MOBILE APPLICATIONS DEVELOPMENT
Course Code: 5378
As aligned to:
South Carolina Computer Science High School Process and Content Standards

The South Carolina Computer Science and Digital Literacy Process Standards should be integrated into every grade level within the South Carolina Computer Science and Digital Literacy Content Standards. Because the Process Standards drive the pedagogical component of teaching and serve as the means by which students should demonstrate understanding of the content standards, the process standards must be incorporated as an integral part of overall student expectations when assessing content understanding.

A computer science literate student can:

1. Foster an inclusive computing culture.
   a. Recognize that equitable access to computing benefits society as a whole.
   b. Consider others' perspectives as well as one's own perspective when developing computational solutions.
   c. Consider the needs of a variety of end users regarding accessibility and usability.

2. Collaborate around computing.
   a. Select appropriate technological tools that can be used to collaborate on a project.
   b. Collaborate productively with individuals of varying perspectives, skills, and backgrounds.
   c. Set and implement equitable expectations and workloads when working in teams.
   d. Integrate constructive feedback while working in teams.

3. Recognize, define, and analyze computational problems.
   a. Recognize when it is appropriate to solve a problem computationally.
   b. Make sense of computational problems and persevere in solving them.
   c. Relate computational problems to prior knowledge.
   d. Recognize that there may be multiple approaches to solving a problem.
   e. Approach problem solving iteratively, using a cyclical process.

4. Create, test, and refine computational artifacts.
   a. Consider the purpose of computational artifacts for practical use, personal expression, and/or societal impact.
   b. Recognize when to use the same solution for multiple problems.
   c. Test computational artifacts systematically by considering multiple scenarios and using test cases.
   d. Approach troubleshooting systematically.
   e. Consider performance, reliability, usability, and accessibility when evaluating and refining computational artifacts.

5. Communicate about computing.
   a. Select and use appropriate technological tools to convey solutions to computing problems.
   b. Communicate about computational processes and solutions using appropriate terminology consistent with the intended audience and purpose.
   c. Articulate ideas responsibly by observing intellectual property rights and giving appropriate attribution.

South Carolina Computer Science High School Content Standards

Computing Systems

Standard 1: Examine how hardware and software contribute to computing devices solving relevant problems.

HS1.CS.1.1

Analyze the impact that computing devices have in real-world settings (e.g., traffic lights, medical devices, facial recognition).

HS2.CS.1.1

Investigate how a problem is systematically solved through the selection and integration of hardware and software components.

HS3.CS.1.1

Recommend modifications for existing computing devices and software to improve functionality for end users.

HS4.CS.1.1

Develop a solution to a given problem using appropriate hardware and software (e.g., sensor devices, Wi-Fi capabilities, specialized displays, runtime modules, operating systems, application programming interfaces (APIs)).

HS1.CS.1.2

Compare and contrast the elements of a computing system by examining hardware elements for their intended use (e.g., input-output (I/O) devices, random access memory (RAM), read only memory (ROM), storage devices, motherboards, and processors including the arithmetic logic unit (ALU), control unit, registers, cache memory, example implementations of some of these components using logic gates) (Virginia, 2017).

HS3.CS.1.2

Justify hardware and software selections for specific applications by evaluating the components (e.g., databases, sensors, application programming interfaces (APIs)) of various computing devices (e.g., desktops, laptops, tablets, smartphones, specialized devices like global positioning systems (GPSs)).

HS4.CS.1.2

Cite evidence of how selecting appropriate hardware and software components enhances user interfaces to provide better solutions for real-world problems.

HS2.SC.1.3

Analyze the roles of operating system software components (e.g., memory management, data storage and retrieval, process management, access control) in a specific solution (CSTA, 2017).

Standard 2: Troubleshoot common hardware and software problems.
HS1.CS.2.1
Interpret various types of error messages from various sources (e.g., operating systems, applications, application programming interfaces (APIs)) to assist in solving common computer malfunctions.
HS2.CS.2.1
Research credible sources of information that can be used for complex troubleshooting strategies (e.g., modifying system settings, correcting connectivity problems).
HS3.CS.2.1
Solve common computer malfunctions or describe the problem accurately, using technical vocabulary, so that others can solve it.
HS4.CS.2.1
Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors (CSTA, 2017).

Networks and the Internet
Standard 1: Evaluate data transmission across networks, including the Internet.
HS1.NI.1.1
Describe how hardware, software, and protocols work together for transmitting data across networks.

Standard 2: Evaluate cybersecurity threats and appropriate security measures across networks.
HS1.NI.2.1
Reflect on case studies or current events in which governments or organizations experienced data leaks or data loss as a result of cybersecurity attacks (CSTA, 2017).

Data and Analysis
HS1.DA.2.2
Identify various types of computational models and their uses for data composed of multiple data elements that relate to one another (e.g., population data may contain information about age, gender, height) (K – 12 Framework, 2016).
HS2.DA.2.2
Compare and contrast various types of computational models and their uses for data composed of multiple data elements that relate to one another (e.g., population data may contain information about age, gender, height) (K – 12 Framework, 2016).

Standard 3: Create various ways to visually represent data.
HS1.DA.3.1
Identify a data set that could be used to solve a real-world problem.
HS3.DA.3.1
Construct a data visualization to solve a real-world problem using software tools or programming (e.g., generated scatter, bar, and line charts).

HS4.DA.3.1
Analyze patterns in a data visualization then select a collection tool to validate a claim or share information with a group of people.
HS1.DA.3.2
Organize collected data to communicate the solution to a real-world phenomenon and support a claim.

Algorithms and Programming
Standard 1: Design algorithms that can be adapted to express an idea or solve a problem.
HS1.AP.1.1
Create flowcharts and/or pseudocode to express a problem or idea as an algorithm.
HS2.AP.1.1
Create algorithms to solve computational problems that have an application in the real world (e.g., local community, church, civic organization, school, home life).
HS3.AP.1.1
Adapt predefined algorithms to solve computational problems.
HS4.AP.1.1
Evaluate algorithms in terms of efficiency, correctness, and clarity (CSTA, 2017).
Standard 2: Build a combination of control structures that supports complex execution, readability, and program performance.
HS1.AP.2.1
Trace the flow of execution of a program that uses a combination of control structures (e.g., conditionals, loops, event handlers, recursion).
HS2.AP.2.1
Design and iteratively develop programs that combine control structures (e.g., conditionals, loops, event handlers, recursion).
HS3.AP.2.1
Justify the selection of specific control structures explaining the benefits and drawbacks of choices made (e.g., tradeoffs involving implementation, readability, and program performance).

HS1.AP.2.2
Trace the flow of execution of a program that uses a variety of programming constructs (e.g., procedures, modules, objects).
HS2.AP.2.2
Design a solution through systematic analysis using programming constructs (e.g., procedures, modules, objects).
HS3.AP.2.2
Justify the selection of specific programming constructs, explaining the benefits and drawbacks of choices made on the program's execution.

Standard 3: Divide a task into sets of functional units that can be reused to compose a complex solution.
HS1.AP.3.1

Decompose tasks into smaller, reusable parts to facilitate the design, implementation, and review of programs.
HS2.AP.3.1
Develop code to solve the smaller parts of a decomposed task that can be reused to solve similar problems (e.g., procedures, functions, objects).
HS3.AP.3.1
Build a complex solution to a problem that incorporates reusable code (e.g., student created, application programming interfaces (APIs), libraries).
HS4.AP.3.1
Justify the selection of modular parts in the creation of a complex solution.

Standard 4: Plan, build, test, refine, and document programs using text-based coding languages to solve problems with varying degrees of difficulty
HS1.AP.4.1
Plan and develop programs for a variety of audiences using a process that incorporates development, feedback, and revision.
HS2.AP.4.1
Plan and develop a program that addresses potential security issues.
HS3.AP.4.1
Plan and develop a program that is accessible across multiple computing platforms (e.g., iOS, Unix, Windows, web-based).
HS4.AP.4.1
Evaluate a program through a review process (e.g., code review, beta testing, pilot group).

HS1.AP.4.2
Seek and incorporate feedback to refine a solution (e.g., users, team members, code review, teachers).
HS2.AP.4.2
Systematically test programs using a range of test cases to meet design specifications (e.g., specific outcomes, functionality, user interface, error handling) (CSTA, 2017).
HS3.AP.4.2
Evaluate and refine programs to make them more usable, functional, and accessible.
HS4.AP.4.2
Implement version control to track program refinements.

HS1.AP.4.3
Recognize the variety of documentation methods available while developing a program (e.g., inline comments, procedure header, purposeful naming).
HS2.AP.4.3
Document programs in order to make them easier to follow, test, and debug.

HS3.AP.4.3
Document programs that use non-user-created resources (e.g., code, media, libraries) giving attribution to the original creator.
HS4.AP.4.3
Justify design decisions by documenting the design process of complex programs (e.g., developer journal, digital portfolio, presentation).

HS1.AP.4.4
Examine licenses (i.e., permissions) that limit or restrict use of resources (e.g., freeware, shareware, open source, creative commons).
HS2.AP.4.4
Discuss the implications of using licensed resources in a developed solution.
HS3.AP.4.4
Develop a systematic solution that incorporates licensed resources.
HS4.AP.4.4
Research the process for licensing student-created resources.

Standard 5: Choose data types and data structures based on functionality, storage, and performance tradeoffs.
HS1.AP.5.1
Justify and use appropriate data types (i.e., primitive and non-primitive) in simple programs.
HS2.AP.5.1
Determine when data structures (e.g., lists, arrays, tuples, stacks, queues, structures) are more appropriate than simple data types and incorporate them in programs.
HS3.AP.5.1
Determine when external data structures (e.g., databases, flat files) are appropriate and incorporate them in programs.
HS4.AP.5.1
Justify how data structures and abstraction are used to manage program complexity.

Impact of Computing
Standard 1: Evaluate the impact of computing from a global perspective.
HS2.IC.1.1
Compare and contrast the efficiency, feasibility, and ethical impacts of deploying the same computing solution in various countries.
HS1.IC.1.2
Research traditional and non-traditional computer science careers.
HS2.IC.1.2
Identify a computer science career in a non- traditional computer science industry for each of the five computing disciplines *(*i.e., computer science, software engineering, information technology, information systems, computer engineering).

Standard 2: Evaluate the evolving legal and ethical tradeoffs that shape computing practices.
HS1.IC.2.1
Select the most appropriate means of communication for given situations (e.g., personal versus professional communication, communication with teachers and employers).
HS2.IC.2.1
Discuss how social media and computing devices have positively and negatively impacted communication.
HS3.IC.2.1
Justify proper and improper use of social media and computing devices (e.g., role-playing and example scenarios).
HS4.IC.2.1
Create rules of etiquette for proper use of social media and computing devices.

HS1.IC.2.2
Discuss issues related to personal security (e.g., personal, financial, professional).
HS2.IC.2.2
Define and visually display students' digital footprint.
HS3.IC.2.2
Analyze the relationship between students' digital footprint and personal security.
HS4.IC.2.2
Recommend methods to protect digital information in different situations (e.g., traveling to other countries, two-factor authentication, encryption).

HS1.IC.2.3
Explain the implications of proper and improper use of social media (e.g., college admissions, employment, cyberbullying laws).
HS2.IC.2.3
Identify ethical and legal computing practices.
HS3.IC.2.3
Distinguish among ethical, unethical, legal, and illegal computing practices (e.g., fair-use, illegal music/video downloads, sharing copyrighted pictures/videos, black-hat hacking, white-hat hacking).
HS4.IC.2.3
Investigate how computer use and digital privacy are governed across the globe (e.g., government regulations for computer use in the United State, Canada, China, North Korea, and Russia).

Standard 3: Understand the importance of access and equity in computing.
HS1.IC.3.1
Identify factors (e.g., net neutrality, government regulations, infrastructure, funding) that impact equitable access to computing resources for underrepresented groups (e.g., race, ethnicity, gender, geographic location, socioeconomic status).

HS2.IC.3.1
Research current efforts to provide equitable access to computing resources for underrepresented groups (e.g., race, ethnicity, gender, geographic location, socioeconomic status).
HS4.IC.3.1
Design a solution to improve equitable access to computing resources for underrepresented groups (e.g. classroom, school, neighborhood).

HS1.IC.3.2 Identify computer scientists from underrepresented groups and their specific contributions (e.g., African-American, Latino, women, disabled).

HS1.IC 3.3
Identify the advantages and disadvantages of diverse perspectives and backgrounds when solving computational problems.
HS2.IC.3.3
Evaluate existing computing solutions according to inclusivity or non-inclusivity (e.g., sight and hearing impairment, ethnicity, age).
HS3.IC.3.3
Recommend modifications to make a current computing solution more inclusive for all users.

Standard 4: Evaluate the history of computers and computing.
HS2.IC.4.1
Discuss the advantages and disadvantages of advancing and emerging technologies over time (e.g., the impacts of artificial intelligence, virtual reality, and biometrics on productivity, job loss, inventions, quality of life, and globalization).
HS3.IC.4.1
Hypothesize problems that the next generation of computing will solve.
HS1.IC.4.2
Define and provide examples of big data (e.g., information collected from social media or smartphone use).
HS2.IC.4.2
Research how big data is used to solve computing problems.