SOFTWARE AND APP DESIGN 11.0202.00

TECHNICAL STANDARDS

A Standards Validation Committee of industry representatives and educators reviewed and updated these standards on December 11, 2017. Completion of the program prepares students to meet the requirements of one or more industry certification: Cybersecurity Fundamentals Certificate, Oracle Certified Associate, Java SE 8 Programmer, Certified Internet Web (CIW) - JavaScript Specialist, CompTIA A+, CompTIA IT Fundamentals, CSX Cybersecurity Fundamentals Certificate, and Microsoft Technology Associate (MTA). The Arizona Career and Technical Education Quality Commission, the validating entity for the Arizona Skills Standards Assessment System, endorsed these standards on January 25, 2018.

Note: Arizona's Professional Skills are taught as an integral part of the Software and App Design program.

The Technical Skills Assessment for Software and App Design is available SY2020-2021.

Note: In this document i.e. explains or clarifies the content and e.g. provides examples of the content that must be taught.

STANDARD 1.0 APPLY PROBLEM-SOLVING AND CRITICAL THINKING SKILLS

1.1 Establish objectives and outcomes for a task

1.2 Explain the process of decomposing a large programming problem into smaller, more manageable procedures

1.3 Explain "visualizing" as a problem-solving technique prior to writing code

1.4 Describe problem-solving and troubleshooting strategies applicable to software development

STANDARD 2.0 RECOGNIZE SECURITY ISSUES

2.1 Identify common computer threats (e.g., viruses, phishing, suspicious email, social engineering, spoofing, identity theft, and spamming)

2.2 Describe potential vulnerabilities in software (e.g., OWASP's Top 10)

2.3 Identify procedures to maintain data integrity and security (e.g., lock the screen, delete unrecognized emails, use trustworthy thumb drives, and use approved software)

2.4 Explain best practices to maintain integrity and security in software development (e.g., encryption, hashing, and digital signatures)

2.5 Describe methods for sanitizing user input to prevent issues (e.g., buffer overflows and SQL injection)

2.6 Explain the CIA (confidentiality, integrity, and availability) triad

2.7 Explain how software defects relate to software security (e.g., buffer overflows and cross-site scripting)

STANDARD 3.0 EXAMINE LEGAL AND ETHICAL ISSUES RELATED TO INFORMATION TECHNOLOGY

3.1 Explore intellectual property rights including software licensing and software duplication [e.g., Digital Millennium Copyright Act (DMCA), software licensing, and software duplication]

3.2 Compare and contrast open source and proprietary systems in relation to legal and ethical issues (e.g., data pricing, use of public and private networks, social networking, industry-related data, and data piracy)

3.3 Identify issues and regulations affecting computers, other devices, the internet, and information privacy (e.g., HIPAA, COPPA, CISPA, FERPA, PCI, GDPR, and data brokers)

STANDARD 4.0 UTILIZE PRIMITIVE DATA TYPES AND STRINGS IN WRITING PROGRAMS

4.1 Declare numeric, Boolean, character, string variables, and float and double

4.2 Choose the appropriate data type for a given situation

4.3 Identify the correct syntax and usage for constants and variables in a program

4.4 Identify the correct syntax and safe functions for operations on strings, including length, substring, and concatenation

4.5 Explain complications of storing and manipulating data (i.e., the Big-O notation for analyzing storage and efficiency concerns, etc.)

4.6 Research industry relevant programming languages (i.e., Java, JavaScript, Python, etc.)

STANDARD 5.0 PERFORM BASIC COMPUTER MATHEMATICS IN INFORMATION TECHNOLOGY

5.1 Apply basic mathematics to hardware (e.g., bits, bytes, kilobytes, megabytes, gigabytes, and terabytes)

5.2 Use binary to decimal, decimal to hexadecimal, hexadecimal to decimal, binary to hexadecimal, and hexadecimal to binary conversions to solve hardware and software problems

5.3 Identify and correctly use arithmetic operations applying the order of operations (precedence) with respect to programming

5.4 Interpret and construct mathematical formulas

5.5 Identify correct and problematic uses of integers, floating-point numbers, and fixed-point numbers in arithmetic

STANDARD 6.0 UTILIZE CONDITIONAL STRUCTURES IN WRITING PROGRAMS

6.1 Use the correct syntax for decision statements (e.g., if/else, if, and switch case)

6.2 Compare values using relational operators (e.g., =, >, <, >=, <=, and not equal)

6.3 Evaluate Boolean expressions (e.g., AND, OR, NOT, NOR, and XOR)

6.4 Use the correct nesting for decision structures

STANDARD 7.0 UTILIZE ITERATIVE STRUCTURES IN WRITING PROGRAMS

7.1 Identify various types of iteration structure (e.g., while, for, for-each, and recursion)

7.2 Identify how loops are controlled (variable conditions and exits)

7.3 Use the correct syntax for nested loops

7.4 Compute the values of variables involved with nested loops

STANDARD 8.0 UTILIZE BASIC DATA STRUCTURES IN WRITING PROGRAMS

8.1 Demonstrate basic uses of arrays including initialization, storage, and retrieval of values

8.2 Distinguish between arrays and hash maps (associative arrays)

8.3 Identify techniques for declaring, initializing, and modifying user-defined data types

8.4 Search and sort data in an array

8.5 Create and use two-dimensional arrays

8.6 Describe the efficiency of different sorting algorithms (e.g., bubble, insertion, and merge)

8.7 Describe the efficiency of linear vs. binary searches [e.g., O(n) and O(log n)]

STANDARD 9.0 IDENTIFY INTERNET PROTOCOLS AND OPERATIONS

9.1 Explain cloud-based computing and content delivery networks

9.2 Identify the components and functions of the internet (e.g., HTTP, HTTPS, FTP, IP addresses, and IMAP)

9.3 Identify services run by web servers [e.g., scripting languages (client- and server-side scripting), databases, and media]

9.4 Identify performance issues (e.g., bandwidth, internet connection types, pages loading slowly, resolution, and size graphics)

9.5 Differentiate among shared hosting, dedicated server, and virtual private server (VPS)

9.6 Identify Internet of Things (IOT) and common communication interfaces (e.g., Bluetooth, NFC, Wi-Fi, and LTE)

STANDARD 10.0 APPLY CLIENT-SIDE INTERNET SOFTWARE

10.1 Identify key components and functions of internet and web specialty browsers

10.2 Use client collaboration sources/platforms (e.g., GitHub, Google Drive, Dropbox, JSFiddle, and browser developer tools)

10.3 Analyze remote computing tools and services and their application

STANDARD 11.0 DEMONSTRATE PROGRAM ANALYSIS AND DESIGN

11.1 Implement the steps in the System Development Life Cycle (SDLC) (e.g., planning, analysis, design, development, testing, implementation, and maintenance)

11.2 Develop program requirements/specifications and a testing plan (e.g., user stories, automated testing, and test procedures)

Note: In this document i.e. explains or clarifies the content and e.g. provides examples of the content that must be taught.

11.3 Apply pseudocode or graphical representations to plan the structure of a program or module (e.g., flowcharting, white boarding, and UML)

11.4 Create and implement basic algorithms

STANDARD 12.0 DEVELOP A PROGRAM

12.1 Use a program editor to enter and modify code

12.2 Identify correct input/output statements

12.3 Choose the correct method of assigning input to variables including data sanitization

12.4 Choose the correct method of outputting data with formatting and escaping

12.5 Differentiate between interpreted and compiled code (e.g., steps necessary to run executable code)

12.6 Identify the purpose of a build system (e.g., make, rake, ant, maven, SCons, and grunt)

12.7 Apply industry standards in documentation (e.g., self-documenting code; function-level, program-level, and user-level documentation)

12.8 Name identifiers and formatting code by applying recognized conventions

12.9 Demonstrate refactoring techniques to reduce repetitious code and improve maintainability

12.10 Demonstrate the use of parameters to pass data into program modules

12.11 Demonstrate the use of return values from modules

STANDARD 13.0 TEST AND DEBUG TO VERIFY PROGRAM OPERATION

13.1 Identify errors in program modules

13.2 Identify boundary cases and generate appropriate test data

13.3 Perform integration testing including tests within a program to protect execution from bad input or other run-time errors

13.4 Categorize, identify, and correct errors in code, including syntax, semantic, logic, and runtime

13.5 Perform different methods of debugging (e.g., hand-trace code and real time debugging tools)

STANDARD 14.0 UTILIZE AND CREATE COMMUNITY RESOURCES

14.1 Use standard library functions

14.2 Find and use third party libraries (e.g., web-based and package managers)

14.3 Explain and interact with an Application Program Interface (API)

STANDARD 15.0 USE VERSION CONTROL SYSTEMS

15.1 Identify the purpose of version control systems (e.g., Git and Mercurial)

15.2 Create a new repository

15.3 Add, push, and pull source code from repository

15.4 Explain branching and its uses

15.5 Restore previous versions of code from the repository

STANDARD 16.0 APPLY USER DESIGN PRINCIPLES TO INCLUDE WEBSITES AND APPLICATIONS

16.1 Apply W3C standards and style conventions

16.2 Construct web pages and applications that are compliant with ADA and sections 504 and 508 standards

16.3 Explain the concept of responsive design and applications

16.4 Employ graphics methods to create images at specified locations

16.5 Choose correct GUI objects for input and output of data to the GUI interface (e.g., text boxes, labels, radio buttons, check boxes, dropdowns, and list boxes)

STANDARD 17.0 USE AND UPDATE DATA STORAGE AND MANAGEMENT

17.1 Input/output data from a sequential file or database

17.2 Demonstrate creating, reading, updating, and dropping a database

17.3 Demonstrate the proper use of SQL database applications that work with different languages (e.g., MongoDB, Microsoft Access, Oracle Databases, and Code.org's App Lab)

Note: In this document i.e. explains or clarifies the content and e.g. provides examples of the content that must be taught.

STANDARD 18.0 EMPLOY OBJECT-ORIENTED PROGRAMMING TECHNIQUES

18.1 Make a distinction between an object and a class
18.2 Differentiate among inheritance, composition, and class relationships
18.3 Instantiate objects from existing classes
18.4 Read the state of an object by invoking accessor methods
18.5 Change the state of an object by invoking a modifier method
18.6 Determine the requirements for constructing new objects by reading the documentation
18.7 Create a user-defined class
18.8 Create a subclass of an existing class
18.9 Identify the use of an abstract class as opposed to an interface
18.10 Explain the object-oriented concepts of polymorphism, inheritance, and encapsulation

STANDARD 19.0 EMPLOY RUNTIME AND ERROR HANDLING TECHNIQUES

19.1 Identify runtime errors
19.2 Describe error handling strategies
19.3 Handle unexpected return values
19.4 Handle (catch) runtime errors and take appropriate action
19.5 Throw standard exception classes
19.6 Develop and throw custom exception classes