	Software Development KSAs	
		AVG
Tasks SPECIFIC THINGS an entry level person would BE EXPECTED TO PERFORM on the job WITH LITTLE SUPERVISION.		
	Analysis and Design	
T-1	Identify and analyze user needs and use needs to establish a plan in the selection, creation, evaluation,	3.0
T-2	Identify, document and effectively communicate security concerns and/or threat vulnerabilities.	3.0
T-3	Analyze information to determine, recommend, and plan development and installation of a new system or modification of an existing system.	2.8
T-4	Translate security requirements into application design elements including documenting the elements of the software attack surfaces, conducting threat modeling, and defining any specific security criteria.	2.9
T-5	Prepare detailed interface diagrams that describe input, output, and logical operation, and convert them into a series of instructions coded in a computer language (This statement is agnostic with respect to specific technology or tool).	3.2
	Programming	
T-6	Develop code to read and write files.	3.3
T-7	Create webpages using data from a database.	3.5
T-8	Create applications such as Servlets that send HTML pages to Internet clients.	3.0
T-9	Write and debug effective code using various scripting languages.	3.7
T-10	Assist with development on multiple platforms (e.g. Linux, Windows, AppleOS, etc.).	3.0
T-11	Design, develop and validate stable, robust, secure, and efficient code following industry best practices.	3.5
T-12	Develop secure code and error handling.	3.6
T-13	Develop cross-platform applications targeted for an OS or hardware different from the development machine.	2.9
T-14	Develop applications that run on multiple browsers.	3.3
T-15	Design, create, manage, and evaluate Apps.	2.9
T-16	Manipulate the objects contained in the Document Object Model (DOM).	2.9
T-17	Demonstrate familiarity with at least one current IDE and other developer productivity tools.	3.6
T-18	Identify, evaluate, and apply efficient algorithms and data structures (e.g. sorting, multithreading).	3.3
T-19	Apply SDLC (software development lifecycle) industry practices (e.g. Agile, waterfall, scrum, etc.).	3.5
T-20	Enable applications with public keying by leveraging existing public key infrastructure (PKI) libraries and incorporating certificate management and encryption functionalities when appropriate.	2.8
T-21	Identify and leverage the enterprise-wide security services while designing and developing secure applications (e.g., Enterprise PKI, Federated Identity server) when appropriate.	2.6
T-22	Assist in designing countermeasures and mitigations against potential exploitations of programming language weaknesses and vulnerabilities in system and elements.	2.7
T-23	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.	2.9
T-24	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-25	Identify and leverage the enterprise-wide version control system while designing and developing secure applications.	3.4
T-26	Collaborate with a wide range of technical professionals, in person and virtually, using tools and strategies that support cooperative software development practices.	3.5
	Testing	
T-27	Conduct trial runs of programs and software applications to ensure that the desired information is produced and instructions and security levels are correct.	3.4
T-28	Test and evaluate any software code/processes you developed - unit testing.	3.8
T-29	Utilize software testing tools to implement various test strategies.	3.4

T-30	Perform integrated quality assurance testing for security functionality and resiliency attack.	3.0
T-31	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.	2.7
T-32	Assist in developing software system testing and validation procedures, programming, and documentation.	3.4
T-33	Correct errors by making appropriate changes and rechecking the program to ensure that desired results are produced.	3.7
T-34	Apply coding and testing standards, security testing tools including "'fuzzing" static-analysis code scanning tools, and conduct code reviews.	2.9
	Implementation	
T-35	Determine system performance against standards and follow appropriate action plan when issues arise.	3.0
T-36	Implement and properly document software patches and report any software security issues that would leave software vulnerable.	2.9
T-37	Modify existing software to correct errors, adapt it to new hardware, or upgrade interfaces and improve performance.	3.2
T-38	Develop presentation materials/presentations and effectively present to a technical/non-technical audience.	2.7
T-39	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.	3.2
T-40	Contribute to team, follow directives from designers and engineers related to software design and implementation.	3.3
	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).	3.4
K-2	Knowledge of system design tools, methods, and techniques, including automated systems analysis and design tools.	3.2
K-3	Knowledge of effective software debugging principles.	3.8
K-4	Knowledge of computer programming languages and principles in general	3.9
K-5	Knowledge of web services (e.g. service-oriented architecture, Simple Object Access Protocol, and web service description language).	3.5
K-6	Knowledge of UML documents that model a program.	2.7
K-7	Knowledge of how programs communicate across the Internet using conventions such as Remote Method Invocation.	3.0
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.	2.8
K-9	Knowledge of event handling in a GUI.	3.2
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.	2.7
K-13	Knowledge of two or more operating systems that are current industry standards (e.g. Linux, Windows Apple OS).	3.2
K-14	Knowledge of error handling constructs.	3.5
K-15	Knowledge of the differences between client-side scripting and server-side scripting.	3.6
K-16	Knowledge of block chain processes and practices.	2.3
K-17	Knowledge of common program architectures (e.g. standalone, three-tier, web-based, cloud-based, serverless, microservice).	3.4
K-18	Knowledge of the local development cycle (e.g. build, deploy, test, debug).	3.9
K-19	Knowledge of server software patterns, messaging patterns both async and synch.	2.7

K-20	Knowledge of Enterprise application integration software.	2.3
K-21	Knowledge of database integration/management software.	3.2
K-22	Knowledge of AI and ML methods and algorithms.	2.5
K-23	Knowledge of software collaboration tools (e.g. version control, bug tracking, continuous integration).	3.7
K-24	Knowledge of continuous automation and production deployment practices.	3.2
K-25	Knowledge of cybersecurity and privacy principles and methods that apply to software development.	3.5
K-26	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).	3.2
K-27	Knowledge of code security (e.g. hashing, encryption, cryptography, threat modeling).	3.1
K-28	Knowledge of Privacy Impact Assessments in terms of privacy and identify management.	2.7
K-29	Knowledge of risk management framework and processes (e.g. methods for assessing and mitigating risk).	2.8
K-30	Knowledge of cyber threats and vulnerabilities.	3.1
K-31	Knowledge of software related information technology (IT) security principles and methods (e.g. modularization, layering, abstraction, data hiding, simplicity/minimization).	3.1
K-32	Awareness of standards such as PCI, PHI, and GDPR.	2.4
K-33	Knowledge of basic security practices including threats and vulnerabilities that may arise from interactions with other systems, external and legacy code.	2.8
K-34	Knowledge of computer network fundamentals (e.g. TCP/IP, HTTPS, ports, firewall, LAN/WAN etc.)and network security methodologies.	2.9
K-35	Knowledge of implementation and utilization of cloud services including deployment (e.g. AWS, Microsoft Azure).	3.0
K-36	Knowledge of implementing edge-cloud software controls and services.	3.4
K-37	Knowledge of software development and implementation for communicating and gathering data from IoT devices.	3.0
K-38	Knowledge of the difference between AI and ML.	2.7
K-39	Awareness of current and specialized AI and ML tools and their application to business problems.	2.1
K-40	Conceptual knowledge of PKI.	3.3
K-41	Knowledge of DevSecOps.	2.6
K-42	Knowledge of structure and unstructured data sources.	3.6
K-43	Knowledge of open source software.	3.1
K-44	Knowledge of ethics and its application to software development.	3.4
K-45	Knowledge of best practices for Design/UI/UX/accessibility as applied to software development.	3.0
K-46	Knowledge of lifecycle development/steady state/end of life.	3.0
K-47	Knowledge of mobile application development.	2.7
K-48	Knowledge of how to protect data privacy through code.	3.0
K-49	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.	3.0
	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.	3.6

S-2	Skill in integrating standard object model components with server pages.	3.2
S-3	Skill in conducting software debugging.	3.8
S-4	Skill in creating programs that validate and process multiple inputs including command line arguments, environmental variables, and input streams.	3.4
S-5	Skill in writing code in a currently supported programming language (e.g. scripting ,Java, C++, Linux, Ruby or current languages).	3.8
S-6	Skill in developing applications that can log and handle errors, exceptions, and application faults and logging.	3.4
S-7	Skill in applying root cause analysis (RCA) techniques to solving software/customer issues.	3.1
S-8	Skill in the live production environment (e.g. monitoring, logging, alerting, remote debugging).	3.0
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).	2.8
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).	2.7
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).	3.1
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.	3.5
S-15	Skill in using a scripting language on the server side and the client side of a distributed program.	3.4
S-16	Skill in evaluating and reporting software needs, constraints, analysis for application-specific concerns.	3.0
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.	3.1
S-19	Skill in designing software applications that are accessible by a variety of wireless and wired devices.	3.0
S-20	Skills such as time management, risk management.	3.3
S-21	Skill in incorporating user experience feedback into software.	3.0
S-22	Skill in integrating third party open source resources into software.	2.7
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.	3.5
A-2	Ability to comprehend and execute both written and oral instructions by asking clarifying questions.	3.9
A-3	Ability to effectively communicate technical concepts and constraints in written and oral form to technical team members, stakeholders.	3.7
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 write technical documentation for technical and nontechnical audiences.	3.2
A-6	Ability to manage your own software development project activities and deliverables in a timely and efficient manner.	3.4
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.	3.6
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.	3.6