Skill Struck®

Skill Struck's alignment to

Utah K-5 Computer Science Standards

Legend

✅ = Standard aligned

🔹 = Not currently aligned

Standard	Status
K.CS.1 Select computing devices that perform a variety of tasks accurately and quickly based on user needs and preferences. (Practice 1: Fostering an Inclusive Computing Culture) Students will select computing devices (phone, tablet, computer, cameras, software, 3D printers, etc.) and understand that they can be used to aid in a task (email, text message, voice calls, videos, digital imaging, modeling, etc.).	✅
K.NI.1 Model and describe how people connect to other people and information through a network. (Practice 4: Developing and Using Abstractions) Students will be able to model and describe how information is sent and retrieved using a network to share information as a class. For example, demonstrating through the game of "telephone" (asking students to pass a message from one person to another). This can include sending words, images, etc. Students should demonstrate their understanding of this flow	✅

of information by drawing a model of the way packets are transmitted, programming an animation to show how packets are transmitted, or demonstrating this through an unplugged activity which has them act it out.	
K.NI.2 Create patterns to communicate a message. (Practice 4: Developing and Using Abstractions) Students will use digital devices to create patterns with pictures, objects or words to communicate. Examples may include basic coding for simple directions with arrows, manipulatives, simple directions, etc. These basic coding examples can be classroom routines, such as what to do when you enter the classroom each day.	✅
K.DA.1 Identify and describe patterns in data visualizations, such as charts or graphs, to make predictions (Practice 4: Developing and Using Abstractions) Students will show data in a pattern, which could include images of favorite fruit, sport, or cookie. Students will show what would be next in a pattern, or what might be missing from a pattern. This could be color, number, animal, or letter pattern. Teachers can use digital tools to model data visualizations for students, this could be done with an interactive board, tablets, or computers.	✅
K.AP.1 Model processes by creating and following algorithms to complete tasks. (Practice 3: Recognizing and Defining Computational Problems and Practice 4: Developing and Using Abstractions) Students will complete a familiar process or activity by creating algorithms to follow specific instructions. Emphasize real life examples, ordering, and sequenc- ing such as brushing teeth, lining up to go to lunch, school safety drills. Expose students to algorithms as sequencing events.	✅
K.CT.1 Decompose problems into smaller manageable parts to better understand them.	✅

(Practice 3: Recognizing and Defining Computational Problems) Students will be able to take a complex problem and break it down into smaller components. Examples may include breaking down the steps needed to make breakfast, get ready for school, or to code a character across the screen. Teachers may use digital tools to diagram components of a task such as drawing a shape.	
1.CS.1 Operate a variety of computing devices that perform tasks accurately and quickly based on user needs and preferences. (Practice 1: Fostering an Inclusive Computing Culture) Students will perform a variety of tasks by operating digital devices (laptop, tablet, desktop, etc.) based on availability and the task they are seeking to accomplish. For example, if students are asked to draw a picture, they should be able to open and use a drawing app/program to complete this task.	✅
1.CS.2 Explore the functions of common hardware and software components of computing systems. (Practice 6: Testing and Refining Computational Artifacts and Practice 7: Communicating About Computing) Students will explore and identify common hardware and software components (mouse, keyboard, storage, trackpad, tablet devices, laptop, monitor, application (app), input, output, etc.) as well as their function.	✅
1.DA.1 Collect and present data in various visual formats. (Practice 4: Developing and Using Abstractions and Practice 7: Communicating About Computing) Students will create surveys of things that interest them (such as favorite foods, colors, books, etc.), collect answers, and decide how to present the data in various visual formats (tally marks, color blocks stacked, sticky notes, etc.). Teachers can use digital tools to model data visualizations for students.	✅
1.DA.2 Identify and describe patterns in data visualizations (unplugged or digital), such as charts or graphs, to make predictions.	✅

(Practice 4: Developing and Using Abstractions) Students will identify and describe patterns to create hypotheses based on data visualizations (charts, graphs, etc.). Examples include drawing a picture graph and a bar graph with single-unit scale using a drawing app/program to represent a data set with up to four categories or using a chart to sort and predict colors of M&M's in a bag. This focus is making predictions based on data.	
1.AP.1 Demonstrate understanding of the way programs store and manipulate data as variables, such as numbers, words, colors, and images. (Practice 4: Developing and Using Abstractions) Students will demonstrate understanding of how computers store data (input) in a variety of ways (variables) that a program can use. For example, a number variable can only store a number and not letters. An alphanumeric variable can store numbers or letters but cannot be added or subtracted because it is text. Students could use thumbs up/down as representations of yes/no, use arrows when writing algorithms to represent direction, or in code and decode words using numbers, pictographs, or other symbols to represent letters or words.	✅
1.AP.2 Break down (deconstruct) algorithms and list the steps needed to solve a problem into a sequence of tasks and sub-tasks. (Practice 3: Recognizing and Defining Computational Problems) Students will be able to identify the steps needed to solve a problem. Students will understand that algorithms are specific instructions in order to complete a familiar process or activity. Students may use digital tools to demonstrate order- ing and sequencing of a task. Emphasize real life examples, ordering, and sequencing such as putting on your shoes, writing a story with a beginning, middle, and end, checking out library books, etc. For example, student may break down the steps needed to draw a shape or coding steps to move a character across the screen.	✅
1.AP.3 Create programs with sequences (steps) of com- mands and	✅

simple loops (repeated patterns), to express ideas or address a problem. (Practice 5: Creating Computational Artifacts) Students will create programs using elementary block programing (such as unplugged or ScratchJr on a device) that contain simple loops. These loops are repeating a pattern to create an image (such as a square), or to address a problem (such as hopscotch or designing a code that allows an avatar to avoid a barrier).	
1.IC.1 Develop and demonstrate the ability to work respectfully and responsibly with others whether communicating face-to-face or digitally. (Practice 2: Collaborating Around Computing) Students will develop and demonstrate proper etiquette when collaborating with others, physically or digitally.	✅
1.CT.1 Determine the steps needed to solve a problem and develop a sequence of instructions. (Practice 3: Recognizing and Defining Computational Problems) Students will analyze a real-world problem and develop instructions that determine the steps necessary to achieve the intended outcome such as creating a peanut butter and jelly sandwich. The connection with computer science is the need to be clear and detailed with action steps so the outcome is what is desired. Teachers may use digital tools to diagram components of a task such as making simple foods.	✅
1.CT.2 Recognize similarities between new problems and problems that have been solved in the past. (Practice 3: Recognizing and Defining Computational Problems) Students will have the opportunity to consider how previous problem-solving and code development have similarities. The use of previous solutions and strategies is useful in crafting a new solution. For example, how do the lessons learned from determining steps to create a peanut butter and jelly sandwich inform the development of steps to create a school lunch?	✅
2.CS.1 Describe and solve basic hardware and software problems.	✅

(Practice 7: Communicating About Computing) Students will describe, solve and perform basic troubleshooting tasks such as checking the device for battery charge and/or power connection, checking cord connections, turning a device off and on to reboot it, closing and reopening an app/program, plugging in headphones, etc.	
2.NI.1 Explain what a password or pass phrase is, why it is used, and be able to create a secure password. (Practice 7: Communicating About Computing) Learning to protect one's device or information from unwanted use by others is an essential first step in learning about cybersecurity. Students will be able to explain the reasoning behind having certain digital resources password protected and create an effective password and/or pass phrase.	✅
2.DA.1 Demonstrate how to store, copy, search, retrieve, modify and delete information using a computing device, and define the information stored as data. (Practice 4: Developing and Using Abstractions) Students will demonstrate how to create, modify, and save projects using the devices, platforms, applications, and software available. Data can be images, text documents, audio files, software programs or apps, video files, etc. The information is specific to data, not text within a single text document.	✅
2.DA.2 Collect and present data in various visual formats. (Practice 4: Developing and Using Abstractions and Practice 7: Communicating About Computing) Students will collect and present data in various visual formats, such as drawing a picture graph and a bar graph (with single-unit scale) with up to four categories. Students will create surveys of things that interest them (such as favorite foods, colors, books, etc.), collect answers, and decide how to present the data in various visual formats (picture graphs and bar graphs). Teachers can use digital tools to model data visualizations for	✅

students and students may collaborate to create and present digital data.	
2.DA.3 Identify and describe patterns in data visualizations to make predictions. (Practice 4: Developing and Using Abstractions) Students will make predictions by identifying and describing information in picture graphs and bar graphs. For example, the class may create and analyze a digital pictograph of favorite animals in the class. Students will make predict favorite animals for the grade level based on patterns observed.	✅
2.AP.1 Deconstruct the steps needed to solve a task into a sequence of instructions. (Practice 3: Recognizing and Defining Computational Problems) Students will be able to identify the steps needed to solve a problem. Students will understand that algorithms are specific instructions in order to complete a familiar process or activity. Students may use digital tools to demonstrate ordering and sequencing of a task, such as code a character to move across the screen or solve a coding puzzle. Emphasize real life examples, ordering, and sequencing such as putting on your shoes, writing a story with a beginning, middle, and end, checking out library books, etc.	✅
2.AP.2 Collaboratively develop plans that describe a program's sequence of events, goals, and expected outcomes. (Practice 5: Creating Computational Artifacts and Practice 7: Communicating About Computing) Students will collaborate to develop a digital program using a sequence of events that includes goals and expected outcomes. The focus is on team or pair-programming on device and leveraging roles to develop a shared solution. For example, students may work in teams to navigate an avatar to reach a goal.	✅
2.AP.3 Properly credit others when using their ideas and creations while developing programs.	✅

(Practice 7: Communicating About Computing) Using computers comes with a level of responsibility. Students will properly credit work by citing work inspired by others when developing digital programs.	
2.AP.4 Debug and solve simple problems within an algorithm or program that includes sequences and simple loops. (Practice 6: Testing and Refining Computational Artifacts) Students will test algorithms to find problems and resolve errors (debug) within the program. Students will start with an existing code that includes sequences and/or simple loops and determine the errors in the code to make improvements to achieve the task. This can be done both on device and with unplugged activi- ties. For example, students create an alternative emergency exit plan with alternative routes for evacuation drill.	✅
2.AP.5 Summarize the steps taken and choices made during the iterative process of program development. (Practice 7: Communicating About Computing) Students will summarize how a digital project was created and revised. For example, students will be able to answer who, what, where, when, why, how, etc. about the final program solution.	✅
2.IC.1 Describe how technology has impacted society over time. (Practice 3: Recognizing and Defining Computational Problems) Students will compare the advances in technology and describe how it has impacted society. For example, students can consider how a cell phone saves phone numbers in contacts and consider how that impacts whether people know important phone numbers and how that impacts society.	✅
2.IC.2 Describe rationales for keeping login information private, and for logging off devices appropriately. (Practice 3: Recognizing and Defining Computational Problems) Students will describe why people keep passwords private and secure and demonstrate how to log on and off digital devices appropriately.	✅

Skill Struck ®

3.CS.1 Describe and model how computing devices connect to other components to extend their capabilities and form a system. (Practice 7: Communicating About Computing) Students will describe and model how computing devices connect to other devices or components (physical or wireless) to create a system. For example, the relationship between the respiratory and circulatory system during physical activity serves as a metaphor for how the parts of a computer connect to allow input, processing, and output.	
3.NI.1 Describe physical and digital security measures for protecting personal information. (Practice 3: Recognizing and Defining Computational Problems) Students will identify personal information and describe physical and digital measures for protecting it. Examples of physical security may include a lock on the door, a safe, covering the camera on your device. Examples of digital security may include virus protection software, , strong passwords, biometric scanners (e.g., fingerprint, facial recognition), etc.	✅
3.NI.2 Develop personal patterns of behavior to protect information from unauthorized access. (Practice 4: Developing and Using Abstractions) Students will begin to develop habits that protect their personal information. For example, using strong passwords, changing passwords often, logging off devices, etc.	✅
3.DA.1 Organize and present collected data visually to highlight relationships and support a claim. (Practice 7: Communicating About Computing) Students will organize and present data collected using visualizations. For example, draw a scaled picture and scaled bar graph to represent data, with several categories. Gathering data may be used as an instructional strategy, but it is not required of students.	✅
3.DA.2 Use data to communicate ideas, highlight relationships, and predict outcomes.	✅

(P7.1) Students will use data to communicate ideas to emphasize relationships and predict outcomes. For example, using a scaled bar graph, students will predict what flavors of ice cream will be most popular among the third grade population.	
3.AP.1 Create programs that include events, sequences, loops, and simple conditionals to express ideas or address a problem. (Practice 5: Creating Computational Artifacts) Students will create programs using an elementary block coding program (e.g. ScratchJr.) that include events, sequences, loops, and simple conditionals to complete a task. The new components for third grade are events (starting your computer and having applications automatically start) and simple conditionals (if you click on the character then the character jumps 3 times).	✅
3.AP.2 Modify a previously created program that uses variables to store and modify data. (Practice 5: Creating Computational Artifacts) Students will save and modify data of previously created programs that use variables. For example, students can take an existing elementary block coding program (e.g. ScratchJr.) that collects what time you get up for school in the morning and modify it to collect what time you get home from school in the afternoon.	✅
3.AP.3 Test and debug a program or algorithm to ensure it accomplishes the intended task. (Practice 6: Testing and Refining Computational Artifacts) Students will test and make corrections (debug) to verify programs (an existing elementary block coding program or a recipe) run properly, similar to proofreading writing. The focus is on testing all aspects of a program before beginning the debugging process.	✅
3.AP.4 Perform different roles when collaborating with peers during the design, implementation, and review stages of program development.	✅

(Practice 2: Collaborating Around Computing) Students will collaborate, in a variety of roles, in the program development process (design, implementation, and review). This builds on the team or peer programming from the previous year. The students will take steps to define and select roles, as well as trading roles during the project to learn different aspects of collaboration in computer science. For example, roles may include being the navigator, developer, time manager, quality control, etc.	
3.AP.5 Use an iterative design process to plan and develop a program by considering the perspectives and preferences of others. (Practice 1: Fostering an Inclusive Computing Culture and Practice 5: Creating Computational Artifacts) Students will understand the process of planning (key features, time and resource constraints, and user expectations) before developing a program. Once the program is created, they will review the program with another team for feedback before revising (iterating) and creating an improved program.	✅
3.AP.6 Create programs by incorporating smaller portions of existing programs to develop something new or add more advanced features. (Practice 4: Developing and Using Abstractions and Practice 5: Creating Computational Artifacts) Students will incorporate pre-established programs into their original draft. The existing program will only address part of the necessary solution, requiring students to develop and add new code to achieve the desired outcome. For example, using an existing program that collects data on student lunch preferences, and adding a feature that directs the program to display the class data at the end of the program.	✅
3.IC.1 Evaluate how computing technologies have changed the world, and express how those technologies influence, and are influenced by, cultural practices.	✅

(Practice 3: Recognizing and Defining Computational Problems) Students will evaluate how the advances in technology have impacted society and analyze how those technologies have influenced culture. For example, students may consider how the use of headphones has changed the world and consider societal changes such as how people wearing headphones may not engage in conversations while waiting for public transportation, but also have access to voice translation when speaking with people in different languages.	
3.IC.2 Describe reasons creators might limit the use of their work. (Practice 7: Communicating About Computing) Students will describe piracy and copyright and why owners limit the use of their work.	✅
3.CT.1 Decompose problems into smaller manageable tasks which may themselves be decomposed. (Practice 3: Recognizing and Defining Computational Problems) Students will consider a broader challenge, such as improving the classroom recycling program, and identify and decompose the problem into smaller tasks such as signage, education, using different receptacles for paper vs. plastic, etc.	✅
3.CT.2 Recognize common patterns between problems and recurring patterns within problems. (Practice 3: Recognizing and Defining Computational Problems) Students will be able to recognize common patterns within problems, such as the challenges of accommodating all students and parents at school drop off and similarities with challenges of the entire school eating lunch at the same time. After identifying the similarities in challenges, students can brainstorm other problem scenarios that share in these patterns.	✅
4.CS.1 Demonstrate how computer hardware and software work together as a system to accomplish tasks. (Practice 4: Developing and Using Abstractions) Students will describe and model how computing devices connect to	✅

other devices or components (physical or wireless) to create a system. For example, the relationship between the respiratory and circulatory system during physical activity serves as a metaphor for how the parts of a computer connect to allow input, processing, and output.	
4.NI.1 Model how information is broken down into smaller pieces called packets and transmitted through multiple devices over physical or wireless paths and reassembled at the destination. (Practice 4: Developing and Using Abstractions) Students will learn and model different pathways information travels to and from devices. For example, students will have a set of tennis balls (packets) that have information on each one. Students will hit tennis balls one at a time over the net. If the ball does not clear net, that represents packet loss and represents message not being sent. Tennis balls that clear the net will then be reassembled to deliver message.	✅
4.DA.1 Select, organize, and categorize data and represent that data visually to provide clarity or support a claim. (Practice 7: Communicating About Computing) Students will organize and present data collected using visualizations. For example, when working with a data set of popular songs, data could be shown by genre or artist. Graphs, charts, and infographics can all represent the statistical characteristics of the data. An additional visualization may include making a line plot using provided data sets; include a horizontal scale, title, labels, and straight columns of symbols to represent the data points (• or X).	✅
4.DA.2 Use data to highlight and propose relationships, predict outcomes, or communicate ideas. (Practice 7: Communicating About Computing) Students will use data to communicate ideas to emphasize relationships and predict outcomes. For example, demonstrating irrelevant data connections such as predicting age by eye color or predicting the outcome of an election by	✅

polling only a few people.	
4.AP.1 Compare and refine multiple algorithms for the same task, using computer and non-computer languages, and determine which is the most appropriate. (Practice 3: Recognizing and Defining Computational Problems and Practice 6: Testing and Refining Computational Artifacts) Students will collaborate, in a variety of roles, in the program development process (design, implementation, and review). This builds on the team or peer programming from the previous year. The students will take steps to define and select roles, as well as trading roles during the project to learn different aspects of collaboration in computer science. For example, roles may include being the navigator, developer, time manager, quality control, etc.	✅
4.AP.2 Create programs that include events, loops, and conditionals. (Practice 5: Creating Computational Artifacts) Students will create a set of instructions (a program) that include events, loops, and conditionals to facilitate and manage tasks. Students will create programs using an elementary block coding program (e.g. ScratchJr.) that include events, sequences, loops, and simple conditionals to complete a task. Event examples include mouse clicks, typing on the keyboard, and collisions between objects. Conditional statements are sets of commands that are tied to specific actions based on whether the condition evaluates to TRUE or FALSE.	✅
4.AP.3 Decompose problems into smaller, manageable tasks which may be then be broken down further. (Practice 3: Recognizing and Defining Computational Problems) Students will decompose a program into smaller, more manageable parts. For example, decomposition at this level is creating an animation by separating a story into different scenes. For each scene, a background needs to be selected, characters placed, and actions programmed. The instructions required to program each scene may be like instructions in	✅

other programs.	
4.AP.4 Test and debug a program or algorithm to ensure it accomplishes the intended task. (Practice 6: Testing and Refining Computational Artifacts) Students will test and make corrections (debug) to verify programs run properly.	✅
4.IC.1 Evaluate computing technologies that have changed the world and express how those technologies influence and are influenced by cultural practices. (Practice 3: Recognizing and Defining Computational Problems) Students will evaluate how the advances in technology have impacted society and explain how those technologies have influenced culture. For examples, students can investigate the evolution of a technology (such as cameras, phones, or audio devices) and discuss the impact of those changes.	✅
4.IC.4 Propose ways to improve the accessibility and usability of technology products for the diverse needs and wants of users. (Practice 1: Fostering an Inclusive Computing Culture) Students will reference current technology and diverse user needs to brainstorm, collaborate, and propose innovative (new) technologically accessible ideas. For example, students may investigate voice-to-text, translation to other languages, and adaptive devices.	✅
4.CT.1 Determine specific aspects of patterns between or within problems that can be abstracted out to leave only the common or important elements. (Practice 3: Recognizing and Defining Computational Problems and Practice 4: Developing and Using Abstractions) Students will determine patterns within problems to identify core elements. Students will seek to identify key strategies to address the core elements, and then build a solution to address the comprehensive problem. For example, when the school is purchasing recess equipment,	✅

the students can identify possible challenges and problems that may exist for their community. Students can identify how to address those problems individually, then create a comprehensive solution to make sure recess is a success.	
5.CS.1 Create potential solutions to solve hardware and software problems using common troubleshooting strategies. (Practice 4: Developing and Using Abstractions and Practice 6: Testing and Refining Computational Artifacts) Students will find common hardware and software troubleshooting solutions. For example, checking power source, restarting programs and/or device, checking physical and wireless connections, etc.	✅
5.NI.1 Model how information is broken down into smaller pieces, transmitted as packets (data groups) through multiple devices over networks and the Internet, and reassembled at the destination. (Practice 4: Developing and Using Abstractions) Students will understand the functional use of routers and switches to send packets across multiple paths for communicating information to its destinations (such as wired connections, Wi-Fi, light (fiber optics), etc.). For example, students may create diagrams, models, written explanations, presentations etc. to demonstrate their understanding of the concept of transmitting packets.	✅
5.DA.1 Explain how the amount of space required to store data differs based on the type of data and level of detail and that the utility of that data varies. (Practice 7: Communicating About Computing) Students will be able to explain that text files are smaller than picture files which are smaller than movie files. Additionally, students will be able to identify the utility of different data types, such as how you can use numerical data vs. alpha- numeric data in your analysis. For example, a number variable can only store a number and not letters. An alphanumeric variable can store numbers or letters but cannot be added	✅

or subtracted because it is text.	
5.DA.2 Organize and share collected data visually to highlight relationships and support a claim. (Practice 7: Communicating About Computing) Students will be able to refer to organized data when communicating an idea. For example, students may make a line plot about how many students share a length of their shoe size in inches using provided data sets; include a horizontal scale, title, labels, and straight columns of symbols to represent the data points (• or X).	✅
5.DA.3 Prioritize, analyze and use data to communicate ideas, highlight relationships and predict outcomes. (Practice 7: Communicating About Computing) Students should will be able to select and prioritize relevant data from large or complex data sets in support of a claim or to communicate the information in a more sophisticated manner. For example, looking at a data set of earthquakes over the past 20 years to determine what data is relevant to predict a future earthquake.	✅
5.AP.1 Compare and refine multiple algorithms for the same task and determine which is the most appropriate. (Practice 3: Recognizing and Defining Computational Problems and Practice 6: Testing and Refining Computational Artifacts) Students will compare different algorithms that achieve the same result, and determine which algorithm is more appropriate. For example, students will compare different ways to get ready in the morning before school or which is the best route to get to the lunchroom.	✅
5.AP.2 Decompose problems into smaller, manageable tasks which may themselves be deconstructed and analyzed. (Practice 3: Recognizing and Defining Computational Problems) Students will decompose problems into smaller tasks to complete said task. For example, students creating maps of counties to compose a full state map or students working in teams to create acts to put on a	✅

Skill Struck ®

three-scene play.	
5.AP.3 Create programs by incorporating smaller portions of existing programs, to develop something new or add more advanced features. (Practice 4: Developing and Using Abstractions and Practice 5: Creating Computational Artifacts) Students will create a new program, based on portions of existing programs. For example, teacher gives a writing prompt where students create an animation and design alternative endings.	✅
5.AP.4 Use an iterative process to plan and develop a program by considering the perspectives and preferences of others. (Practice 1: Fostering an Inclusive Computing Culture and Practice 5: Creating Computational Artifacts) Students will plan and develop a solution for another person's problem. For example, a student has a hard time completing homework. The team designs a solution for how to manage time in order to complete homework, gathers data on the new solution, and revises the solution.	✅
5.AP.5 Recognize and observe intellectual property rights and give appropriate attribution when creating, remixing, or combining programs. (Practice 5: Creating Computational Artifacts and Practice 7: Communicating About Computing) Students will explain the concepts of ownership and sharing and be able to cite the owner. For example, a downloaded image may have restrictions that prohibit modification of an image or using it for commercial purposes.	✅
5.AP.6 Describe choices made during program development using code comments, presentations, and demonstrations. (Practice 7: Communicating About Computing) Students will describe the process used to develop a program using comments, presentations, and demonstrations. For example, students will be able to explain their selection of controlled variables in a science	✅

Skill Struck ®

investigation.	
5.IC.1 Propose ways to improve the accessibility and usability of technology products for the diverse needs and wants of users. (Practice 1: Fostering an Inclusive Computing Culture) Students will propose improvements of current technology based on needs and wants of a user. For example, having programs read in multiple languages, modifying hardware to meet the needs of a user, etc.	✅
5.IC.2 Seek and explain the impact of diverse perspectives for the purpose of improving computational artifacts. (Practice 1: Fostering an Inclusive Computing Culture) Students will research and explain how computing technologies influence, and are influenced by, cultural practices. For example, looking at school website and diverse student and parent needs to improve upon the website design and layout.	✅
5.CT.1 Develop algorithms in computer programs to solve problems, including unique and repeated sub-tasks within a larger program. (Practice 3: Recognizing and Defining Computational Problems and Practice 5: Creating Computational Artifacts) Students will research and explain how computing technologies influence, and are influenced by, cultural practices. For example, looking at school website and diverse student and parent needs to improve upon the website design and layout.	✅