0.33	1.00	0.00	0.67	0.22	1.00	0.00	0.60	0.34
Software Design Review	0.66	0.22	0.43	0.29	0.43	0.14	0.57	0.43	0.59	0.28	0.50	0.50	0.50	0.50	0.78	0.22	1.00	0.00	0.60	0.27
Critical Thinking	0.66	0.31	0.57	0.29	0.57	0.29	0.86	0.29	0.53	0.22	0.67	0.00	0.50	0.00	0.33	0.33	1.00	0.00	0.59	0.25
Algorithms	0.56	0.38	0.43	0.29	0.29	0.57	0.71	0.71	0.66	0.50	0.83	0.17	0.50	0.50	0.44	0.22	1.00	0.00	0.58	0.42
Incident Management	0.56	0.31	0.29	0.43	0.57	0.29	0.29	0.29	0.63	0.28	0.83	0.17	1.00	0.00	0.67	0.22	1.00	1.00	0.58	0.29
Recursion	0.59	0.22	0.43	0.14	0.57	0.14	0.71	0.29	0.66	0.31	0.50	0.50	0.50	0.50	0.44	0.11	0.00	0.00	0.58	0.25

	Backend		Data science		DevOps		Frontend		Full stack		QA		Security		Other		Mobile		Total	
	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH
Creative Thinking	0.66	0.22	0.43	0.43	0.29	0.43	0.86	0.29	0.44	0.38	0.83	0.17	0.50	0.00	0.78	0.44	0.00	0.00	0.57	0.31
Event driven Programming	0.66	0.25	0.57	0.29	0.57	0.29	0.86	0.43	0.53	0.34	0.33	0.67	1.00	0.00	0.33	0.22	0.00	0.00	0.57	0.31
Software Testing Techniques	0.59	0.25	0.43	0.43	0.43	0.14	0.43	0.00	0.56	0.41	1.00	0.00	1.00	0.00	0.44	0.22	1.00	0.00	0.57	0.26
System Debugging	0.69	0.16	0.29	0.43	0.71	0.14	0.43	0.29	0.53	0.44	0.33	0.50	1.00	0.00	0.56	0.11	1.00	0.00	0.57	0.28
Configuration Management	0.56	0.34	0.29	0.57	0.57	0.29	0.57	0.29	0.59	0.25	0.50	0.50	1.00	0.00	0.56	0.33	1.00	1.00	0.56	0.33
System Monitoring	0.56	0.34	0.71	0.14	0.71	0.14	0.43	0.43	0.56	0.34	0.50	0.50	1.00	0.00	0.33	0.22	1.00	0.00	0.56	0.31
Systems thinking	0.66	0.38	0.29	0.43	0.43	0.14	0.86	0.29	0.50	0.28	0.50	0.33	0.50	0.00	0.56	0.56	1.00	0.00	0.56	0.33
Developing Others	0.59	0.19	0.57	0.14	0.57	0.29	0.71	0.43	0.47	0.31	0.67	0.00	0.50	0.00	0.56	0.11	0.00	0.00	0.55	0.22
Quality assurance	0.44	0.22	0.71	0.14	0.29	0.43	0.43	0.29	0.63	0.31	0.83	0.33	1.00	0.00	0.56	0.11	1.00	0.00	0.55	0.25
Resource Management	0.63	0.25	0.57	0.00	0.57	0.14	0.57	0.29	0.50	0.28	0.50	0.33	1.00	0.00	0.33	0.22	1.00	1.00	0.55	0.24
Computational Thinking	0.63	0.25	0.57	0.14	0.43	0.29	0.71	0.29	0.56	0.28	0.50	0.33	0.50	0.00	0.11	0.44	1.00	1.00	0.54	0.28
System Analysis	0.59	0.16	0.29	0.14	0.71	0.14	0.57	0.29	0.44	0.31	0.50	0.50	1.00	0.00	0.44	0.11	1.00	0.00	0.52	0.22
Data Querying	0.53	0.41	0.71	0.00	0.43	0.43	0.29	0.29	0.50	0.28	0.67	0.17	0.50	0.00	0.44	0.11	1.00	0.00	0.51	0.28
Functional Programming	0.47	0.16	0.43	0.57	0.43	0.43	0.71	0.14	0.47	0.31	0.67	0.50	1.00	0.00	0.56	0.11	1.00	0.00	0.51	0.26

	Backend		Data science		DevOps		Frontend		Full stack		QA		Security		Other		Mobile		Total	
	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH
Testing Levels	0.66	0.34	0.43	0.57	0.43	0.43	0.43	0.00	0.38	0.41	1.00	0.17	0.50	0.50	0.33	0.33	1.00	0.00	0.51	0.35
Responsive Design	0.31	0.19	0.29	0.29	0.57	0.14	0.86	0.29	0.59	0.25	0.50	0.67	1.00	0.00	0.56	0.00	1.00	1.00	0.50	0.23
Test Automation	0.59	0.19	0.43	0.29	0.43	0.43	0.43	0.00	0.41	0.44	0.67	0.17	1.00	0.00	0.33	0.22	1.00	0.00	0.50	0.27
Promoting Inclusivity	0.53	0.44	0.43	0.29	0.43	0.29	0.71	0.29	0.34	0.41	0.67	0.17	0.50	0.00	0.56	0.11	1.00	1.00	0.49	0.35
Cross platform Development	0.34	0.22	0.43	0.29	0.57	0.43	0.86	0.29	0.56	0.41	0.17	0.83	1.00	0.00	0.33	0.22	0.00	0.00	0.47	0.33
Audit Policies	0.47	0.31	0.29	0.43	0.86	0.00	0.14	0.29	0.38	0.38	0.33	0.67	1.00	0.00	0.67	0.22	1.00	0.00	0.46	0.32
Data Generation	0.41	0.19	0.71	0.29	0.29	0.57	0.29	0.29	0.38	0.38	0.67	0.33	1.00	0.00	0.44	0.11	1.00	0.00	0.44	0.28
Infrastructure Security	0.22	0.41	0.43	0.29	0.71	0.14	0.29	0.43	0.56	0.31	0.33	0.67	1.00	0.00	0.56	0.33	0.00	0.00	0.43	0.35
Scripting	0.34	0.28	0.43	0.14	0.43	0.29	0.29	0.14	0.47	0.34	0.83	0.33	1.00	0.00	0.33	0.11	0.00	0.00	0.43	0.26
UI Implementation	0.13	0.28	0.29	0.29	0.14	0.57	0.71	0.14	0.66	0.38	0.50	0.50	1.00	0.00	0.33	0.11	1.00	1.00	0.41	0.32
Data Preparation	0.34	0.22	0.57	0.29	0.57	0.14	0.29	0.14	0.34	0.28	0.33	0.50	0.50	0.00	0.56	0.00	1.00	0.00	0.40	0.22
Linear Programming	0.38	0.28	0.43	0.29	0.57	0.29	0.71	0.14	0.34	0.44	0.17	0.67	0.50	0.50	0.44	0.11	0.00	0.00	0.40	0.33
Data Reporting	0.25	0.25	0.57	0.14	0.43	0.29	0.29	0.14	0.34	0.19	0.67	0.33	0.50	0.00	0.56	0.11	1.00	0.00	0.38	0.20
Data Sourcing	0.31	0.31	0.71	0.29	0.29	0.43	0.29	0.43	0.31	0.22	0.50	0.33	0.50	0.00	0.33	0.00	1.00	0.00	0.36	0.26
Low level Programming	0.44	0.16	0.29	0.29	0.29	0.29	0.43	0.14	0.31	0.28	0.17	0.50	0.50	0.50	0.33	0.11	1.00	1.00	0.36	0.24

	Backend		Data science		DevOps		Frontend		Full stack		QA		Security		Other		Mobile		Total	
	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH
User accessibility	0.22	0.19	0.29	0.29	0.14	0.57	0.71	0.57	0.41	0.31	0.50	0.33	1.00	0.00	0.33	0.22	1.00	0.00	0.36	0.29
Cryptography	0.31	0.38	0.14	0.43	0.00	0.71	0.57	0.14	0.38	0.34	0.33	0.83	1.00	0.00	0.56	0.33	0.00	0.00	0.35	0.39
Data Exploration	0.38	0.19	0.57	0.29	0.43	0.29	0.29	0.43	0.25	0.31	0.33	0.50	0.50	0.00	0.11	0.11	1.00	0.00	0.33	0.26
Penetration (Pen) Testing	0.34	0.28	0.29	0.29	0.29	0.57	0.29	0.29	0.25	0.34	0.50	0.50	0.50	0.50	0.44	0.33	0.00	0.00	0.32	0.34
Data Visualization	0.22	0.34	0.43	0.29	0.57	0.14	0.43	0.43	0.28	0.28	0.33	0.50	0.50	0.00	0.33	0.00	0.00	0.00	0.31	0.28
Prompt Writing	0.34	0.25	0.29	0.29	0.43	0.29	0.43	0.43	0.22	0.34	0.17	0.67	0.00	0.50	0.44	0.00	0.00	0.00	0.30	0.30
Model Maintenance	0.22	0.22	0.71	0.29	0.29	0.43	0.14	0.14	0.22	0.28	0.50	0.50	0.50	0.50	0.22	0.11	0.00	0.00	0.27	0.26
Large Scale Data Analysis	0.22	0.28	0.71	0.00	0.57	0.14	0.29	0.29	0.22	0.31	0.17	0.67	0.00	0.50	0.11	0.22	0.00	0.00	0.26	0.28
AI proofreading	0.28	0.22	0.29	0.29	0.43	0.29	0.29	0.29	0.19	0.31	0.00	0.83	0.00	0.50	0.22	0.11	0.00	0.00	0.23	0.29
Large Language Model (LLM) frameworks	0.22	0.28	0.29	0.29	0.43	0.29	0.43	0.29	0.16	0.31	0.00	0.67	0.00	0.50	0.44	0.00	0.00	0.00	0.23	0.29
AI models	0.16	0.31	0.43	0.14	0.57	0.14	0.14	0.14	0.22	0.19	0.17	0.50	0.00	0.50	0.22	0.11	0.00	0.00	0.22	0.23
AI output validation	0.19	0.28	0.29	0.29	0.57	0.14	0.43	0.43	0.16	0.28	0.17	0.67	0.00	0.50	0.22	0.00	0.00	0.00	0.22	0.28
Machine Learning Operations	0.13	0.31	0.43	0.14	0.43	0.29	0.14	0.14	0.25	0.22	0.00	0.67	0.00	0.50	0.44	0.00	0.00	0.00	0.22	0.25
Model Tuning	0.19	0.28	0.57	0.00	0.57	0.14	0.14	0.14	0.06	0.47	0.33	0.50	0.00	0.50	0.11	0.11	1.00	0.00	0.20	0.30

	Backend		Data science		DevOps		Frontend		Full stack		QA		Security		Other		Mobile		Total	
	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH	Crit	RUH
Statistical ML Modeling	0.19	0.25	0.57	0.00	0.29	0.43	0.14	0.14	0.06	0.41	0.17	0.83	0.00	0.50	0.22	0.11	0.00	0.00	0.17	0.31

Appendix F: Cross-Functional Relevance of AI Skills (Engineering vs. General Workforce)

Skill	Skill Definition	Sub-skills	Sub-skill Definitions	AI Skills: General	AI Skills: Engineers
Working with Data	Examining and interpreting data through querying, visualization, reporting, and statistical/ML modeling to gain insights and make informed decisions. Includes data preparation, sourcing, exploration, and ongoing model tuning and maintenance.	Data Querying	Applies searching and sorting techniques (e.g., joins, views) within databases via a query language (e.g., SQL, Realm, SQLite).		X
		Data Exploration	Understands and applies data exploration concepts and techniques to identify data characteristics, trends, or problems (e.g., computing and interpreting central tendencies, reviewing distributions, finding missing values).	X	X
		Data Preparation	Understands and applies processes for fetching, selecting, cleaning, manipulating, and transforming data to prepare for analysis.	X	X
		Data Reporting	Understands and presents data to stakeholders and other audiences by translating insights and implications from analyses at an appropriate level.	X	X
		Data Sourcing	Understands and/or applies processes for extracting and/or finding data from internal and external sources.	X	X
		Data Privacy	Understands and upholds data privacy principles to ensure the secure and ethical handling of sensitive information.	X	X
		Data Visualization	Understands and produces graphic representations of data using charts, graphs, and maps.	X	X
		Large-Scale Data Analysis	Analyzes, interprets, and gleans insights from large datasets using data analytics tools and methodologies, statistical analysis, and data visualization techniques.		X
AI Building	Designs systems that leverage machine learning and artificial intelligence to solve specific problems or enhance processes.	Statistical Modeling	Applies knowledge of statistical and machine learning techniques to select, develop, or tailor models suited to the research question and available data. Validates and evaluates model performance by assessing key metrics such as accuracy, precision, recall, F1-score, and AUC, ensuring the quality and reliability of AI-generated results through defined benchmarks and statistical measures.		X
		Model Tuning	Understands, selects, and modifies appropriate hyperparameters to improve model fit.		X
		Model Maintenance	Understands and applies monitoring techniques to maintain model performance and handle model concept changes or deterioration (e.g., concept drifting, back testing).		X
		Machine Learning (ML) Operations	Applies knowledge across the full machine learning lifecycle, including data preparation, model training, deployment, monitoring, and maintenance. Ensures seamless integration of machine learning models and data pipelines at scale.		X

Skill	Skill Definition	Sub-skills	Sub-skill Definitions	AI Skills: General	AI Skills: Engineers
AI Literacy	Understanding foundational concepts of artificial intelligence, including its capabilities, limitations, ethical considerations, and core principles of AI. This skill encompasses familiarity with common AI tools and applications, enabling individuals to grasp how AI operates, recognize its potential and boundaries, and critically assess its uses. For technical professionals, this may involve a deeper conceptual understanding of machine learning frameworks and algorithms.	Machine Learning (ML) Fundamentals	Understands core concepts of machine learning, including supervised and unsupervised learning, model evaluation techniques, and basic algorithms such as linear regression and decision trees. This skill encompasses knowledge of the full lifecycle of machine learning projects, including data preparation, model training, deployment, monitoring, and maintenance.		X
		Large Language Model (LLM) Frameworks	Understands the architecture, training processes, and use cases of large language models (LLMs). This skill involves tailoring LLMs for specific applications by analyzing and addressing computational and data requirements, ensuring optimal performance and alignment with desired outcomes. Proficiency requires knowledge of LLM capabilities, limitations, and techniques for fine-tuning and deployment in targeted scenarios.		X
		AI Awareness	Recognizes AI's strengths and weaknesses while maintaining realistic expectations about its current capabilities. This skill includes understanding the limitations of AI systems, while appreciating their potential applications. Proficiency enables informed decision-making and the strategic use of AI in various contexts.	X	X
		Responsible AI	Identifies and addresses ethical and legal issues related to artificial intelligence, including privacy, social responsibility, accountability, and fairness. This skill involves understanding the ethical and legal implications of AI applications and ensuring that AI systems are designed and used in ways that align with organizational values.	X	X

Skill	Skill Definition	Sub-skills	Sub-skill Definitions	AI Skills: General	AI Skills: Engineers
AI Evaluation	Assessing the performance, reliability, and appropriateness of AI systems and their outputs. This includes testing for accuracy, identifying biases, validating outputs for real-world applicability, and ensuring outputs align with intended objectives and ethical standards. For technical professionals, this skill involves applying advanced analytical methods, such as error analysis and context-specific validation, to evaluate the effectiveness, quality, and integrity of AI systems across diverse scenarios.	AI Performance Testing	Uses systematic testing techniques, including retrieval-augmented generation (RAG), semantic checks, and cross-validation using other AI systems, while selecting and interpreting appropriate metrics to optimize AI performance.		x
		AI Output Proofreading	Refines AI-generated content to ensure accuracy, reliability, coherence, and contextual relevance. This involves reviewing and editing text, code, or other outputs to meet desired quality standards, verifying facts, identifying biases, ensuring alignment with objectives, and maintaining integrity by recognizing and addressing AI-produced material.	x	x

Skill	Skill Definition	Sub-skills	Sub-skill Definitions	AI Skills: General	AI Skills: Engineers
AI Application	Utilizing artificial intelligence in practical, hands-on scenarios. This includes skills such as creating AI-driven workflows, writing effective prompts for AI tools, and integrating AI solutions into existing systems. For more technical professionals, this involves building AI models, developing custom solutions, collaborating with AI engineers, and implementing advanced AI or ML techniques to address specific challenges or objectives.	AI Integration	Incorporates machine learning and artificial intelligence capabilities into software systems and workflows. This technical skill includes leveraging APIs, embedding AI agents into scripts, and integrating AI models into codebases to enhance functionality. Proficiency requires integrating AI models into solutions that align with organizational goals while considering technical constraints.		X
		Prompt Engineering	Designs and refines precise, context-aware prompts to optimize the outputs of generative AI systems and large language models. This technical skill involves understanding how AI models interpret input, leveraging model-specific features, and iteratively adjusting prompts to achieve desired responses or behaviors. Proficiency in prompt engineering requires knowledge of model capabilities and token limitations.		X
		AI Tools and Applications	Incorporates AI tools and solutions into everyday tasks to improve efficiency and outcomes. This includes using pre-built AI tools, automating repetitive processes, and integrating AI capabilities into workflows without requiring coding expertise. Uses user-friendly, graphical interface tools (e.g., ChatGPT) to make AI accessible for non-technical users.	X	X
		Prompt Writing	Crafts clear, concise, and effective prompts designed to elicit specific responses or outputs from generative AI systems or large language models. This skill involves understanding how to structure questions or instructions in a way that maximizes the relevance, accuracy, and quality of AI-generated content.	X	X

Appendix G: Full Skill Definitions and Subskills

Skill Category	Skill / Subskill	Definition
Technical	Configuration Management	Understanding and implementing processes and tools to track, control, and document individual components and assets within IT systems and software to ensure consistency, traceability, and reproducibility throughout their lifecycle (e.g., setting up build systems, makefiles, versioning software components, maintaining a comprehensive record of system configurations).
Technical	Release Management	Understands common notions and patterns of revision control (e.g., file revision, history, log, blaming, branching, release, patching, backporting), applies common revision control procedures in specific revision control system (e.g. Git, Perforce, Mercurial, Subversion)
Technical	Engineering Operations	Planning, delivering, and controlling operations to streamline software development. Applying automation, using continuous integration and delivery (CI/CD), and applying deployment strategies and infrastructure as code paradigms, while adhering to security and observability practices.
Technical	Security	Safeguarding software and systems through adherence to secure coding standards, implementation of cryptographic techniques, and ensuring robust infrastructure security measures to protect against potential threats and vulnerabilities.
Technical	Cryptography	Understands and applies techniques for securing data and communication (e.g., encryption).
Technical	Infrastructure Security	Understands or applies cloud infrastructure security concepts (e.g., Vnets, VPN, AWS WAF/network firewalling).
Technical	Secure Coding Standards	Understands secure coding practices and defensive programming techniques such as threat and risk testing.
Technical	Penetration (Pen) Testing	Understands and applies ethical hacking techniques to simulate attacks that expose security vulnerabilities on apps, data networks, and other assets.
Technical	Incident Management	Identifies, responds to, investigates, and resolves security incidents and documents and reviews incident reports or risk assessments.
Technical	Audit Policies	Understands legal and regulatory security requirements and checks for compliance among security controls using security frameworks, forensic tools, and risk assessments.
Technical	Software Architecture	Understands fundamental software structures such as design patterns, microservices, object-oriented design, and code structure.
Technical	Design Patterns	Understands and applies general, reusable solutions to common problems in software design.
Technical	Software Construction	Developing software by implementing user interfaces, managing errors, asynchronous tasks, and memory, utilizing object-oriented principles, algorithms, and APIs, while supporting cross-platform compatibility, scripting, concurrency, event-driven and functional programming, recursion, linear and low-level programming, and ensuring user accessibility.
Technical	Algorithms	Implements software that runs within time and memory constraints.
Technical	Asynchronous operations	Understands and implements code that executes a sequence of operations that do not coincide with any timed event.
Technical	Basic Programming	Understands and/or applies fundamental coding practices and techniques (e.g., assignment, arithmetic expressions, arrays, strings, variables, loops, conditions, input/output, functions).
Technical	Code Review	Systematically examines source code changes to evaluate implementation, identify defects, and ensure adherence to standards and best practices. Provides constructive feedback on code quality and facilitates knowledge sharing among team members.

Skill Category	Skill / Subskill	Definition
Technical	Concurrency	Understands and applies common concurrency patterns (e.g., synchronization primitives, inter-process communication, threads, shared memory), and understands common concurrency issues (e.g., deadlocks, livelocks, starvation, race conditions, distributed systems).
Technical	Cross-platform Development	Develops the frontend layer of web-based applications that allow users to access applications from a browser of their choice.
Technical	Data Structures	Understands and applies data structures across complexity levels (e.g. set, list, queue, stack, dictionary, trees, graphs, priority queue, tabular data).
Technical	Error/Exception handling	Understands and applies common error or exception handling patterns and/or understands concepts of error or exceptions safety.
Technical	Event-driven Programming	Understands common event-driven programming issues (e.g. at least once/at most once delivery, event taxonomy, eventual consistency) and designs and debugs event-driven systems (e.g. event sourcing, event bus, event loop).
Technical	Functional Programming	Builds software by composing pure functions, avoiding shared state, mutable data, and side-effects in a declarative way.
Technical	Leveraging APIs	Understands, consumes, or implements operating system or web APIs and their documentation, and applies common standards (e.g., OpenAPI, POSIX, REST, GraphQL, Oauth, HTTP).
Technical	Linear Programming	Understands the principles of linear optimization and works with application libraries implementing linear optimization algorithms.
Technical	Low-level Programming	Understands designing, implementing, and optimizing software code to run on embedded systems (e.g., mobile hardware, automotive systems, wearable devices, industrial controls, networking equipment).
Technical	Object-oriented Programming	Understands and applies objects, classes, types, encapsulation, inheritance and interfaces to improve maintainability and reduce complexity.
Technical	Recursion	Designs functions or data structures that are self referential.
Technical	Resource Management	Understands and applies issues related to pointer/reference semantics and memory leaks (e.g. sharing, object ownership, garbage collection, NULL pointer/reference, cyclic structures, memory allocation, copy-on-write).
Technical	Responsive Design	Understands and applies responsive design principles for both desktop and mobile websites or applications.
Technical	Scripting	Builds scripts in "glue languages" to integrate and orchestrate existing programs.
Technical	UI Implementation	Understands and/or implements user interface designs (e.g., visual elements, interactions, animations, transitions).
Technical	User Accessibility	Understands and/or applies methods that make websites usable by all people, regardless of disability (e.g., WCAG standards, ARIA, accessilbility patterns).
Technical	Software Maintenance	Continuously managing and improving software code to sustain its functionality, performance, and reliability.
Technical	Code Optimization	Identifies and removes bottlenecks through re-coding and re-architecting to optimize for performance, scaling, memory usage, or other criteria.
Technical	Code Readability	Ensures software that is easy for other developers to understand, read, and reuse (e.g. appropriately uses comments, documentation, code structure, coding convention, self-explanatory naming).
Technical	Debugging	Identifies, diagnoses, reproduces, or removes bugs in code.
Technical	Software Requirements	Collects and communicates functional and non-functional requirements for a software product and translates them into technical designs, while considering user and business perspectives and the technical context .
Technical	Software Testing	Applying software testing techniques, levels, and automation to ensure software functionality, reliability, and quality.

Skill Category	Skill / Subskill	Definition
Technical	Data Generation	Generates sample data for tests.
Technical	Quality Assurance	Reviews, monitors, and audits software product and development activities to verify quality criteria and standards are met.
Technical	Software Design Review	Reviewing or auditing software design, mockups, documentation, architecture, or requirements.
Technical	Software Testing Techniques	Understands and/or implements software testing techniques (e.g., boundary analysis, use cases matrix, equivalence partitioning, use case testing, white/black box testing, feature testing).
Technical	Test Automation	Designs, writes, or implements automated test scenarios including test prerequisites, steps, and results.
Technical	Testing Levels	Understands and/or implements software testing levels (e.g., unit testing, integration testing, system testing, performance testing).
Technical	Systems Architecture	Determining system configurations, monitoring systems, employing architectural principles, and analyzing/debugging systems to ensure robust and efficient software systems.
Technical	System Analysis	Determines how a system should work and how changes in conditions, operations, and the environment will affect outcomes.
Technical	System Debugging	Identifies, diagnoses, reproduces, or removes bugs in distributed systems using strategies such as heuristics (e.g., USE), tests, and model checks.
Technical	System Design	Understands availability, consistency models, cross-system comms, scalability, single points of failure, observability, and related concepts (e.g., microservices).
Technical	System Monitoring	Understands or applies methods for tracking, reviewing, and regulating system performance (e.g., system configuration metadata changes).
Technical	Working with Data	Examining and interpreting data through querying, visualization, reporting, and statistical/ML modeling to gain insights and make informed decisions. Includes data preparation, sourcing, exploration, and ongoing model tuning and maintenance.
Technical	Data Querying	Applies searching and sorting techniques (e.g., joins, views) within databases via a query language (e.g., SQL, Realm, SQLite).
Technical	Data Exploration	Understands and applies data exploration concepts and techniques to identify data characteristics, trends, or problems (e.g., computing and interpreting central tendencies, reviewing distributions, finding missing values).
Technical	Data Preparation	Understands and applies processes for fetching, selecting, cleaning, manipulating, and transforming data to prepare for analysis.
Technical	Data Reporting	Understands and presents data to stakeholders and other audiences by translating insights and implications from analyses at an appropriate level.
Technical	Data Sourcing	Understands and/or applies processes for extracting and/or finding data from internal and external sources.
Technical	Data Privacy	Understands and upholds data privacy principles to ensure the secure and ethical handling of sensitive information.
Technical	Data Visualization	Understands and produces graphic representations of data using charts, graphs, and maps.
Technical	Large-Scale Data Analysis	Analyzes, interprets, and gleans insights from large datasets using data analytics tools and methodologies, statistical analysis, and data visualization techniques.
Technical	AI Literacy	Understanding foundational concepts of artificial intelligence, including its capabilities, limitations, ethical considerations, and core principles of AI. This skill encompasses familiarity with common AI tools and applications, enabling individuals to grasp how AI operates, recognize its potential and boundaries, and critically assess its uses. For technical professionals, this may involve a deeper conceptual understanding of machine learning frameworks and algorithms.

Skill Category	Skill / Subskill	Definition
Technical	Machine Learning (ML) Fundamentals	Understands core concepts of machine learning, including supervised and unsupervised learning, model evaluation techniques, and basic algorithms such as linear regression and decision trees. This skill encompasses knowledge of the full lifecycle of machine learning projects, including data preparation, model training, deployment, monitoring, and maintenance.
Technical	Large Language Model (LLM) Frameworks	Understands the architecture, training processes, and use cases of large language models (LLMs). This skill involves tailoring LLMs for specific applications by analyzing and addressing computational and data requirements, ensuring optimal performance and alignment with desired outcomes. Proficiency requires knowledge of LLM capabilities, limitations, and techniques for fine-tuning and deployment in targeted scenarios.
Technical	AI Awareness	Recognizes AI's strengths and weaknesses while maintaining realistic expectations about its current capabilities. This skill includes understanding the limitations of AI system, while appreciating their potential applications. Proficiency enables informed decision-making and the strategic use of AI in various contexts.
Technical	Responsible AI	Identifies and addresses ethical and legal issues related to artificial intelligence, including privacy, social responsibility, accountability, and fairness. This skill involves understanding the ethical and legal implications of AI applications and ensuring that AI systems are designed and used in ways that align with organizational values.
Technical	AI Evaluation	Assessing the performance, reliability, and appropriateness of AI systems and their outputs. This includes testing for accuracy, identifying biases, validating outputs for real-world applicability, and ensuring outputs align with intended objectives and ethical standards. For technical professionals, this skill involves applying advanced analytical methods, such as error analysis and context-specific validation, to evaluate the effectiveness, quality, and integrity of AI systems across diverse scenarios.
Technical	AI Performance Testing	Uses systematic testing techniques, including retrieval-augmented generation (RAG), semantic checks, and cross-validation using other AI systems, while selecting and interpreting appropriate metrics to optimize AI performance.
Technical	AI Output Proofreading	Refines AI-generated content to ensure accuracy, reliability, coherence, and contextual relevance. This involves reviewing and editing text, code, or other outputs to meet desired quality standards, verifying facts, identifying biases, ensuring alignment with objectives, and maintaining integrity by recognizing and addressing AI-produced material.
Technical	AI Application	Utilizing artificial intelligence in practical, hands-on scenarios. This includes skills such as creating AI-driven workflows, writing effective prompts for AI tools, and integrating AI solutions into existing systems. For more technical professionals, this involves building AI models, developing custom solutions, collaborating with AI engineers, and implementing advanced AI or ML techniques to address specific challenges or objectives.
Technical	AI Integration	Incorporates machine learning and artificial intelligence capabilities into software systems and workflows. This technical skill includes leveraging APIs, embedding AI agents into scripts, and integrating AI models into codebases to enhance functionality. Proficiency requires integrating AI models into solutions that align with organizational goals while considering technical constraints.
Technical	Prompt Engineering	Designs and refines precise, context-aware prompts to optimize the outputs of generative AI systems and large language models. This technical skill involves understanding how AI models interpret input, leveraging model-specific features, and iteratively adjusting prompts to achieve desired responses or behaviors. Proficiency in prompt engineering requires knowledge of model capabilities and token limitations.
Technical	AI Tools and Applications	Incorporates AI tools and solutions into everyday tasks to improve efficiency and outcomes. This includes using pre-built AI tools, automating repetitive processes, and integrating AI capabilities into workflows without requiring coding expertise. Uses user-friendly, graphical interface tools (e.g., ChatGPT) to make AI accessible for non-technical users.
Technical	Prompt Writing	Crafts clear, concise, and effective prompts designed to elicit specific responses or outputs from generative AI systems or large language models. This skill involves understanding how to structure questions or instructions in a way that maximizes the relevance, accuracy, and quality of AI-generated content.
Technical	AI Building	Designs systems that leverage machine learning and artificial intelligence to solve specific problems or enhance processes.

Skill Category	Skill / Subskill	Definition
Technical	Statistical Modeling	Applies knowledge of statistical and machine learning techniques to select, develop, or tailor models suited to the research question and available data. Validates and evaluates model performance by assessing key metrics such as accuracy, precision, recall, F1-score, and AUC, ensuring the quality and reliability of AI-generated results through defined benchmarks and statistical measures.
Technical	Model Tuning	Understands, selects, and modifies appropriate hyperparameters to improve model fit.
Technical	Model Maintenance	Understands and applies monitoring techniques to maintain model performance and handle model concept changes or deterioration (e.g., concept drifting, back testing).
Technical	Machine Learning (ML) Operations	Applies knowledge across the full machine learning lifecycle, including data preparation, model training, deployment, monitoring, and maintenance. Ensures seamless integration of machine learning models and data pipelines at scale.
Problem-Solving	Systems Thinking	Understanding how different parts of a system interrelate and how changes in one part of the system affect the whole. Taking into account the user's perspective and the broader context in which the system and its components will function together to fulfill its intended objectives and satisfy user requirements.
Problem-Solving	Creative Thinking	Generating original ideas, finding unique solutions to problems, and combining existing ideas in new ways.
Problem-Solving	Critical Thinking	Evaluating information and arguments through reasoned analysis to make well-informed decisions, solve problems, and facilitate learning. It involves skepticism, logical reasoning, and the ability to question assumptions and biases.
Problem-Solving	Computational Thinking	Formulating and solving problems using strategies common to computer science and programming. Breaking down complex problems into smaller, more manageable components and expressing solutions in a language that a computer can execute.
Supporting	Teamwork	Working effectively with others on shared tasks (e.g., co-creating, pair programming). Resolving conflicts through facilitation and reconciliation. Actively participating in team discussions, assisting other members in addressing problems, and emphasizing the team's collective interests. Readily sharing information and knowledge with the team. Acknowledging and advocating for others' contributions.
Supporting	Communication	Conveying information, ideas, thoughts, or feelings effectively and clearly to others through various means, such as verbal, written, non-verbal, or digital channels. Listening actively, expressing oneself coherently, adapting communication style to different audiences, and comprehending and interpreting information from others accurately.
Supporting	Developing Others	Developing, guiding, teaching, coaching, and mentoring others toward achieving personal and organizational goals. Delivering effective feedback on an ongoing basis to improve performance.
Supporting	Promoting Inclusivity	Respecting, appreciating, and encouraging contributions and participation from all individuals, including those with diverse backgrounds and cultures. Fostering an inclusive mindset that evaluates perspectives, practices, and products from different cultural perspectives.
Supporting	Learning and Adapting	Responding to volatility, uncertainty, complexity, and ambiguity (VUCA) with curiosity and flexibility. Seeking new ideas and approaches by applying learning to work and discarding ineffective methods. Adapting plans, priorities, and goals in response to difficult conditions, tight deadlines, obstacles, or setbacks.
Supporting	Taking Ownership	Taking responsibility for one's decisions and behaviors, holding oneself accountable for meeting work objectives within set timeframes, and taking action without waiting for direction.
Supporting	Behaving Ethically	Exhibiting integrity, trustworthiness, honesty, and fairness in one's decisions and actions. Demonstrating global, social, intellectual, and technological responsibility.
Supporting	Achieving Results	Setting goals for personal and group accomplishment, monitoring progress toward goal attainment, and working to meet or exceed goals. Persisting to accomplish tasks. Being outcome-driven versus output-driven.