IT SKILL STANDARDS 2020 AND BEYOND

“Software Development”

Job Cluster

Acknowledgements

The development and publication of these skill standards has been a joint and collaborative effort between business and industry representatives and the education community. We are grateful to the industry personnel who participated in the development and validation process. Industry subject matter experts, technical executives, supervisors and technicians donated their time and effort to assure the relevancy of the standards 12 to 36 months into the future.

We gratefully acknowledge funding from the National Science Foundation and the leadership by the team on the IT Skill Standards 2020 and Beyond grant, based at Collin College.

Our leaders are strategically divided into Central, Western, and Eastern teams.

Central

Dr. Ann Beheler, Principal Investigator

Christina Titus, Program Director

Deborah Roberts, Co-Principal Investigator

Helen Sullivan, Senior Staff

West Coast

Terryll Bailey, Co-Principal Investigator

Dr. Suzanne Ames, Co-Principal Investigator

East Coast

Peter Maritato, Co-Principal Investigator

Gordon Snyder, Senior Staff

This material is based upon work supported by the National Science Foundation under Grant No. 1838535. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

itskillstandards.org

Software Development

The definition for Software Development as developed by approximately 100 Thought Leaders (mostly Chief Technology Officers and Chief Information Officers) through three meetings and follow-up surveys to gain consensus is:

> Software development and engineering includes the research, design, secure creation, delivery and quality assurance/testing of computer software and applications including mobile. Additionally, web development can range from developing a simple single static page of plain text to complex web-based internet applications (web apps), and social network services. This definition was adapted from Wikipedia with input from national IT Thought Leaders.

This packet includes…

Job skills as developed by subject matter experts (SMEs) via multiple synchronous meetings (Page 5).

The tasks, knowledge, skills and abilities (KSAs) were developed with a focus 12 to 36 months in the future for an entry-level employee working in that specific cluster.

More specific definitions can be found within the KSA list.

The average was calculated from the subject matter expert votes.

- A vote of "4" indicated the item must be covered in the curriculum.
- A vote of "3" indicated the item should be covered in the curriculum.
- A vote of "2" indicated that it would be nice for the item to be covered in the curriculum.
- A vote of "1" indicated the item should not be covered in the curriculum.

Employability Skills as developed by SMEs via multiple synchronous meetings (Page 9).

Employability competencies are essential for every IT job and are based on what the work requires. SMEs were offered three clearly-defined “levels of proficiency” for each employability skill. The proficiency scale is defined as Level 1 – basic: Level 2- intermediate; and Level 3 - advanced. The levels are cumulative, so a “Level 3” assumes the employee can perform all characteristics of “Level 1” and “Level 2.”

For each employability skill, SMEs selected the competency level that best aligned with what would be expected from an entry-level worker for the job cluster in question.

Key Performance Indicators (KPIs) as developed by SMEs (Page 10).

Key Performance Indicators answer the question, “How do we know when a task is performed well?”

A search was performed to locate validated/verified KPIs for technician level work in IT fields. Sources included the Texas Skill Standards System, National Skill Standards Board, National Institute of Standards and Technology and other sources. The identified KPIs were then cross-referenced to the tasks for the ITSS 2020 job clusters. They were reviewed and revised by a group of the same subject matter experts who developed the tasks and KSAs for the cluster in a structured, facilitated verification session.

Student Learning Outcomes (SLOs) as identified by educators attending the KSA meetings (Page 12).

The SLOs are for use in the creation of curriculum to help define what the students will know and be able to demonstrate. Each of these SLOs can be observed, measured, and demonstrated.

Degree Expectations as identified by educators (Page 16).

A pool of 20 community college and four-year university faculty members from across the country were asked to categorize each knowledge, skill, ability, and task below. The question posed to them: would these KSA+Ts be reasonably included in a two-year AAS program, a four-year Bachelor's program, both, or neither? These results provide another tool for educators to use in assessing how to best incorporate each knowledge, skill, ability, and task.

	Software Development KSAs	
		AVG
Tasks		
Analysis and Design		
T-1	Identify, document, and effectively communicate security concerns and/or threat vulnerabilities.	2.8
T-2	Analyze information to determine, recommend, and plan development and installation of a new system or modification of an existing system.	2.7
Programming		
T-3	Develop code to read and write files.	3.7
T-4	Create webpages using data from a database.	3.3
T-5	Create applications such as Servlets that send HTML pages to Internet clients.	3.1
T-6	Write and debug effective code using various scripting languages.	3.5
T-7	Assist with development on multiple platforms (e.g., Linux, Windows, AppleOS, etc.).	3.0
T-8	Design, develop, and validate stable, robust, secure, and efficient code following industry best practices.	3.5
T-9	Develop secure code and error handling.	3.6
T-10	Develop cross platform applications targeted for an OS or platform other than the development environment.	2.5
T-11	Develop applications that run on multiple browsers.	3.5
T-12	Design, create, manage, and evaluate Apps.	3.2
T-13	Manipulate the objects contained in the Document Object Model (DOM).	2.8
T-14	Demonstrate familiarity with at least one current IDE and other developer productivity tools.	3.6
T-15	Identify, evaluate, and apply efficient algorithms and data structures (e.g., sorting, multithreading).	3.2
T-16	Apply SDLC (software development lifecycle) industry practices (e.g., Agile, waterfall, scrum, etc.).	3.2
T-17	Assist in designing countermeasures and mitigations against potential exploitations of programming language weaknesses and vulnerabilities in system and elements.	2.6
T-18	Apply secure code documentation in accordance with corporate policy to ensure safety of how code is implemented or processed for user access and security access to code that govern software driven apparatus.	3.4
T-19	Compile and write documentation of existing software program development and subsequent revisions, inserting comments in the coded instructions so others can understand the program.	3.5
T-20	Identify and leverage the enterprise-wide version control system while designing and developing secure applications.	3.4
T-21	Collaborate with a wide range of technical professionals, in person and virtually, using tools and strategies that support cooperative software development practices.	3.7
Testing		
T-22	Conduct trial runs of programs and software applications to ensure that the desired information is produced and instructions and security levels are correct.	3.3
T-23	Test and evaluate any software code/processes you developed (unit testing).	3.8
T-24	Utilize software testing tools to implement various test strategies.	3.3
T-25	Assist in developing software system testing and validation procedures, programming, and documentation.	3.2
T-26	Correct errors by making appropriate changes and rechecking the program to ensure that desired results are produced.	3.7
T-27	Apply coding and testing standards, security testing tools including "'fuzzing" static-analysis code scanning tools, and conduct code reviews.	3.3
Implementation		
T-28	Determine system performance against standards and follow appropriate action plan when issues arise.	2.8
T-29	Implement and properly document software patches and report any software security issues that would leave software vulnerable.	3.4
T-30	Modify existing software to correct errors, adapt it to new hardware, or upgrade interfaces and improve performance.	3.3
T-31	Contribute presentation materials and communicated effectively in a team meeting.	3.4

T-32	Communicate with customers or other departments on project status, proposals, or technical issues, such as software system design or maintenance, including both oral and written communication.	2.9
T-33	Contribute to team, follow directives from designers and engineers related to software design and implementation.	3.6
Knowledge Knowledge focuses on the understanding of concepts. It is theoretical. An individual may have an understanding of a topic or tool or some textbook knowledge of it but have no experience applying it. For example, someone might have read hundreds of articles on health and nutrition, many of them in scientific journals, but that doesn't make that person qualified to dispense advice on nutrition.		
K-1	Knowledge of software development models (e.g., Waterfall Model, Spiral Model).	3.3
K-2	Knowledge of system design tools, methods, and techniques, including automated systems analysis and design tools.	3.3
K-3	Knowledge of effective software debugging principles.	3.6
K-4	Knowledge of computer programming languages and principles in general.	3.7
K-5	Knowledge of web services (e.g., service-oriented architecture, REST, and web service description language).	3.3
K-6	Knowledge of visual representations of a program or system (e.g., UML, etc.).	2.5
K-7	Knowledge of how programs communicate across networks using asynchronous and synchronous techniques (when to use and why).	2.9
K-8	Knowledge of Software Integration Management Systems – how industry documents final product builds to show all of the elements that have changes and checks those that have not changed.	2.7
K-9	Knowledge of event handling in a GUI.	3.3
K-10	Knowledge of Regression Testing Development – how to test software using software.	3.3
K-11	Knowledge of the appropriate use of cookies.	3.2
K-12	Knowledge of how applets differ from applications in terms of program form, operating context, and how they are started.	3.1
K-13	Knowledge of two or more operating systems that are current industry standards (e.g., Linux, Windows Apple OS).	3.4
K-14	Knowledge of error handling constructs.	3.5
K-15	Knowledge of the differences between client-side scripting and server-side scripting.	3.5
K-16	Knowledge of common program architectures (e.g., standalone, three-tier, web-based, cloud-based, serverless, microservice).	3.3
K-17	Knowledge of the local development cycle (e.g., build, deploy, test, debug).	3.7
K-18	Knowledge of server software patterns, messaging patterns both async and synch.	3.3
K-19	Knowledge of database integration/management software.	3.1
K-20	Knowledge of AI and ML methods and algorithms.	2.7
K-21	Knowledge of software collaboration tools (e.g., version control, bug tracking, continuous integration).	3.6
K-22	Knowledge of the limits vs actual process of continuous integration and production deployment practices of devsecops/devnetsecops.	3.0
K-23	Knowledge of cybersecurity and privacy principles and methods that apply to software development.	3.3
K-24	Knowledge of system and application security threats and vulnerabilities (e.g., buffer overflow, mobile code, cross-site scripting, Procedural Language/Structured Query Language [PL/SQL] and injections, race conditions, covert channel, replay, return-oriented attacks, malicious code).	3.3
K-25	Knowledge of code security (e.g., hashing, encryption, cryptography, threat modeling).	3.1
K-26	Knowledge of Privacy Impact Assessments in terms of privacy and identity management.	2.7
K-27	Knowledge of cyber threats and vulnerabilities.	3.3
K-28	Knowledge of software related information technology (IT) security principles and methods (e.g., modularization, layering, abstraction, data hiding, simplicity/minimization).	3.2
K-29	Awareness of standards such as PCI, PHI, and GDPR.	2.8
K-30	Knowledge of basic security practices including threats and vulnerabilities that may arise from interactions with other systems, external and legacy code.	3.5
K-31	Knowledge of computer network fundamentals (e.g., TCP/IP, HTTPS, ports, firewall, LAN/WAN, etc.) and network security methodologies.	3.3

K-32	Knowledge of implementation and utilization of cloud services including deployment (e.g., AWS, Microsoft Azure).	3.1
K-33	Awareness of cloud computing concepts (e.g., IoT, edge computing).	2.5
K-34	Knowledge of software development and implementation for communicating and gathering data from IoT devices.	2.6
K-35	Knowledge of the difference between AI and ML.	2.9
K-36	Awareness of current and specialized AI and ML tools and their application to business problems.	2.7
K-37	Conceptual knowledge of PKI.	2.9
K-38	Knowledge of DevSecOps.	2.9
K-39	Knowledge of structured and unstructured data sources.	3.3
K-40	Knowledge of open source software and risks involved.	3.4
K-41	Knowledge of ethics and its application to software development.	3.4
K-42	Knowledge of best practices for Design/UI/UX/accessibility as applied to software development.	3.2
K-43	Knowledge of lifecycle development/steady state/end of life.	3.1
K-44	Knowledge of mobile application development.	2.8
K-45	Knowledge of how to protect data privacy through code.	3.1
K-46	Knowledge of process flow and how the upgrade/implementation of software is accomplished through definitive understanding of team collaboration in DevOps, End of Life Cycle, and including importance of foundational security.	2.7
K-47	Knowledge of performing integrated quality assurance testing for security functionality and resiliency attack.	2.6
K-48	Knowledge of how to identify security implications in the software acceptance phase including completion criteria, risk acceptance and documentation, common criteria, and methods of independent testing and report concerns to IT/software team.	2.6
K-49	Knowledge of applications with public keying by leveraging existing public key infrastructure (PKI) libraries and incorporating certificate management and encryption functionalities when appropriate.	2.5
K-50	Knowledge of how to identify and leverage the enterprise-wide security services while designing and developing secure applications (e.g., Enterprise PKI, Federated Identity server) when appropriate.	2.5
K-51	Knowledge of how to identify and analyze user needs and use needs to establish a plan in the selection, creation, evaluation, implementation and administration of information technology systems.	2.4
K-52	Knowledge of security requirements into application design elements including documenting the elements of the software attack surfaces, conducting threat modeling, and defining any specific security criteria.	2.4
K-53	Knowledge of architecture patterns and when to use them to build applications.	2.8
K-54	Knowledge of algorithms and data structures (e.g., big-O, linked lists, hash maps, sorting, etc.).	2.9
K-55	Knowledge of Binary search tree and how binary search works.	2.8
K-56	Knowledge of Hash maps.	2.8
Skills The capabilities or proficiencies developed through training or hands-on experience. Skills are the practical application of theoretical knowledge. Someone can take a course on investing in financial futures, and therefore has knowledge of it. But getting experience in trading these instruments adds skills.		
S-1	Skill in using built-in functions as well as skill in creating custom functions, subroutines, and procedures within software using scripting languages.	3.3
S-2	Skill in integrating standard object model components with server pages in support of the User Experience.	2.9
S-3	Skill in conducting software debugging.	3.6
S-4	Skill in creating programs that validate and process multiple inputs including command line arguments, environmental variables, and input streams.	3.5
S-5	Skill in writing code in current programming languages and frameworks.	3.7

S-6	Skill in developing applications that can log and handle errors, exceptions, and application faults and logging.	3.3
S-7	Skill in applying root cause analysis (RCA) techniques to solving software/customer issues.	3.2
S-8	Skill in the live production environment (e.g., monitoring, logging, alerting, remote debugging).	3.2
S-9	Skill in using electronic mail software (e.g., Google Gmail; IBM Notes Hot technology; Microsoft Exchange Server Hot technology; Microsoft Outlook Hot technology).	3.4
S-10	Skill in using graphical user interface development software (e.g., Graphical user interface GUI builder software; Graphical user interface GUI design software; Salesforce Visualforce Hot technology).	3.1
S-11	Skill in using object or component-oriented development software (e.g., C++ Hot technology; Document Object Model DOM Scripting; Python Hot technology; Simple API for XML SAX).	3.2
S-12	Skill in creating classes that use inheritance aspects of the object-oriented paradigm.	3.2
S-13	Skill in using, incorporating and utilizing cookies.	2.8
S-14	Skill in implementing programs that use local or remote databases with standard protocols.	3.2
S-15	Skill in using a scripting language on the server side and the client side of a distributed program.	3.2
S-16	Skill in evaluating and reporting software needs, constraints, analysis for application-specific concerns.	2.9
S-17	Skill in implementing levels of security in distributed software applications and applets.	2.8
S-18	Skill in deploying secure software according to secure software deployment methodologies, tools, and practices (e.g., PCI,GDPR, HIPPA, CCPA).	3.1
S-19	Skill in mobile application development.	2.7
S-20	Skills such as time management, risk management.	3.3
S-21	Skill in incorporating user experience feedback into software.	3.1
S-22	Skill in integrating third party open source resources into software including minimizing risk.	3.1
S-23	Skill in learning new and/or industry standard tools involved in the development of software.	3.6
Abilities Abilities have historically been used to describe the innate traits or talents that a person brings to a task or situation. Many people can learn to negotiate competently by acquiring knowledge about it and practicing the skills it requires. A few are brilliant negotiators because they have the innate ability to persuade. In reality, abilities may be included under skills or may be separated out.		
A-1	Ability to both mentor and be mentored; provide critical feedback as well as accept critical feedback two-way.	3.5
A-2	Ability to comprehend and execute both written and oral instructions by asking clarifying questions.	3.8
A-3	Ability to effectively communicate technical concepts and constraints in written and oral form to technical team members, stakeholders.	3.6
A-4	Ability to work effectively in multi-disciplinary teams to apply information technology in support of organizational goals.	3.6
A-5	Ability to produce technical content for tech writers.	2.9
A-6	Ability to manage your own software development project activities and deliverables in a timely and efficient manner.	3.5
A-7	Ability to work on team projects and demonstrate critical thinking, teamwork, oral communications, inter-cultural appreciation, and technical and information literacy skills.	3.9
A-8	Ability to research and be able to find other sources to answer the problem.	3.7
A-9	Ability to engage with users and understand their user experience.	3.3
A-10	Ability to draw on prior knowledge and experience in a new situation.	3.7

Software Development Employability Skills

Workplace Professionalism & Work Ethics	Level 1 - Employee learns expectations of workplace environment (professional behavior and ethics) and adheres to practices with some guidance. Level 2 - Employee exhibits sound professionalism, judgment, and integrity and accepts responsibility for own behavior. Employee exhibits these qualities without guidance but occasionally refers to policies as needed.
Written Communication	Level 1 - Employee understands written instructions and executes tasks with guidance and feedback from supervisor. Employee clearly communicates concepts in writing. Level 2 - Employee comprehends and executes written instructions with minimal guidance. Employee composes well-organized written documents.
Oral Communication	Level 1 - Employee understands oral instructions and executes tasks with guidance and feedback from supervisor. Employee communicates concepts orally while clarifying for meaning. Employee develops listening skills. Level 2 - Employee comprehends and executes oral instructions with minimal guidance and exhibits good listening skills. Employee clarifies for meaning without needing prompting from supervisor.
Teamwork	Level 1 - With guidance and feedback from supervisor, employee obeys team rules and understands team member roles. Employee actively participates in team activities, volunteers for special tasks, and establishes rapport with co-workers. Level 2 - Employee demonstrates commitment, enthusiasm and supports team members. Employee follows up on assigned tasks and leads by example.
Problem Solving & Critical Thinking	Level 1 - Employee identifies the problem and relevant facts and principles with guidance and feedback from supervisor. Employee summarizes existing ideas and demonstrates creative thinking process while problem solving. Level 2 - With minimal supervision, employee analyzes underlying causes, considers risks and implications, and uses logic to draw conclusions. Employee applies rules and principles to processes and recommends solutions.
Organization and Planning	Level 1 - Employee prepares schedule for self, monitors and adjusts task sequence, and analyzes work assignments with guidance from supervisor. Level 2 - Employee manages timelines and recommends timeline adjustments. Employee escalates timeline-impacting issues as appropriate.
Adaptability and Flexibility	Level 1 - With guidance and feedback from supervisor, employee is able to adjust ways of doing work based on changing dynamics. Working under pressure is difficult, but employee makes it through the project with guidance and oversight. Level 2 - Employee makes inquiries of co-workers regarding possible changes needed in ways of doing work and adapts accordingly. Observes co-workers increasing work productivity under pressure and follows their lead.
Initiative	Level 1 - Employee finishes a step in a project and waits for direction before going on to the next step. Level 2 - Employee finishes multiple steps in a project and appropriately begins working on the next step without being asked.
Accuracy	Level 1 - Employee makes mistakes routinely but is committed to learning to adjust work habits to prevent them in the future. Level 2 - Employee occasionally makes mistakes but quickly makes adjustments to work habits to avoid making the same mistake twice.
Cultural Competence	Level 1 - Employee is inexperienced with working with diverse teams. With support and guidance and getting to know team members, employee develops working relationships. Level 2 - Employee is committed to working with diverse teams but struggles when differences arise. Employee identifies those challenges and works with colleagues to find ways to work effectively.
Self and Career Development	Level 1 - Employee requires feedback and direction from supervisor regarding improvement needed in professional and technical skills. Employee follows through with skills development with monitoring by supervisor. Level 2 - Employee builds upon self-assessment experience and can develop a professional and technical skills improvement plan in conjunction with supervisor. Employee completes development plan without prompting from supervisor.

Software Development Key Performance Indicators

For the entry-level employee, all tasks are typically done under supervision for much of the first year and then with some independence with verification after the employee has more experience. All tasks are done according to company guidelines.

	Tasks	Key Performance Indicators
Analysis & Design		
T-1	Identify, document, and effectively communicate security concerns and/or threat vulnerabilities.	Requirements are properly understood, interpreted, and evaluated, and conflicting requirements are identified and resolved. Time, technology, and resource constraints are defined, alternatives are presented, and risk analysis and contingency plans are implemented. Security requirements are consistent with company standards and all applicable laws and regulations. Analysis of new and existing software security concerns and/or threat vulnerabilities are provided to IT/software team to guide development/modification of a security application.
T-2	Analyze information to determine, recommend, and plan development and installation of a new system or modification of an existing system.	
Programming		
T-3	Develop code to read and write files.	Code is developed and documented using efficient software design processes. Links between web applications and associated databases are properly established. Appropriate debugging tools are used in an efficient manner. High-quality software of multiple types is produced that meets or exceeds customer expectations, follows industry best practices, and is completed within engineering time and cost estimates. Application, programming, or communication errors and security vulnerabilities are correctly anticipated, detected, and resolved. Authoring, modifying, compiling, deploying, and debugging of software are completed in a thorough and efficient manner. Programs are written in the most efficient way, and data is organized in such a way that it can be updated, deleted, retrieved efficiently, and securely protected. Software is deployed in accordance with secure software deployment methodologies, tools, documentation, and other practices. Implementing solutions known threats known to produce a reduction in threats and vulnerabilities. Documentation is clear and complete, including consistent use of enterprise-wide version control. The appropriate Integrated Development Environment is used in code creation. SDLC industry practices, as specified by the company, are consistently followed. Cooperative software development practices are utilized.
T-4	Create webpages using data from a database.	
T-5	Create applications such as Servlets that send HTML pages to Internet clients.	
T-6	Write and debug effective code using various scripting languages.	
T-7	Assist with development on multiple platforms (e.g., Linux, Windows, AppleOS, etc.).	
T-8	Design, develop and validate stable, robust, secure, and efficient code following industry best practices.	
T-9	Develop secure code and error handling.	
T-10	Develop cross platform applications targeted for an OS or platform other than the development environment.	
T-11	Develop applications that run on multiple browsers.	
T-12	Design, create, manage, and evaluate Apps.	
T-13	Manipulate the objects contained in the Document Object Model (DOM).	
T-14	Demonstrate familiarity with at least one current IDE and other developer productivity tools.	
T-15	Identify, evaluate, and apply efficient algorithms and data structures (e.g., sorting, multithreading).	
T-16	Apply SDLC (software development lifecycle) industry practices (e.g., Agile, waterfall, scrum, etc.).	
T-17	Assist in designing countermeasures and mitigations against potential exploitations of programming language weaknesses and vulnerabilities in system and elements.	
T-18	Apply secure code documentation in accordance with corporate policy to ensure safety of how code is implemented or processed for user access and security access to code that govern software driven apparatus.	
T-19	Compile and write documentation of existing software program development and subsequent revisions, inserting comments in the coded instructions so others can understand the program.	
T-20	Identify and leverage the enterprise-wide version control system while designing and developing secure applications.	
T-21	Collaborate with a wide range of technical professionals, in person and virtually, using tools and strategies that support cooperative software development practices.	

	Testing	
T-22	Conduct trial runs of programs and software applications to ensure that the desired information is produced and instructions and security levels are correct.	Unit testing is accomplished using standard testing procedures, and testing on each unit is repeated until the unit is free of errors. Appropriate software testing tools are used. Testing identifies errors, gaps, or missing requirements and results in reliability, security, and high performance. Errors identified during testing are corrected and code is retested until no errors are identified. A systematic testing program is implemented that is relevant to application and test requirements and is in compliance with legal requirements, policies, procedures, and customer requirements. Code reviews are performed in a regular and timely manner.
T-23	Test and evaluate any software code/processes you developed (unit testing).	
T-24	Utilize software testing tools to implement various test strategies.	
T-25	Assist in developing software system testing and validation procedures, programming, and documentation.	
T-26	Correct errors by making appropriate changes and rechecking the program to ensure that desired results are produced.	
T-27	Apply coding and testing standards, security testing tools including "'fuzzing" static-analysis code scanning tools, and conduct code reviews.	
	Implementation	
T-28	Determine system performance against standards and follow appropriate action plan when issues arise.	Software upgrades and patches are applied with minimal service disruptions to clients/users in a timely manner. Software performance meets design specs and client/user requirements. Software is modified on an ongoing basis to adapt to hardware and software changes. Recommendations based on customer input and analysis of system data are presented to appropriate personnel. Client/users are informed regarding requirements and technology. Effective presentations are used to communicate both internally and externally. Team members collaborate and follow the design and implementation guidelines provided.
T-29	Implement and properly document software patches and report any software security issues that would leave software vulnerable.	
T-30	Modify existing software to correct errors, adapt it to new hardware, or upgrade interfaces and improve performance.	
T-31	Contribute presentation materials and communicated effectively in a team meeting.	
T-32	Communicate with customers or other departments on project status, proposals, or technical issues, such as software system design or maintenance, including both oral and written communication.	
T-33	Contribute to team, follow directives from designers and engineers related to software design and implementation.	

Software Development Student Learning Outcomes		
	Knowledge	**Student Learning Outcomes**
K-22	Knowledge of the limits vs actual process of continuous integration and production deployment practices of devsecops/devnetsecops.	Explain the process of integration, production, and deployment of software development life cycle.
K-36	Awareness of current and specialized AI and ML tools and their application to business problems.	Stay informed regarding current and specialized AI and ML tools to solve business problems.
K-16	Knowledge of common program architectures (e.g., standalone, three-tier, web-based, cloud-based, serverless, microservice).	Describe common application architectures.
K-32	Knowledge of implementation and utilization of cloud services, including deployment (e.g., AWS, Microsoft Azure).	Identify the different cloud services and how they are different in their implementation process.
K-5	Knowledge of web services (e.g., service-oriented architecture, Simple Object Access Protocol, and web service description language).	Discuss common services, practices, and protocols used in the development of web services.
K-11	Knowledge of the appropriate use of cookies.	Explain the appropriate usage of cookies.
K-23	Knowledge of cybersecurity and privacy principles and methods that apply to software development.	Explain information security principles and fundamental methods that apply to software development.
K-27	Knowledge of cyber threats and vulnerabilities.	Discuss the potential threats and vulnerabilities that may arise from basic security practices when interacting with other systems, and external and legacy code.
K-30	Knowledge of basic security practices including threats and vulnerabilities that may arise from interactions with other systems,	
K-29	Awareness of standards such as PCI, PHI, and GDPR.	Be aware of laws, regulations, and standards related to cybersecurity and privacy globally.
K-25	Knowledge of code security (e.g., hashing, encryption, cryptography, threat modeling).	Describe secure coding algorithms such as hashing, encryption, cryptography in general, and PKI.
K-37	Conceptual knowledge of PKI.	
K-34	Knowledge of software development and implementation for communicating and gathering data from IoT devices.	Explain how to develop and implement software programs for the purpose of communicating and gathering data from IoT devices.
K-39	Knowledge of structured and unstructured data sources.	Describe how to access data from both structured and unstructured sources.
K-19	Knowledge of database integration/management software.	Discuss database integration tools and techniques for software management.
K-3	Knowledge of effective software debugging principles.	Demonstrate tools and techniques used to debug software applications.
K-18	Knowledge of server software patterns, messaging patterns both async and synch.	Explain async and synch in server software patterns and messaging patterns.
K-42	Knowledge of best practices for Design/UI/UX/accessibility as applied to software development.	Define UI, UX, and accessibility and demonstrate best practices for incorporating them in a software system design.
K-15	Knowledge of the differences between client-side scripting and server-side scripting.	Explain the differences between client-side and server-side scripting in software development.
K-46	Knowledge of process flow and how the upgrade/implementation of software is accomplished through definitive understanding of team collaboration in DevOps, End of Life Cycle, and including importance of foundational security.	Discuss DevSecOps as it relates to DevOps and the secure software development life cycle.
K-6	Knowledge of visual representations of a program or system (e.g., UML, etc.).	Explain the use of visual tools such as UML to represent software design.
K-8	Knowledge of Software Integration Management Systems – how industry documents final product builds to show all of the elements that have changes and checks those that have not changed.	Document software changes in the Software Integration Management Systems.
K-41	Knowledge of ethics and its application to software development.	Explain the importance of ethics and how it is applied to software development.
K-7	Knowledge of how programs communicate across the Internet using conventions such as Remote Method Invocation.	Explain how programs communicate across the Internet using conventions such as Remote Method Invocation.
K-14	Knowledge of error handling constructs.	Discuss how constructs are used to handle errors in software.
K-4	Knowledge of computer programming languages and principles in general.	List commonly used programming languages and general concepts that are common to these languages.

K-12	Knowledge of how applets differ from applications in terms of program form, operating context, and how they are started.	Explain the differences between applets and applications.
K-44	Knowledge of mobile application development.	Describe tools, techniques, and frameworks used for mobile application development.
K-31	Knowledge of computer network fundamentals (e.g., TCP/IP, HTTPS, ports, firewall, LAN/WAN etc.) and network security methodologies.	Explain computer network fundamentals and security methodologies such as TCP/IP, HTTPS, ports, firewall, and LAN/WAN.
K-13	Knowledge of two or more operating systems that are current industry standards (e.g., Linux, Windows Apple OS).	Discuss common operating systems used for software applications.
K-1	Knowledge of software development models (e.g., Waterfall Model, Spiral Model).	Explain the different types of software development models.
K-2	Knowledge of system design tools, methods, and techniques, including automated systems analysis and design tools.	Describe tools, methods, and techniques used for software analysis and design.
K-17	Knowledge of the local development cycle (e.g., build, deploy, test, debug).	List and discuss the phases of the software development lifecycle.
K-43	Knowledge of lifecycle development/steady state/end of life.	
K-10	Knowledge of Regression Testing Development – how to test software using software.	Explain how to test software with Regression Testing Development.
K-9	Knowledge of event handling in a GUI.	Describe how event-handling is implemented in a GUI (graphical user interface).
K-24	Knowledge of system and application security threats and vulnerabilities (e.g. buffer overflow, mobile code, cross-site scripting, Procedural Language/Structured Query Language [PL/SQL] and injections, race conditions, covert channel, replay, return-oriented attacks, malicious code).	Discuss software application and system security threats and vulnerabilities.
K-28	Knowledge of software related information technology (IT) security principles and methods (e.g., modularization, layering, abstraction, data hiding, simplicity/minimization).	Classify the application of secure coding principles and methods.
K-38	Knowledge of DevSecOps.	Describe DevSecOps principles and practices.
K-45	Knowledge of how to protect data privacy through code.	Explain how to protect data privacy by secure coding techniques.
K-26	Knowledge of Privacy Impact Assessments in terms of privacy and identity management.	Describe how Privacy Impact Assessment (PIA) tools are used to identify and mitigate privacy risks.
K-21	Knowledge of software collaboration tools (e.g., version control, bug tracking, continuous integration).	Describe how software tools are used to collaborate and manage the phases of the software development lifecycle.
K-40	Knowledge of open source software and risks involved.	Compare and contrast common open-source frameworks and tools used for software development, including describing the risks involved.
K-35	Knowledge of the difference between AI and ML.	Differentiate between Artificial Intelligence (AI) and Machine Learning (ML) methods and algorithms.
K-20	Knowledge of AI and ML methods and algorithms.	
K-33	Awareness of cloud computing concepts (e.g., IoT, edge computing).	Describe different cloud computing concepts such as IoT and edge computing.
K-47	Knowledge of performing integrated quality assurance testing for security functionality and resiliency attack.	Demonstrate how to perform an integrated quality assurance test for security functionality and resiliency attacks.
K-48	Knowledge of how to identify security implications in the software acceptance phase including completion criteria, risk acceptance and documentation, common criteria, and methods of independent testing and report concerns to IT/software team.	Explain how to identify security implications in the software acceptance phase.
K-49	Knowledge of applications with public keying by leveraging existing public key infrastructure (PKI) libraries and incorporating certificate management and encryption functionalities when appropriate.	Demonstrate how to use public keying by leveraging existing public key infrastructure (PKI) libraries while incorporating certificate management and encryption functionalities.
K-50	Knowledge of how to identify and leverage the enterprise-wide security services while designing and developing secure applications (e.g., Enterprise PKI, Federated Identity server) when appropriate.	Explain how to identify and leverage enterprise-wide security services while designing and developing secure applications.

K-51	Knowledge of how to identify and analyze user needs and use needs to establish a plan in the selection, creation, evaluation, implementation and administration of information technology systems.	Discuss how to identify and analyze user needs when creating information technology systems.
K-52	Knowledge of security requirements into application design elements including documenting the elements of the software attack surfaces, conducting threat modeling, and defining any specific security criteria.	Discuss the security factors to consider when designing software application, including attack surface analysis, threat modeling, and specific security criteria.
K-53	Knowledge of architecture patterns and when to use them to build applications.	Explain contemporary software architecture patterns and how to use them in creating applications.
K-54	Knowledge of algorithms and data structures (e.g., big-O, linked lists, hash maps, sorting, etc.).	Discuss the differences between algorithms and data structures such as big-O, linked lists, hash maps, trees, sorting, and searching.
K-55	Knowledge of Binary search tree and how binary search works.	
K-56	Knowledge of Hash maps.	
Skills		**Student Learning Outcomes**
S-1	Skill in using built-in functions as well as skill in creating custom functions, subroutines, and procedures within software using scripting languages.	Design and create a modularized, distributed application using a scripting language.
S-15	Skill in using a scripting language on the server side and the client side of a distributed program.	
S-2	Skill in integrating standard object model components with server pages in support of the User Experience.	Develop a server page containing integrated object model components.
S-3	Skill in conducting software debugging.	Evaluate a software program's effectiveness by using debugging tools and techniques.
S-4	Skill in creating programs that validate and process multiple inputs including command line arguments, environmental variables, and input streams.	Design and implement applications which acquire and validate input from various sources, including command line arguments, files, environment variables, and input streams.
S-5	Skill in writing code in current programming languages and frameworks.	Develop applications which use a standard current programming language.
S-6	Skill in developing applications that can log and handle errors, exceptions, and application faults and logging.	Create applications that can log and handle errors, and exceptions, and identify application faults.
S-7	Skill in applying root cause analysis (RCA) techniques to solving software/customer issues.	Apply root cause analysis techniques to identify and diagnose software defects.
S-8	Skill in the live production environment (e.g., monitoring, logging, alerting, remote debugging).	Create a live production environment that will support monitoring, logging, alert processing, and debugging remotely.
S-9	Skill in using electronic mail software (e.g., Google Gmail; IBM Notes Hot technology; Microsoft Exchange Server Hot technology; Microsoft Outlook Hot technology).	Demonstrate the use of electronic mail systems.
S-10	Skill in using graphical user interface development software (e.g., Graphical user interface GUI builder software; Graphical user interface GUI design software; Salesforce Visualforce Hot technology).	Create software using a graphical user interface.
S-11	Skill in using object or component oriented development software (e.g., C++ Hot technology; Document Object Model DOM Scripting; Python Hot technology; Simple API for XML SAX).	Use a component-driven development software tool to build software components.
S-12	Skill in creating classes that use inheritance aspects of the object-oriented paradigm.	Create objects that inherit properties and methods from a class.
S-13	Skill in using, incorporating and utilizing cookies.	Create web applications that use common services, practices, protocols, and cookies.
S-14	Skill in implementing programs that use local or remote databases with standard protocols.	Design and develop programs that use databases with standard protocols.
S-16	Skill in evaluating and reporting software needs, constraints, analysis for application-specific concerns.	Evaluate domain-specific needs, identify requirements, and define scope for applications.
S-17	Skill in implementing levels of security in distributed software applications and applets.	Deploy secure software according to secure software deployment methodologies, tools, and practices.
S-18	Skill in deploying secure software according to secure software deployment methodologies, tools, and practices (e.g., PCI,GDPR, HIPPA, CCPA).	
S-19	Skill in mobile application development.	Design and develop mobile applications.
S-20	Skills such as time management, risk management.	Apply time management and risk management skills.

S-21	Skill in incorporating user experience feedback into software.	Incorporate user experience feedback into software applications.
S-22	Skill in integrating third party open source resources into software.	Integrate third-party open source components into applications.
S-23	Skill in learning new and/or industry standard tools involved in the development of software.	Evaluate and learn new and standard software tools as they become available.
Abilities		**Student Learning Outcomes**
A-1	Ability to both mentor and be mentored; provide critical feedback as well as accept critical feedback two-way.	Distinguish the responsibilities of being a mentor and a mentee and perform in both roles.
A-2	Ability to comprehend and execute both written and oral instructions by asking clarifying questions.	Effectively execute written and oral instructions after asking clarifying questions.
A-3	Ability to effectively communicate technical concepts and constraints in written and oral form to technical team members, stakeholders.	
A-4	Ability to work effectively in multi-disciplinary teams to apply information technology in support of organizational goals.	Demonstrate the ability to work effectively in multi-disciplinary teams to meet and support organizational goals.
A-9	Ability to engage with users and understand their user experience.	
A-5	Ability to produce technical content for tech writers.	Organize technical content for technical writers to finalize.
A-6	Ability to manage your own software development project activities and deliverables in a timely and efficient manner.	Organize and schedule software projects to meet one's own deliverables timeline.
A-7	Ability to work on team projects and demonstrate critical thinking, teamwork, oral communications, inter-cultural appreciation, and technical and information literacy skills.	Demonstrate effective team collaboration and communications skills for technical and information proficiency.
A-8	Ability to research and be able to find other sources to answer the problem.	Perform research to solve a problem.
A-10	Ability to draw on prior knowledge and experience in a new situation.	Apply prior knowledge and experience in new situations.

Software Development Degree Expectations

A pool of 20 community college and four-year university faculty members from across the country were asked to categorize each knowledge, skill, ability, and task below. The question posed to them: would these KSA+Ts be reasonably included in a two-year AAS program, a four-year Bachelor's program, both, or neither? These results provide another tool for educators to use in assessing how to best incorporate each knowledge, skill, ability, and task.

		% Best Estimate			
	Tasks	2 Year AAS	Both 2 yr AAS and 4 yr Academic Degree	4 Year Academic Degree	Number of responses
T-1	Identify, document and effectively communicate security concerns and/or threat vulnerabilities.	11%	73%	16%	19
T-2	Analyze information to determine, recommend, and plan development and installation of a new system or modification of an existing system.	25%	45%	30%	20
T-3	Develop code to read and write files.	15%	70%	15%	20
T-4	Create webpages using data from a database.	15%	70%	15%	20
T-5	Create applications such as Servlets that send HTML pages to Internet clients.	25%	35%	40%	20
T-6	Write and debug effective code using various scripting languages.	15%	65%	20%	20
T-7	Assist with development on multiple platforms (e.g. Linux, Windows, AppleOS, etc.).	15%	50%	35%	20
T-8	Design, develop and validate stable, robust, secure, and efficient code following industry best practices.	10%	70%	20%	20
T-9	Develop secure code and error handling.	15%	65%	20%	20
T-10	Develop cross platform applications targeted for an OS or platform other than the development environment.	5%	32%	63%	19
T-11	Develop applications that run on multiple browsers.	11%	72%	17%	18
T-12	Design, create, manage, and evaluate Apps.	6%	67%	28%	18
T-13	Manipulate the objects contained in the Document Object Model (DOM).	17%	50%	33%	18
T-14	Demonstrate familiarity with at least one current IDE and other developer productivity tools.	25%	65%	10%	20
T-15	Identify, evaluate, and apply efficient algorithms and data structures (e.g. sorting, multithreading).	11%	37%	53%	19
T-16	Apply SDLC (software development lifecycle) industry practices (e.g. Agile, waterfall, scrum, etc.).	5%	55%	40%	20
T-17	Assist in designing countermeasures and mitigations against potential exploitations of programming language weaknesses and vulnerabilities in system and elements.	5%	21%	74%	19
T-18	Apply secure code documentation in accordance with corporate policy to ensure safety of how code is implemented or processed for user access and security access to code that govern software driven apparatus.	16%	37%	47%	19
T-19	Compile and write documentation of existing software program development and subsequent revisions, inserting comments in the coded instructions so others can understand the program.	6%	72%	22%	18
T-20	Identify and leverage the enterprise-wide version control system while designing and developing secure applications.	11%	42%	47%	19
T-21	Collaborate with a wide range of technical professionals, in person and virtually, using tools and strategies that support cooperative software development practices.	16%	47%	37%	19
T-22	Conduct trial runs of programs and software applications to ensure that the desired information is produced and instructions and security levels are correct.	11%	68%	21%	19
T-23	Test and evaluate any software code/processes you developed - unit testing.	16%	61%	21%	19
T-24	Utilize software testing tools to implement various test strategies.	17%	56%	28%	18

T-25	Assist in developing software system testing and validation procedures, programming, and documentation.	26%	68%	5%	19
T-26	Correct errors by making appropriate changes and rechecking the program to ensure that desired results are produced.	21%	74%	5%	19
T-27	Apply coding and testing standards, security testing tools including "'fuzzing" static-analysis code scanning tools, and conduct code reviews.	0%	35%	65%	20
T-28	Determine system performance against standards and follow appropriate action plan when issues arise.	10%	40%	50%	20
T-29	Implement and properly document software patches and report any software security issues that would leave software vulnerable.	11%	42%	47%	19
T-30	Modify existing software to correct errors, adapt it to new hardware, or upgrade interfaces and improve performance.	11%	42%	47%	19
T-31	Contribute presentation materials and communicated effectively in a team meeting.	5%	80%	15%	20
T-32	Communicate with customers or other departments on project status, proposals, or technical issues, such as software system design or maintenance, including both oral and written communication.	11%	68%	21%	19
T-33	Contribute to team, follow directives from designers and engineers related to software design and implementation.	20%	55%	25%	20
Knowledge					
K-1	Knowledge of software development models (e.g. Waterfall Model, Spiral Model).	11%	42%	47%	20
K-2	Knowledge of system design tools, methods, and techniques, including automated systems analysis and design tools.	11%	63%	26%	19
K-3	Knowledge of effective software debugging principles.	15%	70%	15%	20
K-4	Knowledge of computer programming languages and principles in general.	25%	60%	15%	20
K-5	Knowledge of web services (e.g. service-oriented architecture, REST, and web service description language).	16%	63%	21%	19
K-6	Knowledge of visual representations of a program or system. (e.g. UML, etc.)	10%	60%	30%	20
K-7	Knowledge of how programs communicate across networks using asynchronous and synchronous techniques. (when to use and why)	0%	47%	47%	20
K-8	Knowledge of Software Integration Management Systems – how industry documents final product builds to show all of the elements that have changes and checks those that have not changed.	16%	32%	53%	20
K-9	Knowledge of event handling in a GUI.	12%	71%	18%	17
K-10	Knowledge of Regression Testing Development – how to test software using software.	10%	35%	55%	20
K-11	Knowledge of the appropriate use of cookies.	10%	65%	25%	20
K-12	Knowledge of how applets differ from applications in terms of program form, operating context, and how they are started.	20%	55%	25%	20
K-13	Knowledge of two or more operating systems that are current industry standards (e.g. Linux, Windows Apple OS).	5%	68%	26%	19
K-14	Knowledge of error handling constructs.	10%	75%	15%	20
K-15	Knowledge of the differences between client-side scripting and server-side scripting.	10%	65%	25%	20
K-16	Knowledge of common program architectures (e.g. standalone, three-tier, web-based, cloud-based, serverless, microservice).	5%	53%	42%	19
K-17	Knowledge of the local development cycle (e.g. build, deploy, test, debug).	15%	75%	10%	20
K-18	Knowledge of server software patterns, messaging patterns both async and synch.	11%	39%	50%	18
K-19	Knowledge of database integration/management software.	5%	58%	37%	19
K-20	Knowledge of AI and ML methods and algorithms.	0%	39%	61%	18
K-21	Knowledge of software collaboration tools (e.g. version control, bug tracking, continuous integration).	17%	56%	28%	18

K-22	Knowledge of the limits vs actual process of continuous integration and production deployment practices of devsecops/devnetsecops.	0%	42%	58%	19
K-23	Knowledge of cybersecurity and privacy principles and methods that apply to software development.	5%	80%	15%	20
K-24	Knowledge of system and application security threats and vulnerabilities (e.g. buffer overflow, mobile code, cross-site scripting, Procedural Language/Structured Query Language [PL/SQL] and injections, race conditions, covert channel, replay, return-oriented attacks, malicious code).	0%	56%	44%	18
K-25	Knowledge of code security (e.g. hashing, encryption, cryptography, threat modeling).	5%	47%	47%	19
K-26	Knowledge of Privacy Impact Assessments in terms of privacy and identity management.	7%	47%	47%	15
K-27	Knowledge of cyber threats and vulnerabilities.	5%	90%	5%	19
K-28	Knowledge of software related information technology (IT) security principles and methods (e.g. modularization, layering, abstraction, data hiding, simplicity/minimization).	0%	63%	37%	19
K-29	Awareness of standards such as PCI, PHI, and GDPR.	0%	47%	53%	17
K-30	Knowledge of basic security practices including threats and vulnerabilities that may arise from interactions with other systems, external and legacy code.	11%	58%	32%	19
K-31	Knowledge of computer network fundamentals (e.g., TCP/IP, HTTPS, ports, firewall, LAN/WAN, etc.)and network security methodologies.	16%	58%	26%	19
K-32	Knowledge of implementation and utilization of cloud services including deployment (e.g. AWS, Microsoft Azure).	16%	53%	32%	19
K-33	Awareness of cloud computing concepts (e.g. IoT, edge computing).	16%	42%	42%	19
K-34	Knowledge of software development and implementation for communicating and gathering data from IoT devices.	18%	35%	47%	17
K-35	Knowledge of the difference between AI and ML.	0%	56%	44%	18
K-36	Awareness of current and specialized AI and ML tools and their application to business problems.	0%	44%	56%	18
K-37	Conceptual knowledge of PKI.	6%	44%	50%	16
K-38	Knowledge of DevSecOps.	0%	53%	47%	17
K-39	Knowledge of structured and unstructured data sources.	0%	58%	42%	19
K-40	Knowledge of open source software and risks involved.	5%	89%	5%	19
K-41	Knowledge of ethics and its application to software development.	5%	84%	11%	19
K-42	Knowledge of best practices for Design/UI/UX/accessibility as applied to software development.	0%	74%	26%	19
K-43	Knowledge of lifecycle development/steady state/end of life.	11%	63%	26%	19
K-44	Knowledge of mobile application development.	28%	50%	22%	18
K-45	Knowledge of how to protect data privacy through code.	5%	47%	47%	19
K-46	Knowledge of process flow and how the upgrade/implementation of software is accomplished through definitive understanding of team collaboration in DevOps, End of Life Cycle, and including importance of foundational security.	6%	33%	61%	18
K-47	Knowledge of performing integrated quality assurance testing for security functionality and resiliency attack.	0%	31%	68%	19
K-48	Knowledge of how to identify security implications in the software acceptance phase, including completion criteria, risk acceptance and documentation, common criteria, and methods of independent testing and report concerns to IT/software team.	0%	42%	58%	19
K-49	Knowledge of applications with public keying by leveraging existing public key infrastructure (PKI) libraries and incorporating certificate management and encryption functionalities when appropriate.	11%	22%	67%	18
K-50	Knowledge of how to identify and leverage the enterprise-wide security services while designing and developing secure applications (e.g., Enterprise PKI, Federated Identity server) when appropriate.	6%	6%	88%	17

K-51	Knowledge of how to identify and analyze user needs and use needs to establish a plan in the selection, creation, evaluation, implementation and administration of information technology systems.	6%	50%	44%	18
K-52	Knowledge of security requirements into application design elements including documenting the elements of the software attack surfaces, conducting threat modeling, and defining any specific security criteria.	6%	28%	67%	18
K-53	Knowledge of architecture patterns and when to use them to build applications.	11%	47%	42%	19
K-54	Knowledge of algorithms and data structures (e.g. big-O, linked lists, hash maps, sorting, etc.).	0%	42%	58%	19
K-55	Knowledge of Binary search tree and how binary search works.	0%	42%	58%	19
K-56	Knowledge of Hash maps.	5%	37%	58%	19
Skills					
S-1	Skill in using built-in functions as well as skill in creating custom functions, subroutines, and procedures within software using scripting languages.	15%	70%	15%	20
S-2	Skill in integrating standard object model components with server pages in support of the User Expierence.	10%	55%	35%	20
S-3	Skill in conducting software debugging.	15%	75%	10%	20
S-4	Skill in creating programs that validate and process multiple inputs including command line arguments, environmental variables, and input streams.	11%	74%	16%	19
S-5	Skill in writing code in current programming languages and frameworks.	16%	79%	5%	19
S-6	Skill in developing applications that can log and handle errors, exceptions, and application faults and logging.	17%	50%	33%	18
S-7	Skill in applying root cause analysis (RCA) techniques to solving software/customer issues.	6%	28%	67%	18
S-8	Skill in the live production environment (e.g. monitoring, logging, alerting, remote debugging).	11%	28%	61%	18
S-9	Skill in using electronic mail software (e.g. Google Gmail; IBM Notes Hot technology; Microsoft Exchange Server Hot technology; Microsoft Outlook Hot technology).	22%	56%	22%	18
S-10	Skill in using graphical user interface development software (e.g. Graphical user interface GUI builder software; Graphical user interface GUI design software; Salesforce Visualforce Hot technology).	11%	53%	37%	19
S-11	Skill in using object or component oriented development software (e.g. C++ Hot technology; Document Object Model DOM Scripting; Python Hot technology; Simple API for XML SAX).	6%	50%	44%	18
S-12	Skill in creating classes that use inheritance aspects of the object-oriented paradigm.	11%	63%	26%	19
S-13	Skill in using, incorporating and utilizing cookies.	11%	58%	32%	19
S-14	Skill in implementing programs that use local or remote databases with standard protocols.	5%	53%	42%	19
S-15	Skill in using a scripting language on the server side and the client side of a distributed program .	11%	61%	28%	18
S-16	Skill in evaluating and reporting software needs, constraints, analysis for application-specific concerns.	16%	58%	26%	19
S-17	Skill in implementing levels of security in distributed software applications and applets.	11%	50%	39%	18
S-18	Skill in deploying secure software according to secure software deployment methodologies, tools, and practices (e.g. PCI,GDPR, HIPPA, CCPA).	5%	21%	74%	19
S-19	Skill in mobile application development.	11%	58%	32%	19
S-20	Skills such as time management, risk management.	5%	79%	16%	19
S-21	Skill in incorporating user experience feedback into software.	11%	63%	21%	18
S-22	Skill in integrating third-party open-source resources into software, including minimizing risk.	21%	37%	42%	19

S-23	Skill in learning new and/or industry standard tools involved in the development of software.	26%	63%	11%	19
Abilities					
A-1	Ability to both mentor and be mentored; provide critical feedback as well as accept critical feedback two-way.	5%	55%	40%	20
A-2	Ability to comprehend and execute both written and oral instructions by asking clarifying questions.	11%	68%	21%	19
A-3	Ability to effectively communicate technical concepts and constraints in written and oral form to technical team members, stakeholders.	11%	79%	11%	19
A-4	Ability to work effectively in multi-disciplinary teams to apply information technology in support of organizational goals.	5%	75%	20%	20
A-5	Ability to produce technical content for tech writers.	5%	60%	35%	20
A-6	Ability to manage your own software development project activities and deliverables in a timely and efficient manner.	0%	72%	28%	18
A-7	Ability to work on team projects and demonstrate critical thinking, teamwork, oral communications, intercultural appreciation, and technical and information literacy skills.	5%	80%	15%	20
A-8	Ability to research and be able to find other sources to answer the problem.	5%	74%	21%	19
A-9	Ability to engage with users and understand their user experience.	10%	70%	20%	20
A-10	Ability to draw on prior knowledge and experience in a new situation.	5%	80%	15%	20