Revised PK–12 Computer Science Standards

DRAFT 3.0

SUGGESTED CITATION: Computer Science Teachers Association. (2025). *Revised PK–12 computer science standards*: *Draft 3.0*. https://csteachers.org/k12standards/revision/

To access a progression view of the foundational standards, see:
https://csteachers.org/PK-12-Foundational-Standards-Draft-3.0-Progression-Chart

Table of Contents

Vision

Every Student Prepared for a World Powered by Computing

Today's students will face pervasive questions that require foundational knowledge of computer science (CS) for them to answer. They will, for example, need to shape their views on the regulation of artificial intelligence, have the ability to automate routine tasks, and analyze and visualize data in a variety of contexts. These situations illustrate the need for early, universal CS education, which will only become more important as society continues to increasingly rely on computing technologies.

In a world increasingly powered by computing, students of all identities and chosen career paths need quality computer science education to become informed citizens and confident creators. Our vision for PK–12 CS education is to ensure:

- All students are engaged and supported in learning CS, including its impacts on individuals, societies, cultures, democracies, and policies.
- Policies, pedagogies, and practices support all students learning CS.
- Standards align with the current and future needs for learning CS.

Defining Computer Science

Computer science is the study and creative practice of how people use data, algorithms, and computing systems to solve problems, make discoveries, and express ideas.

Computer science connects scientific reasoning with creative expression. It helps people make sense of and shape a world increasingly powered by computing. As a discipline, computer science brings together algorithms, data, systems, and design to help learners ask questions, explore patterns, model complex ideas, and build solutions that matter in their communities. It is both a science and an art: a field grounded in computational thinking and driven by human curiosity, imagination, and purpose.

As a field, computer science is constantly evolving. It includes foundational ideas—such as algorithms, programming, and systems—that enable people to represent and process information, as well as specialty areas like artificial intelligence, data science, cybersecurity, and physical computing. Across all of these, students examine how computing interacts with and evolves alongside society, considering issues of ethics, equity, and sustainability.

The CSTA PK–12 Computer Science Standards are designed to make this vision concrete. They organize learning through five **concepts** (Algorithms & Design, Programming, Data & Analysis, Systems & Security, and Computing & Society) supported by four cross-cutting **pillars** (Ethics & Social Responsibility, Inclusive Collaboration, Computational Thinking, and Human-Centered Design). Together, these structures emphasize that computer science is not only about understanding how technologies work but also about how people design, evaluate, and use them to express ideas, make discoveries, and solve problems across a variety of disciplines and contexts.

Through this learning, students develop **dispositions** (habits of mind) that support lifelong learning and creating with computing (curiosity, creativity, persistence, critical thinking, reflectiveness, resourcefulness, and a sense of belonging). They learn that not only is computer science a powerful medium for expression and innovation, but also a shared responsibility for shaping communities. **In a world increasingly powered by computing, computer science empowers all learners as critical consumers, responsible creators, and informed participants in society.**

The following standards translate this vision into specific learning outcomes for all students across all grade levels and contexts.

Computer Science Is:

- **a scientific and creative discipline** focused on understanding and designing algorithms, data practices, and computing systems.
- **the application of computational thinking** to develop both rules-based and data-driven solutions across a variety of disciplines and contexts.
- **a collaborative discipline**, where learners plan, communicate, test, and refine ideas together to design solutions that serve diverse people and communities.
- **an ethical and human-centered practice**, where people examine impacts, identify potential harms and benefits, and design responsibly.

Computer Science Is Not:

- **focused on keyboarding or using technology** such as word processors, slide decks, spreadsheets, or generative AI tools. These tools *are* used in CS instruction but also across other subject areas.
- **limited to coding or programming.** Programming is an essential part of CS, but CS is far broader.
- **limited to any one career path.** CS empowers creativity, research, problem solving, and innovation across every field—from the sciences and arts to health, business, and public service.

About the CSTA PK–12 Standards

The CSTA PK–12 Computer Science Standards define the essential knowledge, skills, and dispositions to prepare all students for a world powered by computing. Specifically, the Standards delineate coherent progressions of student learning outcomes from pre-kindergarten to grade 12. Together, they form the strong foundation for a rigorous and comprehensive computer science (CS) curriculum that is driven by research and informed by teacher practice.

Written *by* teachers *for* teachers, the Standards offer flexibility and guidance to support state and local adaptation, while also promoting instructional coherence across the U.S. and internationally. Grounded in research and equity, these standards center creativity, ethics, data, and human-centered design. They integrate emerging technologies like artificial intelligence while reinforcing foundational computing concepts and inclusive practices that make computer science relevant for every learner. The Standards are a primary resource for state and local education agencies when determining what PK–12 students need to know and be able to do in CS. Widespread adoption impacts millions of students and promotes consistency in state policy, curriculum development, teacher certification, and teacher preparation and professional development across the U.S. and beyond.

Intended Uses

The 2026 CSTA PK–12 Standards are designed for broad adoption and implementation. While many in the CS education community will find value in the Standards, likely use cases for the standards are:

- **PK–12 CS teachers** will use the Standards to design rigorous and relevant learning experiences for all students.
- **Other PK–12 teachers** will use the Standards to plan how they can integrate computer science into other subject areas.
- **PK–12 administrators** will use the Standards to establish and support district- and school-level policies that enable implementation and increase participation.
- **Curriculum providers** will use the Standards to develop new, or refine existing, curriculum and associated tools that guide student learning.
- **PD providers** will use the Standards to design in-service professional learning programs that prepare teachers to effectively teach CS across a variety of contexts.
- **Schools of education** will use the Standards to ensure pre-service teachers have appropriate knowledge and skills to teach foundational CS to their future students.
- **State leaders** will use the Standards to inform the adoption of state CS standards, teacher certification, and other CS education policy decisions.

Revision Overview

The CSTA PK–12 Standards, last published in 2017, required an update due to increased CS implementation in PK–12 schools, continuously evolving research, and recent technological advancements. A three-year revision process began with research in fall 2023, followed by the formal writing process in fall 2024.

CSTA defined the following values to guide the revision process and provide a lens for reflecting on and refining project outputs:

Equity-centered: Promotes broad and equitable access, participation, and experiences in CS education among all students.

Community-generated: Meets the needs of the community, including PK–12 educators, postsecondary institutions, students, parents, and industry.

Future-oriented: Anticipates future needs of current learners, and prepares them for a future that is increasingly reliant on computing.

Grounded in research: Reflects the evolving body of knowledge of how students learn CS.

Flexible in implementation: Considers multiple pathways for meeting individual needs of learners, including regional, cultural, ability, social, and economic factors.

People

The standards revision process involved the coordinated efforts of three main groups.

- **Standards Writing Team:** Included 24 individuals representing 18 states that serve in a variety of roles including CS teacher (44%), state CS specialist (16%), teacher preparation (32%), and researcher (8%). Collectively, the writing team had over 400 years of teaching experience, more than 250 of those years specifically teaching CS. Writers took the lead on drafting standards and made key decisions throughout the writing process.
- **Advisory Board:** Composed of approximately 70 accomplished individuals from across the CS education community including state CS supervisors, district leaders, curriculum and PD providers, nonprofit leaders, international partners, researchers, school of education faculty, and industry partners. Advisors provided additional insight and perspective that helps to ensure a useful, viable, and accessible final product.
- **Asynchronous Reviewers:** Over 500 asynchronous reviewers validated directionality and provided feedback at key intervals to inform the writing process.

Process

The standards revision process was organized around three distinct and overlapping phases: Research, Writing, and Implementation.

Research

The research phase began with the *Reimagining CS Pathways* project, which aimed to answer two research questions:

1. What CS content is essential for all high school graduates to know?
2. What pathways should exist to continue learning beyond the foundational high school content?

This project laid the foundation for identifying priorities for the updated standards and executing additional research efforts to inform those priorities. Additional research efforts intended to inform the structure and content of updated standards included:

- Comparing state and international CS standards
- Conducting literature reviews on K–12 algorithms, programming, data and analysis, cybersecurity, and the history of computing
- Comparing the 2017 CSTA Standards to other sets of standards and frameworks (in CS-related and non-CS areas)
- Identifying AI Priorities for All K–12 Students
- Amplifying Social Impacts of Computing Standards

Writing

The writing phase kicked off in September 2024 and included six in-person convenings and regular synchronous meetings with the writing team. Writers primarily operated in grade band teams when engaging in drafting sprints; however, concept teams and other cross-grade band groupings also worked to ensure logical progressions across grade levels. Advisors, researchers, and technical writers provided just-in-time feedback throughout the writing process to assist writers in making key decisions.

High-Level Writing Timeline

Time Period	Steps Taken
Fall 2024	• Determined organizational structure of standards and drafted all standards within the Systems & Security concept.
Winter 2024–25	• Released Draft 1.0 of standards including one fully drafted concept (Systems & Security). • Collected feedback on organizational structure and general approach to writing.
Spring 2025	• Drafted standards in the remaining four concepts.
Summer 2025	• Released Draft 2.0 of standards including fully drafted progressions across all five concepts. • Collected feedback on coherence and clarity of progressions. • Refined standards based on feedback.
Fall 2025	• Drafted clarification for each standard including boundary statements, practice alignment, implementation examples, and interdisciplinary connections.
Winter 2025–26	• Released Draft 3.0 of standards including boundary statements and practice alignment. • Collected feedback on clarity of individual standards and associated boundary statements and practice alignments. • Refined standards and clarifications based on feedback.
Spring 2026	• Finalized standards and clarifications. • Planned web and print designs.
Summer 2026	• Published the 2026 CSTA PK–12 Standards.

Implementation

Implementation of the updated CSTA PK–12 Standards will begin with the official launch of the standards in summer of 2026. This phase is focused on supporting teachers, teacher leaders, curriculum and professional development providers, and state departments of education to strategize, create structures, and develop student learning experiences that align with the standards.

Priorities for the 2026 CSTA PK–12 Standards

Enduring Priorities

Foundational ideas in CS that remain a priority in the updated standards include Algorithms, Programming, Data & Analysis, and Computing Systems.

Algorithms

Algorithms receive a heightened emphasis in the 2026 CSTA PK–12 Standards as compared to previous iterations due to the profound impact of emerging technologies such as generative AI on society. To reflect this, Algorithms & Design is defined as a distinct concept, as opposed to being combined with Programming previously. The elevated position of Algorithms is underscored by designating Computational Thinking as a crosscutting pillar to be woven throughout instruction.

Programming

Programming is still prioritized, and the value of learning to program remains a core aspect of foundational CS. However, the updated standards represent a shift away from purely generating code to a balanced approach inclusive of developing skills in reading, evaluating, modifying, and debugging code.

Data & Analysis

Content related to data and its analysis is also a priority, reflecting the increased prevalence of data in daily aspects of life as well as the vast amount of data upon which emerging AI technologies are built. This trend also acknowledges data science as a burgeoning and increasingly important field with strong foundations in CS. Data & Analysis is one of five concepts to reflect this prioritization.

Computing Systems and Security

The 2017 CSTA K–12 Standards included (1) Computing Systems and (2) Networks and the Internet as two distinct concepts. In an effort to streamline content and acknowledge the interconnectedness of these two areas, they are represented by a singular concept called Systems & Security in the updated standards. This also draws attention to the increasing importance of security in the safe and responsible implementation of CS.

Emerging Priorities

In light of recent research and technological innovation, the following areas are either new in this iteration of the CSTA standards or have an elevated presence as compared to previous versions:

History of Computing

To fully grasp their sociotechnical world, students must explore the evolution of computing technologies, from early developments to modern innovations, and recognize the key contributors. This priority area is integrated within the Computing & Society concept.

Career Exploration

Computing is foundational to nearly every industry and field of study. As such, it is critical for students to connect computing to their personal interests and career goals. Career-related standards are included in the Computing & Society concept.

Artificial Intelligence and other Emerging Technologies

In an effort to ensure that the 2026 CSTA PK–12 Standards can guide the implementation of a relevant and comprehensive CS education for the next eight to 10 years, the standards necessarily cover key current technological innovations such as foundational aspects of artificial intelligence. In anticipation of future technological innovations, certain standards are intentionally designed to allow standards implementation to evolve as the field of computer science evolves. In this way, the standards accommodate learning around both current and future tech. AI content is integrated throughout appropriate concepts in the standards, and a distinct emerging technologies progression is also included in the Computing & Society concept, which creates space for additional learning around AI and other emerging technologies that may or may not be in existence today.

Inclusive Collaboration

The prioritization of inclusive collaboration as a pillar highlights equity as a core value. It emphasizes key skills like effective communication, collaborative problem-solving, and efficiently navigating computing projects. The practices that define inclusive collaboration are integrated across all CS instruction.

Human-Centered Design

Human-centered design principles are crucial for the ethical development of computing technologies. The level to which a variety of potential user needs, abilities, and contexts are considered in the design process can either remedy or exacerbate equity issues in implementation and profoundly impact the benefits and harms experienced by users. Like Inclusive Collaboration, these practices are woven throughout all CS concepts.

Ethics and Impacts

Perhaps the most critical priority in the updated standards, ethical practices and impacts of computing content can be found throughout the 2026 CSTA PK–12 Standards. The Ethics and Social Responsibility pillar promotes practices such as using computing for social good and respecting other creators. Impact-related content can be found in multiple subconcepts distributed throughout the standards including Impacts of Algorithms, Impacts of Computing Systems, and Impacts of Data Science. This emphasis aims to cultivate computer science learners who are responsible creators and critical consumers of technology, aware of its impact on their lives and communities.

Specific details related to the philosophy and approach to AI and Ethics in the standards are detailed below.

AI is Part of CS

AI use and interaction is increasingly part of students' day-to-day lives, yet most cannot explain how these systems work, why they fail, or how their data shapes the AI shaping their lives. This knowledge gap will continue to widen if steps are not taken to ensure students have opportunities to learn both with and about AI. CS classrooms are a natural fit for students to learn the technical underpinnings of AI—how it is created, how it makes decisions, and how it impacts society. **Including AI learning outcomes as part of a foundational computer science learning experience bolsters students' abilities to become critical consumers, responsible creators, and informed participants in society.**

Critical Consumers: From the media they consume to the jobs they pursue to their doctor's visits, students will encounter AI in daily routines and activities. They must recognize when AI is being used, question whether it is appropriate for a task, evaluate if outputs are trustworthy, and assess whether decisions are fair.

Responsible Creators: Regardless of career, most students will likely use AI tools in some professional capacity. They need to understand when human judgment should override AI, recognize limitations and harms, and apply ethical frameworks to make decisions.

Informed Participants in Society: Students will need to consider the real benefits and risks of AI in their communities, when and how to leverage outputs from AI in daily tasks, and what the future might look like as AI continues to permeate society. They need to understand how bias in training data creates biased outcomes, why AI makes mistakes, who bears responsibility, and the environmental costs of AI systems.

Without knowledge of how AI works, students cannot fully and meaningfully participate in evaluating consequential technologies nor shape their future iterations.

When students learn and understand that AI is created by humans, trained on human-collected data, and reflects human choices, their understanding of a world increasingly powered by computing shifts:

- They understand mistakes aren't mysterious failures but predictable consequences of training data gaps
- They recognize bias stems from human choices about data and design
- They see themselves as potential shapers of AI's future, rather than passive recipients

AI in the CSTA PK–12 Standards

Because AI technologies are built upon fundamental principles of computer science, the standards writing team decided to integrate AI content throughout the organizational structure rather than define AI as its own concept. AI content in the standards serves two primary purposes: 1) to help students understand how AI works and 2) to encourage students to grapple with real impacts and ethical issues related to AI technologies, their creation, and their deployment. Specifically, AI content is represented **across the standards** in the following ways:

- Algorithms & Design: Rule-based vs. data driven approaches, machine learning, and societal impacts (e.g., bias)
- Programming: Reading and evaluating code generated by AI
- Data & Analysis: data fluency and data analysis
- Systems & Security: security implications and environmental impacts
- Computing & Society: history of computing, humans and AI, and emerging technologies

PK–12 students deserve to understand the systems shaping their futures, and learning experiences aligned with the AI content included in the 2026 CSTA PK–12 standards will help them to do so.

A Sociotechnical Approach to Ethics and Impacts of Computing

As computing becomes woven into every aspect of our lives—from the tools we use to communicate to the systems that shape our opportunities—students need more than technical skills. They need to understand how computing and society shape one another. This understanding lies at the heart of the revised CSTA PK–12 Computer Science Standards: preparing all students not only to **use and build technology**, but to **question, imagine, and influence** its role in the world.

Moving Beyond Techno-Myths

Too often, we hear that technology is neutral, inevitable, or outside of human control. These "techno-myths" suggest that computing systems simply evolve on their own. But every technology reflects human decisions—by designers, companies, and communities—and carries the values and power structures of those decisions.

Computer science education can help students replace myths with **sociotechnical understanding**: the recognition that technology and society are inseparable. Students learn that computing systems are not just technical artifacts but products of cultural, political, and economic choices. This prepares them to ask deeper questions, like: *Who benefits from this technology? Whose needs are ignored? What values are embedded in its design?*

Understanding and Addressing Computing's Harms

When we talk about "ethics in computing," it is easy to focus only on isolated examples of bad actors or unintended consequences. A sociotechnical approach encourages students to see **patterns and systems**—how inequities, biases, or environmental impacts arise from both the technology itself and the contexts in which it is built and used.

Students can explore real-world issues such as biased algorithms in hiring, surveillance in schools, or environmental effects of data centers. They examine **types of harm** (like discrimination or misinformation), **mechanisms** (how biased data or business incentives drive those harms), and **mitigation strategies** that range from responsible design to policy and advocacy. In doing so, students develop agency—seeing that harms are not inevitable and that collective action can shape better futures.

Doing, Not Just Discussing: Ethical and Critical Computing Practices

Understanding is only half the goal; the other half is practice. Ethical, responsible, and critical computing becomes meaningful when students *do* it. This means engaging in a range of practices:

- **Ethical design:** creating technologies that reflect values like fairness, privacy, and accessibility.
- **Critical inquiry:** investigating how technologies impact people and the environment—and whether certain technologies should even exist.
- **Hopeful reimagining:** envisioning alternative, more just futures for technology.
- **Advocacy:** recognizing when to push back, speak up, or refuse technologies that cause harm.
- **Responsible use:** making thoughtful decisions about personal and communal use of technology.

These practices move computing education beyond coding for its own sake, helping students connect computing to real-world issues they care about.

Ethics as Dialogue, Not Doctrine

There is no single "right" answer to what is ethical in computing. Instead of prescribing moral rules, educators can help students navigate competing perspectives and values. Some classrooms might focus on identifying harms and responsibilities; others may examine the values built into technologies or explore multiple ethical frameworks. Across these approaches, students learn to reason, debate, and deliberate—to recognize that **ethical computing is a process of ongoing reflection and dialogue**.

Why This Matters

Reimagining computer science education through this sociotechnical lens does not make CS less rigorous—it makes it more relevant, engaging, and empowering. Students see computing not as a distant or predetermined field, but as something they can shape. They learn that coding and critical thinking go hand in hand, and that the most powerful computing education is one that connects knowledge to justice, creativity, and community.

This approach invites all educators—not just computer scientists—to join in preparing students to participate fully in our digital world: **as designers, users, and changemakers**. Together, we can help them see that while AI may be the headline, *human choice and collective responsibility are the story.*

Additional Context for Reviewers

This document represents the third of three major drafts before CSTA publishes the updated standards in the summer of 2026. Feedback on this draft will inform how we refine the language and content of the standards, related clarifying information, and other supports. At this stage, high level feedback (e.g., related to progressions, structure) is welcome. However, feedback that critiques the clarity, specificity, and measurability of individual standards, as well as the utility of the clarifying information, is most helpful.

The writing team is currently using the following assumptions as guides throughout the writing process to help determine the quantity, depth, and breadth of standards:

- Students will experience a certain amount of instructional time at each grade band:
  - » Elementary (Grades PK–5): 20 to 40 hours per year (or 30 to 60 minutes per week)
  - » Middle school (Grades 6–8): the equivalent of one yearlong course
  - » High school (Grades 9–12): the equivalent of one yearlong course
- Implementation will vary and may include discrete courses and/or integration in other subject areas.
- Students will experience the full vertical progression (i.e., students learn the content in the PK–5 standards before entering middle school and learn the content in the 6–8 standards prior to entering high school).

While these assumptions may not reflect the current reality of CS instruction in all schools, they represent an aspirational target. We ask reviewers to focus on the clarity, specificity, and measurability of individual standards and their clarifying information. This will help the writing team to further refine the standards ahead of their official publication.

CSTA and the standards writing team look forward to reviewing your valuable insights as we work to define the future of PK–12 CS education.

Navigating the Standards

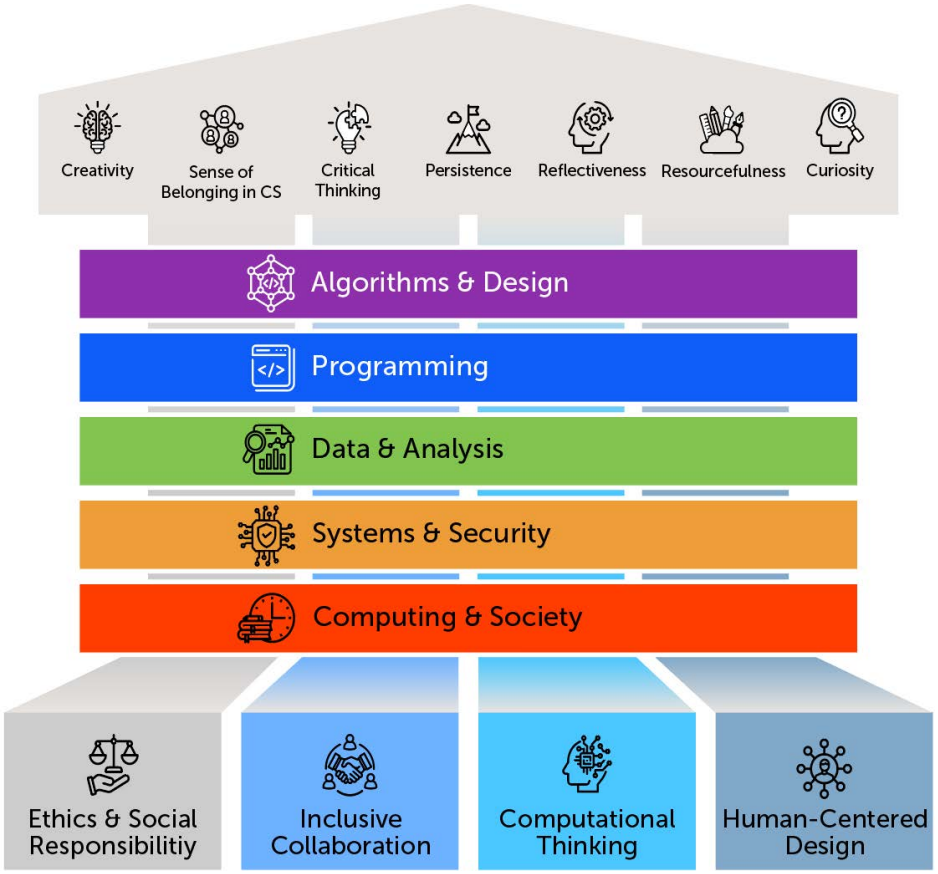
Foundational Standards

The foundational PK–12 Computer Science Standards are structured to provide coherent learning progressions across the PK–12 continuum. These standards begin with a PK and Kindergarten (PK/K) band followed by discrete grade-level expectations for students in grades one through five. The middle school standards are grouped into a single grade band, which then leads into a final grade band that covers foundational high school standards.

We organized the foundational standards for pre-kindergarten through high school around three primary components: concepts, pillars, and dispositions. Concepts serve to organize standards by content. The five concepts are: (1) Algorithms & Design, (2) Programming, (3) Data & Analysis, (4) Systems & Security, and (5) Computing & Society. We recognized artificial intelligence (AI) as a priority during the standards revision process. AI-related content is distributed across the five concepts instead of being a discrete concept.

Pillars consist of Practices that cut across all concepts. The four pillars are: (1) Ethics & Social Responsibility, (2) Inclusive Collaboration, (3) Computational Thinking, and (4) Human-Centered Design.

Dispositions are habits of mind fostered within CS classrooms and developed through instruction that includes the concepts and pillars. The highest-priority dispositions within the CS context are: creativity, sense of belonging in CS, critical thinking, persistence, reflectiveness, resourcefulness, and curiosity.



The draft PK–12 standards within this organizational structure are foundational for all students. We adapted this structure from the *Reimagining CS Pathways: High School and Beyond* project, which aimed to create a community definition of what a foundational CS learning experience for all high school students includes and possible CS learning opportunities beyond that foundation. The graphic above provides a visual representation of the relationship between concepts, pillars, and dispositions.

Once students establish a strong foundation in computer science, they can pursue specialized learning. In support of this, sets of Specialty Standards have been developed to articulate learning objectives for **high school students who have completed foundational computer science education** and elect to pursue deeper study in specific computing domains. These standards are designed to support a transition from foundational CS for all students to specialized postsecondary readiness, leading to enrollment, employment, or enlistment.

Specialty Standards

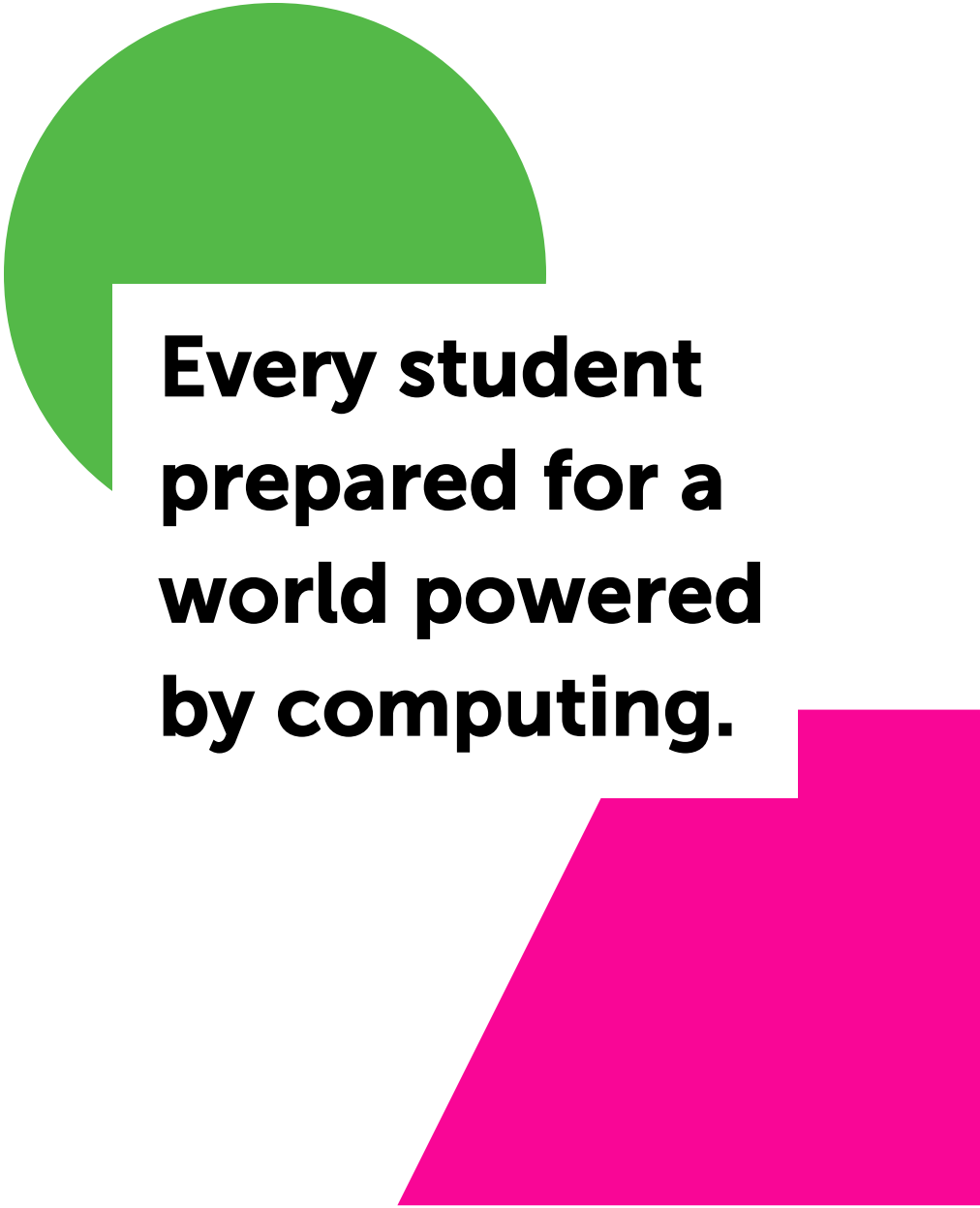
Specialty Standards define **advanced, domain-specific learning** beyond foundational PK–12 CS content. The Standards are organized into two tiers (Specialty I and Specialty II) across six high school specialty areas identified through the *Reimagining CS Pathways* project:

- Software Development
- Cybersecurity
- Data Science
- Physical Computing
- Artificial Intelligence (AI)
- Game Development

Specialty I standards cover the introductory knowledge and skills essential to a chosen specialty area, serving as the first dedicated learning experience in that domain. **Specialty II** standards describe advanced study within the specialty area, designed to prepare students for college-level coursework or industry-level certifications.

Additionally, one level of **X + CS Standards** has been developed to guide the integration of foundational high school CS content into other subject areas, like Journalism or Biology.

Throughout the standards, students attend to the societal and environmental impacts of computing. We use the term societal to encompass social, governmental, political, cultural, and economic factors.



Components of a Standard

In addition to the standards and their associated identifiers, clarifying information accompanies each standard to provide additional guidance on what is included and how a standard might be implemented. Brief descriptions of clarifying components are found in the table below. A subset of these components will be available in the final print version of standards. The remaining components will be accessible through the interactive web interface.

Component	Description
Boundary Statement(s)*	Boundary statements are elaborations on the standard language including important nuances and clarifications that are critical for understanding the full intent of the standard, including what is and is not expected. Boundary statements ensure that those interpreting the standards can understand what the standard means within the context of the particular grade level/band, including what is explicitly considered out of scope. Boundary statements ensure that content remains grade level/band appropriate and that the focus of instruction is placed as intended.
Pillar and Practice Alignment*	Each standard is tagged with one or two computer science practices and their associated pillars. This information provides framing and guidance that informs instructional design decisions and highlights how students should engage with CS content.
Disposition Alignment*	Each standard is tagged with one to three dispositions. This information signals the attitudes and habits of mind that are developed through thoughtfully designed instructional experiences.
Progressions	Progressions contextualize the standard among adjacent grade levels/bands and help to clarify what students will have ideally learned prior to the given grade level/band and what the current standard is building toward in future years. Progression views are available in the interactive web display, as well as via a separate downloadable document.
Implementation Example(s)	One or more implementation examples for each standard include high-level descriptions of activities that can be implemented with students to address the standard. Where possible, multiple examples with differing approaches (e.g., unplugged and computer-based) are included.
Interdisciplinary Connections	Interdisciplinary connections provide examples of implementation strategies within the context of a non-CS or CS-related discipline. The identification of strong connections with the Next Generation Science Standards, Common Core State Standards for Mathematics, and Common Core State Standards for English Language Arts were prioritized. Additional connections with other sets of disciplinary standards such as social studies, the arts, and cybersecurity are also available.
Resources	This component includes vetted instructional resources that can be tailored and used to teach or assess the standard.
Academic Vocabulary	Academic vocabulary are terms that are explicitly referenced in the standard itself and are critical to the development of knowledge within the given concept. These terms are defined as part of the accompanying glossary and are integrated into the interactive web view of the standards.

* Components that are viewable in full in this document.

Example of All Components for a Security Standard

While draft boundary statements and pillar, practice, and disposition alignment are available for all standards in this document, the following is an example of most of the clarifying components that will be available at standards publication (aside from progressions and resources).

Standard:

MS-SAS-32: Explain the effects of failing to use the CIA (Confidentiality, Integrity, Access) Triad.

Boundary Statement(s):

Students should recognize and explain what happens when each element of the CIA Triad is not maintained. When confidentiality is compromised, sensitive data is exposed (e.g., leaked passwords, shared personal information). When integrity is compromised, data becomes corrupted or altered (e.g., incorrect grades in a school database). When availability is compromised, systems become inaccessible (e.g., a denial-of-service attack blocking a website). Students should apply the CIA Triad to age-appropriate, relatable contexts (e.g., school accounts, social media, or online gaming, demonstrating understanding of how failures connect to real-world impacts).

Students are not expected to conduct professional-level security audits or design encryption protocols. Students do not need to develop deep cryptography knowledge.

Pillar(s) and Practice(s):

- Inclusive Collaboration: 3. Communicate effectively about computing.
- Inclusive Collaboration: 5. Act Responsibly in computing collaborations.

Disposition(s):

Critical Thinking, Reflectiveness

Progressions:

Grade 5	Middle School	High School
E5-SAS-13: Describe the concepts of the CIA (Confidentiality, Integrity, Access) Triad and how each part is important in protecting information.	**MS-SAS-32: Explain the effects of failing to use the CIA (Confidentiality, Integrity, Access) Triad.** **MS-SAS-33:** Evaluate common types of cyber attacks, including social engineering and malware, and preventions.	**HS-SAS-32:** Identify different types of cybersecurity and physical security measures and the trade-offs for users, data, and devices. **HS-SAS-33:** Classify the causes and impacts of security breaches and social engineering attacks for individuals, industries, communities, and governments. **HS-SAS-34:** Formulate a solution to a security flaw in a given system.

(example continued on next page)

Implementation Examples

Students could explore failures of the CIA Triad in unplugged and/or computer-based activities.

Unplugged — CIA Failure Case Studies: In small groups students receive scenario cards (e.g., password leaks, altered grades, denial-of-service) and identify which CIA element failed, describe the consequences for users and systems, and propose age-appropriate prevention strategies. Groups present findings and compare cases across contexts such as school accounts, social media, and online gaming. Teacher prompts: "Which CIA element failed? What were the consequences? How could this be prevented?"

Computer-based — Simulated CIA Breakdowns: Teacher-run simulations intentionally demonstrate failures (e.g., overly broad sharing settings to show confidentiality loss; silent modification of a document to show integrity loss; temporary account lockouts to show availability loss). Students work individually or in pairs to identify the specific failure, explain its effect on tasks/users, and recommend safeguards. Wrap-up discussion asks students how those safeguards would change behavior or system settings.

Interdisciplinary Connections

CCSS.MATH.CONTENT.6.RP.A.3: Use ratio and proportional reasoning to solve problems.

Students calculate the percentage of students in class who reuse passwords across accounts and analyze risks.

CCSS.ELA-LITERACY.W.8.1: Write arguments to support claims with evidence.

Students write a persuasive essay arguing which element of the CIA Triad is most critical to protect in schools.

National Core Arts Standards – Visual Creating VA:Cr1.1.6a: Combine concepts collaboratively to generate innovative ideas for creating art.

Students collaboratively design posters or infographics that illustrate the consequences of neglecting confidentiality, integrity, or availability in real-world scenarios.

MS-ETS1-2: Evaluate competing design solutions using a systematic process to determine how well they meet the criteria and constraints of the problem.

Students evaluate different school-level security measures (two-factor authentication, password resets, backups) to determine which best prevents CIA failures.

C3 Framework – D4.8.6-8: Draw on multiple disciplinary lenses to analyze how a specific problem can manifest itself at local, regional, and global levels over time, identifying its characteristics and causes, and the challenges and opportunities faced by those trying to address the problem.

Students debate whether schools should prioritize confidentiality (protecting student data) over availability (keeping systems always online).

Academic Vocabulary

CIA Triad: A three-part model designed to guide policies for information security within an organization. In this context, **confidentiality** is a set of rules that limits access to information, **integrity** is the assurance that the information is trustworthy and accurate, and **availability** is a guarantee of reliable access to the information by authorized people. (CYBER.org)

Concepts

Algorithms & Design

Overview: An algorithm is a sequence of steps designed to accomplish a specific task. Algorithms are translated into programs, or code, to provide instructions for computing devices. In early grades, students learn about age-appropriate algorithms from the real world. As they progress, students learn about the development, combination, and decomposition of algorithms; the evaluation of competing algorithms; and the difference between traditional algorithms and artificial intelligence/machine learning (AI/ML) algorithms.

The Algorithms & Design standards and the Programming standards are complementary and should be considered in tandem. Algorithms & Design standards focus more on program planning and evaluation, while Programming standards focus more on program implementation.

Subconcept	Overview
Algorithm Fundamentals	Designing algorithms, or step-by-step solutions to a task, are an essential component of CS. In early grades, students identify and create algorithms reflecting tasks in their daily lives. As students progress, they develop more complex algorithms and create visual representations of their solutions.
Problem Solving	While there may be many approaches to addressing a task, optimizing an algorithm can result in more efficient and accurate solutions. As students progress, they begin to evaluate the efficiency and accuracy of computational algorithms running under different conditions and use problem-solving skills to explore algorithms' underlying opaque systems.
Machine Learning	Machine learning is a subfield of artificial intelligence in which computers "learn" from data in order to make decisions without being explicitly programmed to do so. This subconcept includes content that is key for understanding foundational components of artificial intelligence. In early grades, students recognize patterns used by people and machines for decision-making. As they advance, students explore how AI models evolve with new training data, train AI models for classification or prediction, and analyze the relationship between training data properties and AI model output. By high school, students justify the selection of AI algorithms, evaluate training data for quality and bias, and develop AI models for specific tasks using appropriate data and tools.
Impacts of Algorithms and Design	Algorithms can have positive and negative effects on people and society. It is important to evaluate not only how well an algorithm works, but also who it benefits, who it may unintentionally harm, and why. In early grades, students begin by exploring how algorithms can lead to different results for themselves and others. As students progress, they learn to identify possible consequences of algorithmic decisions and they apply human-centered design principles to develop and refine computational algorithms. In later grades, students critically analyze the societal impacts of algorithms, including issues of fairness, equity, accessibility, and bias, and consider how algorithmic systems can shape real-world experiences.

Programming

Overview: Programming controls all computing systems, empowering people to communicate with the world in new ways and solve compelling problems. The development process to create meaningful and efficient programs involves choosing which information to use and how to process and store it, breaking apart large problems into smaller ones, recombining existing solutions, and analyzing different solutions.

The Algorithms & Design standards and the Programming standards are complementary and should be considered in tandem. Algorithms & Design standards focus more on program planning and evaluation, while Programming standards focus more on program implementation.

Notes:

- Many standards in this concept discuss creating or reviewing "code." Not all code is text-based or screen-based. In particular, students in early grades may interface with tangible programming systems. Standards were written considering the programming tools commonly used at each grade level.
- While the other four concepts contain a subconcept addressing societal and ethical impacts, the Programming concept weaves societal and ethical impacts throughout its five subconcepts.

Subconcept	Overview
Programming Fundamentals	Across grade levels, students focus on reading and interpreting code, translating algorithms into code, and understanding programming languages' types, syntax, and semantics. Although programming constructs repeat across this subconcept, their complexity is expected to increase as students advance through grade levels.
Program Development	Programming involves paying attention to the organization and structure of code. In early grades, students strengthen their understanding of how to create programs. As students progress, they focus more on building on existing code, modularizing code, documenting code, and using AI tools to support their programming.
Reading and Documenting Code	In a world where AI assistants can generate code based on a prompt, the ability to read and interpret code is increasingly important. In early grades, students describe how code completes tasks including explaining code functionality, program development steps, and how specific code segments contribute to a program's purpose. They also learn to document programs for clarity and create embedded or external documentation for projects. Later on, students evaluate AI-generated code for accuracy, reliability, and usability.
Testing and Refining Code	Ensuring that code works as intended is key to building reliable programs. In early grades, students focus on identifying and fixing errors in their programs. As students progress, they focus on more complex troubleshooting strategies, optimizing their code for efficiency and usability and assessing the accuracy and bias of AI-generated code.
Data Handling	Understanding how programs structure and store data is necessary for successful programming. In early grades, students focus on identifying and labeling data in their daily lives and in age-appropriate programming languages. As students progress, they learn about and manipulate more complex data types. **Note:** This subconcept focuses on data used while programming. The Data & Analysis concept focuses more on collecting, storing, and analyzing data with the use of computing.

Data & Analysis

Overview: Computers collect and store data so it can be analyzed to better understand the world and make more accurate predictions. The amount of digital data generated in the world is rapidly expanding, so the need to process data effectively is increasingly important.

Subconcept	Overview
Data Fundamentals	Data is generated and collected by people, often using computing technologies such as sensors and other automated systems. Metadata is "data about data." Metadata provides context about data, including its origin, structure, and purpose. In early grades, students learn about different types of data and how data is generated, collected, and organized. As they progress, students gain experience with larger and more varied datasets, learn about more advanced data types and organization structures, and develop data documentation.
Data Processing	Computing devices process data to make it useful for analysis. In earlier grades, students learn how to use computational tools to manipulate data: filtering, grouping, summarizing, transforming, and reshaping data. As students progress, they apply computational methods to: automate data cleaning, identify and handle errors in data, and prepare data for analysis.
Data Investigation	Data investigations are a multistep process. When conducting data investigations, students pose data questions, use computational tools to collect and analyze data, create data visualizations, generate insights, and tell the story of their data. In early grades, students focus on asking simple questions that can be answered with small datasets. As students progress, they work with larger datasets, asking and answering more sophisticated questions that consider variability and relationships between multiple variables.
Impacts of Data Science	Students explore how data influences decision-making and impacts individuals and communities. Students examine the benefits, risks, and ethical considerations around data use. In early grades, students discuss how using data can help them make more informed decisions in their daily lives. As students progress, they explore issues related to bias in data, data privacy, artificial intelligence and machine learning, and large-scale societal and environmental impacts of data science applications.

Systems & Security

Overview: Systems & Security includes the broad categories of hardware and software, networks, and cybersecurity. The physical components (hardware) and instructions (software) that make up a computing system communicate and process information in digital form. Networks connect computing devices to share information and resources. Greater connectivity in the computing world has also led to an increased need for security to protect the information being transmitted.

Subconcept	Overview
Hardware and Software	Computing systems use hardware and software to communicate and process information in digital form. In early grades, students learn how systems use both hardware and software to represent and process information. As they progress, students gain a deeper understanding of the interactions between hardware and software at multiple levels within computing systems.
Security	Transmitting information securely across networks requires appropriate protection. In early grades, students learn how to protect their personal information. As they progress, students learn about information transmission across devices, network design, and how to protect networks from different types of threats.
Networks	Computing devices communicate with one another across networks to share information. In early grades, students learn that computers connect them to other people, places, and things around the world. As they progress, students gain a deeper understanding of how information is sent and received across different types of networks.
Impacts of Computing Systems	Humans created computing systems to accomplish tasks and solve problems. While there have been benefits, there have also been harms and the creation of new problems. In early grades, students examine the impacts of computing systems on individuals. As they progress, students learn about the impacts of computing systems on global society.

Computing & Society

Overview: Computing shapes—and is shaped by—individuals, communities, and cultures from around the world. Computing transforms daily life, economies, governments, and global systems in ways that offer both great promise and significant challenges. The impacts of computing are complex, encompassing advances that improve lives alongside harms that deepen inequities and raise ethical concerns. Students learn to critically and responsibly navigate these social implications, considering issues of equity, access, and accountability, while also exploring computing's potential to promote social good. By examining the evolving relationship among computing, culture, and society, students are empowered to contribute thoughtfully and responsibly to a digital future.

Subconcept	Overview
History of Computing	The history of computing reflects contributions from many societies and knowledge traditions. In early grades, students first identify how computing is used in daily life. Then they focus on how technologies have evolved over time in response to social, scientific, and economic needs. In middle grades, students evaluate how historical challenges led to computing innovations and compare the roles played by individuals, communities, organizations, and governments in advancing these technologies. Students also explore the societal impacts of computing innovations. In later grades, students delve into the main eras of computing history and understand policy and legislation related to computing technologies. They also analyze the historic impacts of technologies, giving consideration to the factors that contributed to disparities across communities.
Emerging Technologies	Computing is a rapidly developing discipline. While other concepts cover what students should know about the current field of CS, this subconcept focuses on recently developed technologies that have the potential to significantly impact society. In early grades, students learn how computing technology aids their daily life. They evaluate choices and consequences related to emerging technologies and identify problems that these advances could address. In middle grades, students evaluate how emerging technologies impact user experiences, while attending to ethical design principles. They explain how emerging technologies can inspire innovation and help people accomplish tasks in new ways. In later grades, students understand the core computational principles behind emerging technologies and compare the ethical considerations of emerging technologies, using various frameworks.
Humans and Computing	Humans and computing technologies are intricately connected. Students will learn that people are the fundamental creators of these technologies and will differentiate between tasks best suited for humans versus those for computing. They will investigate how humans leverage computing to solve problems and examine the motivations behind designing and building these systems. A critical focus is distinguishing between human and computational learning processes, enabling students to decide when and where computing technologies, including AI, are appropriate and helpful. As they progress, students will analyze the trade-offs of using AI-powered solutions and evaluate how human choices in designing, deploying, and regulating AI influence its risks, benefits, and long-term societal impacts.
Career Exploration	Computing is foundational to nearly every career field. Understanding the role of computing in the workplace prepares students to make informed choices about their futures. In early grades, students recognize how digital tools and technologies support everyday work across professions. In middle grades, students explore how computational thinking drives innovation across industries and examine the ethical challenges that professionals may encounter. In later grades, students connect computing to their personal interests and career goals, investigate computing-related pathways, and analyze how advancements in technology foster new opportunities for growth across diverse fields.

Pillars & Practices

Pillars differ from concepts and are collections of practices that are integral to each concept. Practices describe the behaviors and ways of thinking that students with a strong foundation in computer science use to fully engage in a world powered by computing. Each standard is intended to reflect both content and one or more practices.

Ethics & Social Responsibility

The goal of the Ethics & Social Responsibility pillar is for students to develop habits that help them become responsible creators of technology. The practices for Ethics & Social Responsibility are based on the Association for Computing Machinery's list of general ethical principles (ACM, 2018).

Practices

1. **Use computing for positive social impact.**
   - a. Imagine and create computing technologies that solve problems, strengthen communities, and improve quality of life.
   - b. Evaluate whether potential benefits of computing solutions outweigh possible harms. Ensure that harms are not concentrated on specific groups and avoid serious environmental harm
   - c. Explain design trade-offs in computing projects, including what values these decisions reflect and what was prioritized or sacrificed.
2. **Respect others' rights when creating computational technologies.**
   - a. Respect other creators of computational technologies. Only use others' work with permission and give appropriate attribution.
   - b. Respect users' privacy and protect their data. Give users choices about how their information is collected and used. Only collect the minimum amount of information necessary.

Inclusive Collaboration

The core of the Inclusive Collaboration pillar is to help students develop productive collaborations with diverse groups of people. The practices in this pillar, which were synthesized from the STEL (ITEEA, 2020), the Social Justice Standards (Learning for Justice, n.d.), the Framework for 21st Century Learning (Partnership for 21st Century Skills, 2009), and the K–12 CS Framework (2016), address communication skills, project management skills, and personal conduct when working with others.

Practices

3. Communicate effectively about computing.

- **a.** Share technical ideas and explain computing concepts clearly to different audiences.
- **b.** Give and actively listen to others' input and constructive feedback. Consider diverse perspectives and multiple solutions to technical challenges.

4. Manage computing projects.

- **a.** Establish shared goals, break the work into discrete tasks, and set development milestones.
- **b.** Document code and development processes.

5. Act responsibly in computing collaborations.

- **a.** Cultivate working relationships with individuals possessing diverse perspectives, skills, and personalities.
- **b.** Participate reliably in computing teamwork, share responsibility for project outcomes, and meet development deadlines.
- **c.** Reflect on contributions to computing projects, including technical decisions and team interactions, to improve technical and collaboration skills.

Computational Thinking

Computational thinking is a way of thinking about problems and formulating problems and solutions so that an information-processing agent (e.g., a computer) can help to solve them. Computational thinking practices should underpin instruction in each concept and connect students' CS learning experiences. Embedded within this pillar is the engineering design process, in which students identify and define computational problems, develop computational solutions, and iteratively test, refine, and optimize those solutions. These practices are largely based on the original Computational Thinking practices from the K–12 CS Framework (2016), but also incorporate ideas from the Next Generation Science Standards (NGSS Lead States, 2013) and Computational Thinking 2.0 (Tedre et al., 2021).

Practices

6. Define computational problems.

- **a.** Identify real-world problems that can be solved computationally using rule-based approaches, data-driven approaches (e.g., machine learning), or hybrid methods.
- **b.** Clearly state criteria for success and identify constraints.
- **c.** Decompose complex problems into manageable subproblems.

7. Develop and use abstractions.

- **a.** Extract common features and patterns from data, processes, or phenomena to create general methods and algorithms.
- **b.** Create reusable modules and procedures that can apply to multiple situations to reduce complexity.
- **c.** Model phenomena and develop simulations, using rule-based and data-driven approaches, to understand and evaluate potential outcomes.

8. Create computational artifacts.

- **a.** Generate and evaluate multiple solution approaches to determine which best meet the defined criteria given the constraints.
- **b.** Plan the development of computational solutions.
- **c.** Implement solutions by developing or modifying artifacts through traditional programming, model training, or hybrid methods.

9. Test and refine computational artifacts.

- **a.** Test, debug, and troubleshoot computational artifacts systematically using appropriate methods, such as generating test cases for rule-based programs and accuracy evaluation for machine learning models.
- **b.** Iteratively refine and optimize solutions to meet criteria for success.

Human-Centered Design

Using human-centered design practices is a critical piece of responsibly creating computational solutions. Human-centered design includes principles of human-computer interaction. The following practices are drawn from a variety of well-known sources on human-centered design, including the National Institute of Standards and Technology (NIST, 2021), the Interaction Design Foundation (IDF, n.d.), and the UX Design Institute (Vinney, 2023), as well as the K–12 CS Framework (2016).

Practices

10. Understand and involve diverse users in design decisions.

- **a.** Learn about different people's experiences with computing technologies, including those with different abilities, backgrounds, and needs.
- **b.** Gather input and feedback from diverse users throughout the design and development process to help create positive user experiences.

11. Use iterative design processes.

- **a.** Start with simple prototypes and continuously test and refine solutions to ensure usability and accessibility.

12. Design computational technologies that empower and inform users.

- **a.** Respect users' autonomy. Be transparent about how computational technologies make decisions that affect users. Give users control over how they interact with technologies rather than leveraging human limitations to serve creators' interests over users' interests.
- **b.** Consider the benefits and harms of human-like behaviors in computational technologies (e.g., conversational AI) and how they influence user perceptions and actions.

Dispositions

What Are Dispositions?

Dispositions are the **habits of mind, attitudes, and approaches** that shape how students engage with learning. They define how students think, persist, collaborate, and reflect, beyond what they can code or recall. In computer science, dispositions influence how students navigate challenges, debug with purpose, and build confidence in problem-solving. Fostering dispositions ensures students not only learn to program but also develop as self-directed, motivated, and resilient learners.

Priorities in CS Education

In *Reimagining CS Pathways*, the CS education community identified **seven key dispositions** essential to equitable and enduring CS learning:

1. **Creativity:** Generating original, meaningful computing ideas and projects.
2. **Sense of Belonging:** Feeling included, respected, and recognized in the CS community.
3. **Critical Thinking:** Using reasoning and evidence to analyze and refine solutions.
4. **Persistence:** Continuing effort despite frustration or setbacks.
5. **Reflectiveness:** Connecting past experiences to future learning choices.
6. **Resourcefulness:** Strategically seeking tools, people, and references to solve problems.
7. **Curiosity:** Asking questions and exploring beyond assigned work.

The majority of these dispositions align with *self-regulated learning*, where students plan, monitor, and evaluate their growth as learners of CS.

Why These Dispositions Matter

Dispositions turn computing from a set of tasks into a process of *growth and identity formation*. They:

- **Advance equity and inclusion** – Belonging and creativity help all students see themselves reflected in computing.
- **Deepen understanding** – Critical thinking and reflection connect conceptual learning to practice.
- **Build resilience** – Persistence and resourcefulness transform frustration into problem-solving progress.
- **Sustain motivation** – Curiosity keeps students exploring beyond the minimum and learning autonomously.

Dispositions are the foundation of lasting CS learning. They connect technical ability to personal growth, ensuring that every student can engage meaningfully with computing. Dispositions are fostered through intentional instructional design decisions and a consistent instructional approach that encourages their development. When classrooms intentionally emphasize these core dispositions, students develop both the confidence and competence to thrive as problem solvers, innovators, and future leaders in computer science.

Foundational Standards for PK–12

Naming Conventions for Foundational Standards

Each of the identifiers for foundational standards follows this naming convention:

Grade band		Concept		Number
XX	-	YYY	-	##

There are standards for each individual elementary grade. All identifiers for elementary standards begin with "E," followed by a letter (K to indicate PK/Kindergarten standards) or number (1–5) to indicate the grade level. Standards for grades 6–8 are banded together as middle school standards. Identifiers for middle school standards begin with "MS." Standards for grades 9–12 are banded together as high school standards. Identifiers for high school standards begin with "HS."

The next set of characters indicate the concept for each standard. The following table shows the abbreviations for each concept

Abbreviation	Concept
ALG	Algorithms & Design
PRO	Programming
DAA	Data & Analysis
SAS	Systems & Security
CAS	Computing & Society

The last two digits of each standard reflect the standard number. The foundational standards begin with 01 for each concept in each grade or grade band. The standards are numbered continuously across all concepts within each grade or grade band.

To access a progression view of the foundational standards, see:

https://csteachers.org/PK-12-Foundational-Standards-Draft-3.0-Progression-Chart

Algorithms & Design

Algorithm Fundamentals

EK-ALG-01: Carry out algorithms in daily activities.

Boundary Statement(s)	Students should enact simple, step-by-step activities in the correct sequence for familiar classroom or home tasks. Following classroom routines (e.g., cleaning up or lining up) or completing personal tasks (e.g., tying shoes or washing hands) are appropriate. Students are not expected to carry out algorithms that include conditionals or iteration. Students are not required to write algorithms or use a programming language.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Computational Thinking: 7. Develop and use abstractions.
Disposition(s)	Critical Thinking

E1-ALG-01: Decompose a problem or task into individual parts to develop an algorithm.

Boundary Statement(s)	Students should be able to break down familiar, multistep tasks into smaller, ordered steps, focusing on sequencing and logical thinking. For example, when preparing to walk in the hallway, students could identify the steps as: (1) quietly stand up and push in your chair, (2) walk to your spot in line, and (3) face forward with hands to yourself. Students are not expected to work with abstract problems or create algorithms involving conditional logic (if/then). Students are not required to use a programming language.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Computational Thinking: 7. Develop and use abstractions.
Disposition(s)	Critical Thinking, Persistence

E2-ALG-01: Create an algorithm that includes sequence, events, and iteration to solve a problem or express ideas.

Boundary Statement(s)	Students should create simple algorithms using a sequence of steps that respond to events (e.g., "Ready, set, go!" or tapping on a character) and include loops to accomplish a task or express an idea. Appropriate activities include creating a storyboard for a story to be coded later, developing an algorithm to solve a math story problem, or designing an algorithm to conduct a science experiment. Students may complete this work individually or with peers. Students are not expected to use complex or nested loops. Students are not required to create algorithms with conditional branches or variables.
Pillar(s) and Practice(s)	Computational Thinking: 8. Create computational artifacts. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Creativity, Critical Thinking

E3-ALG-01: Create an algorithm that includes sequence, events, iteration, and selection to solve a problem or express ideas.

Boundary Statement(s)	Students should focus on solving problems or expressing ideas that include sequences, responses to events, iteration (e.g., loops), or selection (e.g., if/then). Appropriate activities include writing instructions for a game character to move forward, jumping over obstacles repeatedly, and choosing a different path when encountering a fork in the road. Students may complete this work individually or with peers. Students are not expected to write advanced code or use text-based programming languages. Students are not required to understand complex data structures, optimization, or formal syntax.
Pillar(s) and Practice(s)	Computational Thinking: 8. Create computational artifacts. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking

E4-ALG-01: Write a description of an algorithm using everyday language that incorporates a combination of sequence, events, iteration, and selection to solve a problem or express ideas.

Boundary Statement(s)	Students should write step-by-step representations of an algorithm using grade-appropriate, everyday language that includes sequences, responses to events, iteration (e.g., loops), or selection (e.g., if/then). Outlining a "choose your own adventure" story where readers make decisions that alter the path is appropriate. Students may complete this work individually or with peers. Students are not expected to create visual representations, flowcharts, or formal pseudocode. Students are not required to create a single plan that includes every program control structure.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Creativity, Critical Thinking

E5-ALG-01: Construct a visual representation of an algorithm that incorporates a combination of sequence, events, iteration, selection, and variables to solve a problem or express ideas.

Boundary Statement(s)	Students should create visual representations (e.g., storyboards, physical coding blocks, mind maps, or annotated diagrams) showing algorithms that use combinations of the following control structures: sequence, events, iteration, selection, and variables. Illustrating an algorithm as a drawing on index cards showing the sequence of events from start to finish on a horizontal line with branches for selection is appropriate. Students may complete this work individually or with peers. Students are not expected to include complex data structures (e.g., arrays or lists), functions, or more than one level of nested iteration or selection. Students are not required to use formal flowchart notation or a single comprehensive diagram showing all control structures.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Creativity, Critical Thinking

MS-ALG-01: Develop an algorithm that includes variables, data, and storage.

Boundary Statement(s)	Students should design and explain algorithms that use or update information (e.g., keeping score, tracking inventory, or responding to user inputs). Representations may be verbal, visual, physical, or block-based to illustrate logic rather than executable code. Students may complete this work individually or with peers. Students are not expected to write executable programs, manage memory, or use complex data types.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Persistence, Critical Thinking, Creativity, Resourcefulness

MS-ALG-02: Create a flowchart or pseudocode that includes a combination of sequence, events, iteration, selection, and variables to model an algorithm.

Boundary Statement(s)	Students should represent algorithms using flowcharts or clearly structured pseudocode that captures core programming concepts. Teacher-provided templates or agreed-upon formats are appropriate. Students are not expected to use professional programming syntax or convert pseudocode directly into executable code.
Pillar(s) and Practice(s)	Computational Thinking: 7. Develop and use abstractions. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Reflectiveness, Curiosity, Sense of Belonging in CS, Persistence

HS-ALG-01: Develop an algorithm that includes at least one procedure that has sequence, iteration, and selection.

Boundary Statement(s)	Students should design algorithms that use named, reusable procedures incorporating sequence, selection, and iteration. For example, a student could create a procedure to calculate a student's final grade by iterating through a list of scores and applying a rule to drop the lowest grade. Students may complete this work individually or with peers. Students may use real-world data but it is not required. Students are not expected to develop algorithms for complex systems or use advanced data structures such as heaps or trees.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Resourcefulness, Creativity

Problem Solving

Standards do not begin until middle school.

MS-ALG-03: Verify the correctness of an algorithm for given inputs.

Boundary Statement(s)	Students should test whether a given algorithm produces expected outputs for specific inputs by tracing steps or running the algorithm. For example, students could check whether a sorting algorithm correctly orders numbers or whether a set of instructions builds a shape as intended. Students are not expected to perform formal proofs of correctness, analyze efficiency, or optimize algorithms. Students are not required to use mathematical notation or comparison of multiple algorithms for performance.
Pillar(s) and Practice(s)	Computational Thinking: 9. Test and refine computational artifacts. Human-Centered Design: 11. Use iterative design processes.
Disposition(s)	Critical Thinking, Persistence

MS-ALG-04: Decide whether to use rule-based, data-driven, or hybrid approaches when solving problems.

Boundary Statement(s)	Students should distinguish among rule-based (explicit steps), data-driven (patterns learned from data), and hybrid (combined) approaches, and reason about which method would be most appropriate for different kinds of problems. For example, students might discuss why a rule-based approach fits a simple game like tic-tac-toe, why data-driven methods are better for weather prediction, or why a hybrid approach could work for personalized recommendations. Students should focus on evaluating and justifying choices, not on building or implementing the systems themselves. Students are not expected to design or implement AI/machine learning models or explain their underlying statistical or computational mechanisms.
Pillar(s) and Practice(s)	Inclusive Collaboration: 4. Manage computing projects. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Curiosity

MS-ALG-05: Generate outputs from AI models to assist in solving a computational problem.

Boundary Statement(s)	Students should use age-appropriate AI tools or prebuilt models to generate outputs that support solving a computational problem. Examples include generating text to brainstorm ideas, using image recognition to classify objects, or asking a chatbot for troubleshooting help. Students should focus on interpreting and assessing AI outputs, not building or programming AI systems. Students are not expected to train AI models or explain underlying algorithms.
Pillar(s) and Practice(s)	Computational Thinking: 8. Create computational artifacts. Human-Centered Design: 11. Use iterative design processes.
Disposition(s)	Reflectiveness, Curiosity

HS-ALG-02: Evaluate algorithms for efficiency, correctness, and clarity, using metrics or test cases.

Boundary Statement(s)	Students should analyze algorithms for efficiency (time or memory use), correctness (accuracy of outputs), and clarity (human readability), using simple metrics or test cases. For example, they could compare two algorithms for finding the largest number in a list by testing runtime when using different sized lists. Students are not expected to apply formal mathematical proofs or Big O notation. Students are not required to analyze all possible inputs.
Pillar(s) and Practice(s)	Computational Thinking: 9. Test and refine computational artifacts.
Disposition(s)	Critical Thinking, Reflectiveness

HS-ALG-03: Optimize the design of an algorithmic solution using abstractions such as procedures, modules, lists, and/or objects.

Boundary Statement(s)	Students should improve algorithmic designs by identifying repetitive patterns or complex logic and restructuring using abstractions for clarity and efficiency. For example, they might refactor repeated code to calculate the area of a rectangle into a reusable procedure that performs the calculation. Students should focus on incremental improvement of existing designs. Students are not expected to identify the single "best" solution to a problem.
Pillar(s) and Practice(s)	Computational Thinking: 7. Develop and use abstractions. Computational Thinking: 9. Test and refine computational artifacts.
Disposition(s)	Reflectiveness, Critical Thinking, Resourcefulness

HS-ALG-04: Evaluate AI-generated output to assess bias, accuracy, and potential harms.

Boundary Statement(s)	Students should analyze and critique AI-generated content (e.g., text, images, or code) for accuracy, bias, and potential harms. For example, they might examine how an image-generation tool underrepresents certain groups or produces inaccurate results due to biased training data. Students should focus on critical evaluation of outputs rather than technical implementation. Students are not expected to build or debug AI models or understand their internal mathematical algorithms.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact.
Disposition(s)	Critical Thinking, Reflectiveness, Sense of Belonging in CS

Machine Learning

EK-ALG-02: Recognize patterns that people and machines can use to make decisions.

Boundary Statement(s)	Students should notice and identify simple, repeating patterns in objects, sounds, movements, or pictures, and recognize that these patterns can be used by people and machines to sort, predict, or classify. For example, students might identify a sequence of colored blocks (red, blue, red, blue) and reason that a "smart" toy could use this pattern to guess the next color. Students are not expected to define machine learning or distinguish among types of pattern recognition. Students are not required to generate complex or abstract patterns beyond familiar, observable examples.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 2. Respect others' rights when creating computational technologies.
Disposition(s)	Sense of Belonging in CS, Critical Thinking

E1-ALG-02: Investigate how patterns can be used by people and machines to make predictions and classify objects into categories.

Boundary Statement(s)	Students should investigate patterns in familiar objects or pictures and use those patterns to make simple predictions and to classify items into clear, teacher- or student-defined categories (e.g., color, shape, size, or use). For example, students might recognize several rules that function to classify the majority of items. When classifying, students should ask "Who or what doesn't fit our rule?" and notice when a rule mistakenly excludes or mislabels items. Students are not expected to use the term bias or to use probability or statistics beyond counting and comparing. Students are not required to build decision trees.
Pillar(s) and Practice(s)	Computational Thinking: 7. Develop and use abstractions.
Disposition(s)	Critical Thinking, Reflectiveness

E2-ALG-02: Examine how computing technologies can learn from patterns in data.

Boundary Statement(s)	Students should explore simple examples of how a trained machine learning model identifies patterns in data and makes predictions or recommendations based on those patterns. Students should notice that models can make more mistakes for some kinds of examples than others, especially when the training examples were mostly of one kind. Students should focus on observing that AI systems use patterns in data to make predictions or choices and that limited or one-sided examples can lead to unfair or uneven results. For example, they might label images for a teacher-trained model and observe how well it classifies new pictures, or use a drawing app that predicts what object they are sketching before they finish. Students are not expected to independently train AI or machine learning models.
Pillar(s) and Practice(s)	Computational Thinking: 7. Develop and use abstractions. Human-Centered Design: 12. Design computational technologies that empower and inform users.
Disposition(s)	Reflectiveness, Curiosity

E3-ALG-02: Investigate how AI models can evolve when new data is added to a training set.

Boundary Statement(s)	Students should observe and describe what happens when a trained AI or machine learning model receives additional training data. They should notice when adding missing or underrepresented examples helps the model make more accurate predictions. For example, students could use an image-recognition tool to identify animals, then add more pictures to see how the model's predictions change or improve. Students should focus on observing how new data affects outcomes. Students are not expected to train their own AI models or understand the technical or mathematical processes behind them. The term bias may be introduced informally, but students are not required to use it. Students are not required to analyze complex datasets or create models from scratch.
Pillar(s) and Practice(s)	Computational Thinking: 9. Test and refine computational artifacts.
Disposition(s)	Critical Thinking, Curiosity

E4-ALG-02: Train an AI model to make a classification or prediction.

Boundary Statement(s)	Students should upload a labeled dataset (e.g., text, numbers, images, sounds, or poses) to train an AI model that classifies, predicts, or recommends. Students should reflect on results that do not match expectations (e.g., who is missing from the training data, how adding more diverse examples might make results more accurate). Students should focus on recognizing that training data influences model accuracy. For example, students might create a labeled dataset of animal sounds to train an AI tool, test its accuracy on new examples, and reflect on outputs that differ from their expectations. Students are not expected to understand the mathematical processes that enable an AI model to make predictions. Students are not required to use specific AI tools or specialized hardware.
Pillar(s) and Practice(s)	Computational Thinking: 8. Create computational artifacts. Computational Thinking: 9. Test and refine computational artifacts.
Disposition(s)	Critical Thinking, Reflectiveness

E5-ALG-02: Analyze relationships between the properties of training data and an AI model's output.

Boundary Statement(s)	Students should examine how characteristics of training data (e.g., amount, accuracy, labeling quality, and representativeness) affect an AI model's performance. Students should connect unbalanced or poorly labeled data to inaccurate outputs and may learn the term bias to describe this pattern. Students should focus on making qualitative observations (e.g., "more data improves accuracy" or "labeling mistakes lead to incorrect predictions"). For example, students might observe that when too few images are used to train a classifier, the model often mislabels new images, leading them to conclude that data quantity and quality influence accuracy. Students are not expected to use statistical formulas, understand how AI decision-making works mathematically, or define or correct specific bias types.
Pillar(s) and Practice(s)	Computational Thinking: 9. Test and refine computational artifacts. Human-Centered Design: 11. Use iterative design processes.
Disposition(s)	Reflectiveness, Critical Thinking

MS-ALG-06: Make informed predictions about the hidden processes and functions of AI and other complex systems.

Boundary Statement(s)	Students should make reasoned predictions about how an AI system or other complex technology might be processing information based on observed inputs and outputs. Students should focus on forming credible, evidence-based inferences from experimentation and observation. For example, they could compare results from multiple prompts and describe what the AI seems to prioritize or omit. Students are not expected to explain exactly how AI systems work internally.
Pillar(s) and Practice(s)	Computational Thinking: 7. Develop and use abstractions. Computational Thinking: 9. Test and refine computational artifacts.
Disposition(s)	Critical Thinking, Persistence, Reflectiveness

MS-ALG-07: Investigate ways to improve the accuracy of an AI model and reduce bias by refining the quality of examples and non-examples in its training data.

Boundary Statement(s)	Students should test an AI model to evaluate its outputs, identify overrepresented or missing examples in its training data, and suggest improvements to increase accuracy or reduce bias. Students should focus on testing, interpreting results, and suggesting refinements based on evidence. For example, students might notice that a model misidentifies certain objects and propose adding more diverse examples, then retest to see if performance improves. Students are not expected to create AI models or understand the technical mechanics of training.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Computational Thinking: 9. Test and refine computational artifacts.
Disposition(s)	Critical Thinking, Reflectiveness, Resourcefulness

MS-ALG-08: Create a model card to describe the features and limitations of an AI model.

Boundary Statement(s)	Students should create a concise model card that documents an AI model's inputs, outputs, and limitations. The model card should include: (a) a description of the datasets used for training, including potential sources of bias; (b) the contexts or scenarios in which the model should or should not be used; (c) the computing power or time required for operation; and (d) potential risks, ethical concerns, privacy considerations, and recommendations for fair and responsible use. Students should focus on documentation and analysis, not implementation or debugging. Students are not expected to build an AI model or access proprietary training data.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Critical Thinking, Reflectiveness

HS-ALG-05: Justify the selection of an AI algorithm to accomplish a task.

Boundary Statement(s)	Students should justify the selection of an AI algorithm for a specific task by evaluating its strengths and limitations relative to the problem requirements. This justification may consider data size and type, human interpretability, and desired outcomes. Students should focus on evaluating appropriateness, not technical construction. For example, students might explain that a decision tree is suitable for classifying spam emails because its logic is transparent and easy to explain, even if less accurate than other models. Students are not expected to understand or implement the underlying mathematics or statistics of these algorithms.
Pillar(s) and Practice(s)	Computational Thinking: 7. Develop and use abstractions.
Disposition(s)	Critical Thinking, Reflectiveness

HS-ALG-06: Evaluate training data by examining its source, quality, representativeness, potential biases, and privacy implications before using it to solve a problem.

Boundary Statement(s)	Students should critically evaluate datasets before use by examining their origin, purpose, accuracy, completeness, representativeness, and privacy implications. For example, when using crime statistics, students might check where the data originated (e.g., police reports, surveys), assess its quality, ensure all communities are represented, and identify potential biases or privacy concerns. Students are not expected to conduct advanced statistical analyses to prove representativeness or to apply formal bias correction techniques.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Ethics and Social Responsibility: 2. Respect others' rights when creating computational technologies.
Disposition(s)	Reflectiveness, Critical Thinking

HS-ALG-07: Develop an AI model for a chosen task using appropriate data and tools.

Boundary Statement(s)	Students should develop an AI model for a defined task by selecting suitable tools and data, training the model, and testing its performance against task requirements. Students should focus on applying accessible AI tools to create and evaluate functional models, not on mastering the mathematics that underlie them. For example, a student could use a block-based AI platform or a simple Python library to create a model that recognizes handwritten digits or classifies objects. Students are not expected to write algorithms from scratch or use advanced programming languages.
Pillar(s) and Practice(s)	Computational Thinking: 8. Create computational artifacts. Computational Thinking: 9. Test and refine computational artifacts.
Disposition(s)	Creativity, Critical Thinking, Persistence

Impacts of Algorithms and Design

EK-ALG-03: Describe how people make algorithms.

Boundary Statement(s)	Students should recognize that computers function according to algorithms created by people. They should describe, in simple terms, the human process of creating an algorithm (i.e., identifying a task, breaking it into smaller parts, and ordering steps to reach a goal). For example, students might describe how people design instructions for making a sandwich or sorting classroom supplies. Students are not expected to describe all aspects of computational thinking. Students are not required to understand how algorithms are translated into computer code.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Reflectiveness

E1-ALG-03: Illustrate how changes to algorithms lead to different outcomes for people.

Boundary Statement(s)	Students should illustrate, through everyday activities or drawings, that changing the order of steps can produce different results. For example, when following two sets of instructions to build a block tower, one set might create a tall, stable tower while another creates a shorter or weaker one. Students are not expected to write or modify computer code. Students are not required to understand complex algorithms used in digital tools.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Reflectiveness, Curiosity

Algorithms & Design

Algorithm Fundamentals

Problem Solving

Machine Learning

Impacts of Algorithms and Design

E2-ALG-03: Describe how algorithms might impact peers in varied situations.

Boundary Statement(s)	Students should describe how algorithms can affect people in fair or unfair ways. The goal is to describe how an existing algorithm might affect others. For instance, lining up alphabetically may always place the same student last, influencing how that student feels. Students are not expected to analyze technical details of algorithms or understand bias. Students are not required to change or fix algorithms.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Reflectiveness

E3-ALG-03: Compare how different algorithms for solving the same problem produce outcomes that may benefit or disadvantage different groups of people.

Boundary Statement(s)	Students should compare two or more algorithms that solve the same problem and explain how each affects people differently. Examples include different ways to sort items, plan routes, or schedule activities, noting who benefits or is inconvenienced. Students are not expected to design new algorithms or evaluate effectiveness. Students are not required to analyze algorithmic bias.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Human-Centered Design: 12. Design computational technologies that empower and inform users.
Disposition(s)	Critical Thinking, Reflectiveness

Algorithms & Design
Algorithm Fundamentals
Problem Solving
Machine Learning
Impacts of Algorithms and Design

E4-ALG-03: Evaluate how different algorithms may affect outcomes, situations, and people with a wide range of needs.

Boundary Statement(s)	Students should evaluate examples where algorithms produce different outcomes and discuss how those differences may help or harm people with diverse needs or perspectives. Students should focus on comparing impacts, not on programming or implementation. For example, students might consider how a navigation app choosing the "fastest" route differs from one choosing the "safest" route. Students are not expected to analyze algorithms at a technical level or evaluate efficiency or performance trade-offs.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Reflectiveness

E5-ALG-03: Articulate how human-centered design principles can be incorporated into the development of computational solutions, including AI and other emerging technologies.

Boundary Statement(s)	Students should identify familiar examples of computing solutions (e.g., websites, voice assistants, video games, recommendation systems) and explain how human-centered design considerations (e.g., empathy, user needs, requirements, accessibility, fairness, sustainability, accountability) can be incorporated into their development. Students should consider the impact of their values (e.g., personal, community, and larger values) on the design. They should also recognize that AI and other computing technologies are created by people, which may sometimes lead to unfair or inaccurate outcomes. Students are not expected to understand the technical details of AI or other computing technologies.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Reflectiveness

Algorithms & Design
Algorithm Fundamentals
Problem Solving
Machine Learning
Impacts of Algorithms and Design

MS-ALG-09: Plan the design of a computational solution, considering human-centered design principles.

Boundary Statement(s)	Students should apply principles of empathy, accessibility, and user needs when planning computational solutions. They should identify the needs of diverse users (e.g., people with disabilities, multilingual users, those with limited internet access), propose design strategies to address them (e.g., larger buttons, alt-text for images, alternative color schemes), and discuss trade-offs between different design choices and their repercussions (e.g., fairness in automated decision-making algorithms). Students should also identify laws that set requirements for accessibility (e.g., ADA, IDEA, Section 508). Students are not expected to produce fully functional solutions or address every identified need. Students are not required to develop an in-depth understanding of accessibility laws or complex user-testing processes.
Pillar(s) and Practice(s)	Inclusive Collaboration: 4. Manage computing projects. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Creativity, Sense of Belonging in CS, Critical Thinking, Curiosity

MS-ALG-10: Describe evidence of beneficial and harmful impacts, ethical issues, and biases of algorithms encountered in daily life.

Boundary Statement(s)	Students should identify and describe common ethical issues and biases in AI systems using familiar examples (e.g., photo apps misidentifying people, voice assistants misunderstanding accents, or recommendation systems reinforcing prior interests). They should evaluate whether the outcomes of these systems seem fair and inclusive and explain why biases may occur in simple, concrete terms. Students are not expected to program or modify AI tools, understand neural networks, or study global AI policy.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Creativity, Sense of Belonging in CS, Critical Thinking, Reflectiveness

MS-ALG-11: Modify an algorithm to address a specific societal impact, ethical issue, or bias.

Boundary Statement(s)	Students should propose modifications to an existing algorithm, verbally or through diagrams or flowcharts, to make its outcomes more fair or inclusive. Students should focus on reducing bias and improving equity through simple, conceptual modifications. For example, they might suggest adding a language option to a chatbot or adjusting a music playlist generator to include more diverse artists. Students should explain how their modification reduces bias or improves equity. Students are not expected to design complex algorithms from scratch or apply statistical fairness techniques.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Human-Centered Design: 12. Design computational technologies that empower and inform users.
Disposition(s)	Creativity, Critical Thinking, Reflectiveness, Curiosity

HS-ALG-08: Develop computational solutions using human centered design principles.

Boundary Statement(s)	Students should design computational solutions that are inclusive and accessible by following a human-centered design process. This includes researching user needs, exploring design options, and anticipating the social and ethical impacts of their design. For example, when creating a website, students might ensure sufficient color contrast, keyboard navigation, and clear language for users with differing abilities. Students are not expected to conduct large-scale user studies or meet every criterion of formal accessibility standards (e.g., WCAG).
Pillar(s) and Practice(s)	Computational Thinking: 8. Create computational artifacts. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Reflectiveness, Resourcefulness, Critical Thinking

HS-ALG-09: Evaluate the ethical implications, societal impacts, and potential biases of rule-based and data-driven algorithms.

Boundary Statement(s)	Students should evaluate how rule-based and data-driven algorithms can produce biased or inequitable outcomes. They should identify potential sources of bias (e.g., unrepresentative training data or biased design rules) and discuss the social impacts of these systems. For example, students might analyze a hiring algorithm that filters candidates by zip code or a facial recognition model that performs less accurately for certain skin tones. Students' analysis should remain conceptual, focusing on observable impacts and ethical reasoning. Students are not expected to audit proprietary systems or use advanced statistical methods.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Computational Thinking: 9. Test and refine computational artifacts.
Disposition(s)	Critical Thinking, Reflectiveness, Persistence

HS-ALG-10: Articulate the values embedded in the design of algorithmic systems.

Boundary Statement(s)	Students should identify and explain the values reflected in the design of algorithmic systems, recognizing that every design choice involves prioritizing certain outcomes or perspectives. Students should focus on identifying underlying priorities (e.g., efficiency, privacy, fairness, transparency) that shape algorithmic behavior. For example, students might examine a social media platform's recommendation system and discuss how its design favors engagement and attention over accuracy or civility. Students are not expected to deconstruct proprietary systems or analyze full business models.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Persistence, Reflectiveness

Programming

Programming Fundamentals

EK-PRO-04: Create a sequence of commands to complete a simple task or express ideas.

Boundary Statement(s)	Students should arrange a small set of directions in order (e.g., to help a character reach an object, tell a short story, or follow a classroom routine). Using manipulatives, arrows, or simple digital programs is appropriate. Unplugged activities are appropriate. Students are not expected to write complex code or use control structures beyond sequencing. Students are not required to use computers.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Creativity, Curiosity

E1-PRO-04: Create code from an algorithm that includes sequence and events to complete a task or express ideas.

Boundary Statement(s)	Students should read algorithms with sequential steps and an initiating event (e.g., "when start is clicked") and translate them into block-based code. Tasks such as moving a character or navigating a robot through a maze are appropriate. Students are not expected to design the algorithm themselves or use selection or iteration.
Pillar(s) and Practice(s)	Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Creativity, Persistence

E2-PRO-04: Create code from an algorithm that includes sequence, events, and iteration to complete a task or express ideas.

Boundary Statement(s)	Students should translate a provided algorithm that begins with an event and includes sequential steps and loops into a working program. Writing an animation in a block-based language from a storyboard or coding a robot to dance from an algorithm of arrows are appropriate. Students are not expected to work with selection, variables, or nested loops, or to author the original algorithm.
Pillar(s) and Practice(s)	Computational Thinking: 7. Develop and use abstractions. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Critical Thinking, Persistence

E3-PRO-04: Develop code from a student-created algorithm that includes sequence, events, iteration, and selection to complete a task or express ideas.

Boundary Statement(s)	Students should implement their own or a classmate's algorithm in a block-based environment using events, loops for efficiency, and basic decisions. Students should also give credit to the algorithm author and implementer. For example, turning a storyboard for a game that includes a game character moving forward, jumping over obstacles repeatedly, and choosing different paths when encountering a fork in the road into a program using a block-based programming language is appropriate. Students are not expected to use text-based languages or design multi-level projects. Students are not required to implement a single program that includes every control structure.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 2. Respect others' rights when creating computational technologies. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Persistence, Resourcefulness

E4-PRO-04: Compare different programming solutions to the same problem based on their correctness and clarity.

Boundary Statement(s)	Students should compare two or more short programs or code segments that solve the same task, noting which produces the correct result and which are easier to understand or follow. Examples that use short block-based or text-based code segments are appropriate. Students are not expected to create multiple versions themselves, evaluate efficiency, or conduct formal code reviews.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Computational Thinking: 9. Test and refine computational artifacts.
Disposition(s)	Critical Thinking, Curiosity

E5-PRO-04: Create a novel program by modifying or combining elements of existing programs.

Boundary Statement(s)	Students should make purposeful changes to a program by incorporating teacher-created, peer-created, or other code segments to create a unique variation of the original program. Remixing or adapting existing code to add new characters, actions, or outcomes is appropriate. Students are not expected to write entirely original, complex programs from scratch or manage large-scale integrations of multiple projects. Students are neither required nor restricted from incorporating AI-generated code.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 2. Respect others' rights when creating computational technologies. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Resourcefulness, Critical Thinking

Standards end after Grade 5.

Program Development

Standards do not begin until Grade 1.

E1-PRO-05: Discuss how a program might affect different users.

Boundary Statement(s)	Students should discuss ways programs may help or hinder different people. Using familiar programs (e.g., read-aloud apps, drawing tools, simple games) and concrete examples is appropriate. For example, students might express how a read-aloud app helps kids who are learning to read or how a game with small buttons could be hard for someone with limited hand mobility. Students are not expected to analyze unfamiliar, complicated, or adult-oriented software, explain how programs are built, or address security or accessibility root causes.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Inclusive Collaboration: 3. Communicate effectively about computing.
Disposition(s)	Sense of Belonging in CS, Curiosity

E2-PRO-05: Collaborate with a partner to develop a program that solves a problem or expresses an idea.

Boundary Statement(s)	Students should work with a partner to iteratively plan, create, and test a grade-appropriate block-based program. Collaboratively planning, coding, and testing a program that includes events, sequencing, and simple iteration is appropriate. Students are not expected to build complex or large-scale programs or use programming constructs like selection and variables.
Pillar(s) and Practice(s)	Inclusive Collaboration: 5. Act responsibly in computing collaborations. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Creativity, Persistence

E3-PRO-05: Use structured, constructive feedback to improve programs.

Boundary Statement(s)	Students should give and apply specific, helpful suggestions to improve the function or design of their programs. Incorporating feedback from peers, teachers, or users into a program is appropriate. Students are not expected to interpret complex error reports, use advanced tools, or overhaul program structure or design. Students are not required to defend against all critiques or implement all feedback.
Pillar(s) and Practice(s)	Inclusive Collaboration: 5. Act responsibly in computing collaborations. Computational Thinking: 9. Test and refine computational artifacts.
Disposition(s)	Critical Thinking, Persistence

E4-PRO-05: Collaborate with a team by offering a meaningful contribution to creating a program.

Boundary Statement(s)	Students should work in small groups to plan and create a simple program, contributing ideas, code segments, or testing feedback while practicing respectful communication, active listening, and shared tasking. Adding a feature, fixing an error, improving a design element, or helping organize the code are appropriate. Students are not expected to use professional collaboration tools, manage large projects, or apply formal methodologies.
Pillar(s) and Practice(s)	Inclusive Collaboration: 4. Manage computing projects. Inclusive Collaboration: 5. Act responsibly in computing collaborations.
Disposition(s)	Sense of Belonging in CS, Persistence

E5-PRO-05: Construct individual components of a program that are collaboratively assembled into a working project.

Boundary Statement(s)	Students should individually build small code components and integrate them with teammates into a complete functional program. For example, in a block-based environment, one student might construct the code component for a scoring system using variables and selection, while another constructs the player movement component, and they then integrate and test these pieces to ensure the final project works. Students are not expected to use version control or design full system architecture.
Pillar(s) and Practice(s)	Inclusive Collaboration: 5. Act responsibly in computing collaborations. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Resourcefulness, Persistence

MS-PRO-12: Use procedures to structure code for clarity and reusability.

Boundary Statement(s)	Students should identify repetition in code, define a clearly named procedure to perform the repeated task, call it where needed, and explain benefits of using the procedure (e.g., less repetition, easier updates, clearer code). Students are not expected to write procedures with parameters or return values. Students are not expected to manage many interacting procedures.
Pillar(s) and Practice(s)	Inclusive Collaboration: 4. Manage computing projects. Computational Thinking: 7. Develop and use abstractions.
Disposition(s)	Creativity, Critical Thinking, Resourcefulness

MS-PRO-13: Use reference documentation and online resources to write, debug, and improve programs.

Boundary Statement(s)	Students should use search engines with specific keywords to find reliable programming information that helps them write, debug, and refine their programs. They should use teacher-provided or self-selected documentation to understand functions or blocks, and search for common error messages to identify causes and solutions. Students are not expected to conduct open-ended research on complex programming problems, read professional-level documentation, or apply coding concepts beyond their grade level.
Pillar(s) and Practice(s)	Inclusive Collaboration: 4. Manage computing projects. Computational Thinking: 9. Test and refine computational artifacts.
Disposition(s)	Critical Thinking, Resourcefulness, Curiosity

MS-PRO-14: Explain the importance of attribution and intellectual property in programming.

Boundary Statement(s)	Students should define intellectual property in programming and explain why giving credit to original creators matters. They should recognize that programming makes use of multiple artifacts (e.g., code, images, music, and text), and understand the difference between learning from others' code and plagiarizing it. Identifying whether code is open source, free to use, or proprietary, and providing basic attribution in their own work, are appropriate. Students should focus on ethical awareness and responsible use of others' work. Students are not expected to study copyright law, fair use, or licensing in depth, or to license their own code.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 2. Respect others' rights when creating computational technologies. Inclusive Collaboration: 3. Communicate effectively about computing.
Disposition(s)	Critical Thinking

MS-PRO-15: Apply inclusive collaboration practices to develop a program.

Boundary Statement(s)	Students should actively share ideas during planning, provide and receive constructive feedback in all stages of program development, and collaborate effectively on shared digital tools. Students should focus on productive teamwork and inclusive communication. Practicing equal participation, clear communication, and respect for others' contributions is appropriate. Students are not expected to use formal collaboration methods like Agile or Scrum, or professional project management software.
Pillar(s) and Practice(s)	Inclusive Collaboration: 4. Manage computing projects. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Sense of Belonging in CS

HS-PRO-11: Create a modular program that uses procedures, modules, or objects to improve reusability and readability.

Boundary Statement(s)	Students should decompose a complex problem into smaller, manageable parts and implement each part as a distinct procedure, module, or object. Students should focus on creating clear, reusable, and well-organized code. For example, a game program might separate player movement, scoring, and enemy behavior into separate functions or objects. Students are not expected to use advanced software design patterns or deep inheritance structures.
Pillar(s) and Practice(s)	Computational Thinking: 7. Develop and use abstractions. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Critical Thinking

HS-PRO-12: Use documentation, libraries, Application Programming Interfaces (APIs), and development tools to write, debug, and improve programs.

Boundary Statement(s)	Students should locate and use external resources to support their programming, including built-in documentation, common libraries, simple APIs, and features of an Integrated Development Environment (IDE) (e.g., debuggers or syntax highlighting). Students should focus on effectively using available resources to enhance programs. For example, using a math library instead of writing a trigonometric function from scratch is appropriate. Students are not expected to create APIs or advanced development tools, or to debug complex systems.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 2. Respect others' rights when creating computational technologies. Computational Thinking: 9. Test and refine computational artifacts.
Disposition(s)	Resourcefulness

HS-PRO-13: Apply proper attribution and respect intellectual property in digital artifacts.

Boundary Statement(s)	Students should apply intellectual property principles by giving accurate credit for any code or digital media created by others. They should demonstrate ethical practice by documenting sources according to their terms of use or license. For example, a student creating a game might include a "Credits" section listing each asset's creator, source, and license. Students are not expected to master copyright law.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 2. Respect others' rights when creating computational technologies. Inclusive Collaboration: 3. Communicate effectively about computing.
Disposition(s)	Reflectiveness

HS-PRO-14: Collaborate on a programming project using a defined workflow that includes design documentation, version control, and clear task roles.

Boundary Statement(s)	Students should use collaborative workflows to organize group projects (e.g., assigning roles, maintaining shared design documents, and using basic version control tools to manage changes). Students should focus on learning structured teamwork practices with accessible, classroom-ready tools. For example, a group of students might use a project management tool to assign tasks, a collaborative document (e.g., Google Docs) for design specifications, and a version control system (e.g., Git) to manage code changes. Students are not expected to use professional-grade tools or manage large-scale projects.
Pillar(s) and Practice(s)	Inclusive Collaboration: 4. Manage computing projects. Inclusive Collaboration: 5. Act responsibly in computing collaborations.
Disposition(s)	Sense of Belonging in CS, Persistence

HS-PRO-15: Translate an algorithm written in pseudocode into a working program that includes sequence, iteration, selection, variables, procedures, parameters, and data structures.

Boundary Statement(s)	Students should convert a provided pseudocode algorithm into a functioning program that uses core programming concepts (variables, sequence, selection, iteration, data structures, procedures, parameters). Students should focus on understanding how to implement algorithms accurately in code. For example, translating pseudocode for calculating the average of a list into executable code is appropriate. Students are not expected to create the pseudocode themselves or use a specific programming language.
Pillar(s) and Practice(s)	Computational Thinking: 7. Develop and use abstractions. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Critical Thinking, Persistence

Reading and Documenting Code

EK-PRO-05: Describe how code has completed a task.

Boundary Statement(s)	Students should observe and describe, in their own words, what happens when simple code runs (e.g., "the robot moved three steps," "the character turned red"). Gestures or short phrases are appropriate. Students are not expected to read text-based code, explain why it works, use technical vocabulary, or explain advanced programming concepts.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing.
Disposition(s)	Critical Thinking

E1-PRO-06: Explain the function of code that includes sequence and events.

Boundary Statement(s)	Students should explain what each step in a short program does and how an event triggers what happens next. Expressing what happens when a sequence of movement blocks runs or when a tap or collision event starts an animation are appropriate. Students are not expected to analyze advanced control structures (e.g., loops, conditionals), formal flow, syntax, or efficiency.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Computational Thinking: 6. Define computational problems.
Disposition(s)	Reflectiveness, Curiosity

E2-PRO-06: Explain the steps taken during program development, recognizing the contributions of others in the process.

Boundary Statement(s)	Students should describe, in simple sequence, what they did first, next, and last while creating a program, and acknowledge classmates' and teachers' help. Teacher-provided graphic organizers or sentence starters are appropriate. Students are not expected to use advanced vocabulary, formal documentation, or complex debugging narratives.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 2. Respect others' rights when creating computational technologies. Inclusive Collaboration: 3. Communicate effectively about computing.
Disposition(s)	Sense of Belonging in CS, Reflectiveness

E3-PRO-06: Articulate how a specific segment of code contributes to the overall purpose of a program.

Boundary Statement(s)	Students should use age-appropriate academic language to describe what a code segment does and how it supports the program's goal. For example, students sharing their understanding via annotated screenshots, prompted explanations, or brief presentations is appropriate. Students are not expected to conduct formal code reviews or be the original author of the code.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Reflectiveness

E4-PRO-06: Document a program to clarify its functionality.

Boundary Statement(s)	Students should use simple documentation (e.g., brief comments, labeled diagrams, short descriptions) to explain what their program does and how it works. The goal is to make the program easier for others to understand or modify. Students are not expected to produce professional design documents or complex explanations of code structure, use version control, or write industry-style technical prose.
Pillar(s) and Practice(s)	Inclusive Collaboration: 4. Manage computing projects.
Disposition(s)	Critical Thinking, Resourcefulness

E5-PRO-06: Create embedded or external documentation for a programming project.

Boundary Statement(s)	Students should produce documentation common to their programming language environment (e.g., in-code comments, project notes) describing purpose, use, and code organization. Comments, virtual sticky notes, program descriptions or an external document are all appropriate options. Students are not expected to follow professional templates or produce formal documentation sets. While teachers may opt to introduce guidelines or templates, students are not required to follow any specific guidelines for documentation.
Pillar(s) and Practice(s)	Inclusive Collaboration: 4. Manage computing projects. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Resourcefulness

MS-PRO-16: Analyze how a segment of code works by identifying the roles of iteration, selection, variables, and procedures.

Boundary Statement(s)	Students should demonstrate how a segment of code works. Tracing the execution of a code segment and the value of variables through sequences of code (e.g., step-by-step in order), iteration (e.g., repeated sections of code), selection, and procedures (e.g., separate modules of code) are appropriate. Students are not expected to debug or test the code, or analyze procedures with parameters.
Pillar(s) and Practice(s)	Inclusive Collaboration: 4. Manage computing projects. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking

MS-PRO-17: Examine AI-generated code for accuracy and usability in a programming project.

Boundary Statement(s)	Students should design and run simple test cases on AI-generated code, use varied inputs to check behavior, and report how well the code meets the stated purpose and if the code includes any bias. Students should focus on comparing results to the intended goal. Students are not expected to generate code with AI themselves or understand every implementation detail.
Pillar(s) and Practice(s)	Computational Thinking: 9. Test and refine computational artifacts. Human-Centered Design: 11. Use iterative design processes.
Disposition(s)	Curiosity, Reflectiveness, Critical Thinking

HS-PRO-16: Analyze how a segment of code works, including the role of sequence, iteration, selection, variables, procedures, parameters, and data structures.

Boundary Statement(s)	Students should read existing code and explain how parts interact to achieve the goal (e.g., variables hold state, loops repeat, conditionals branch, procedures encapsulate logic). For example, students could analyze a segment of code for a simple guessing game and explain how a variable stores the secret number, how a while loop continues until the correct number is guessed, and how an if/else statement provides feedback on whether the guess was too high or too low. Students are not expected to analyze complex algorithms or systems with many interconnected data structures.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Computational Thinking: 7. Develop and use abstractions.
Disposition(s)	Critical Thinking, Persistence

HS-PRO-17: Evaluate AI-generated code for accuracy, reliability, and alignment with program requirements.

Boundary Statement(s)	Students should evaluate AI-generated code by testing it against predetermined requirements and verifying that it produces correct and reliable results. They should analyze whether the code meets its intended purpose, handles a variety of inputs appropriately, and follows any given design constraints (e.g., using a specific data structure or algorithm). Students should focus on evaluating the quality and reliability of the output, not the internal mechanisms of the AI system. For example, a student using an AI tool to generate a function that calculates an average should test edge cases (e.g., empty lists, single values, mixed positive and negative numbers) and confirm that outputs are accurate and consistent with the task description. Students are not expected to understand the underlying machine learning models or algorithms that power the AI tools.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Computational Thinking: 9. Test and refine computational artifacts.
Disposition(s)	Critical Thinking, Reflectiveness

Testing and Refining Code

EK-PRO-06: Identify steps in a sequence of commands that do not work as expected.

Boundary Statement(s)	Students should recognize which step in a simple sequence is not working correctly or is causing a problem. Observing and identifying a block that led to an unexpected result in a block-based coding app or a command card that was incorrect in an unplugged activity are appropriate. Students are not expected to explain why the error occurred, analyze program logic, or use formal debugging strategies.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Computational Thinking: 9. Test and refine computational artifacts.
Disposition(s)	Critical Thinking, Persistence

E1-PRO-07: Debug programs that include sequence and events.

Boundary Statement(s)	Students should identify and fix simple errors in block-based programs they have made or were given. Students should focus on identifying and correcting simple, visible mistakes. For example, if a character is supposed to move and then jump, but the code blocks are in the wrong order, the student should be able to identify the problem and reorder the blocks. Students are not expected to find complex logical errors or debug programs that include loops or conditionals.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Computational Thinking: 9. Test and refine computational artifacts.
Disposition(s)	Critical Thinking, Persistence

E2-PRO-07: Debug programs that include sequence, events, and iteration.

Boundary Statement(s)	Students should test a simple, block-based program, find a bug, and fix it. Students should focus on locating and correcting one obvious bug in a short program that includes iteration. For example, if a character draws only three sides of a square instead of four, students should identify the issue and adjust the repeat block. Students are not expected to fix multiple or complex errors or explain problems using technical terms.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Computational Thinking: 9. Test and refine computational artifacts.
Disposition(s)	Critical Thinking, Persistence

E3-PRO-07: Debug iteration errors and selection errors in a program.

Boundary Statement(s)	Students should identify and correct errors in programs that use loops and in programs that use conditional logic. Debugging one level of iteration and selection is appropriate. Students are not expected to debug nested iterations or selections.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Computational Thinking: 9. Test and refine computational artifacts.
Disposition(s)	Critical Thinking, Persistence

E4-PRO-07: Debug programs that include sequence, events, iteration, and selection.

Boundary Statement(s)	Students should locate and fix common logical or structural errors in programs using sequence, events, loops, and conditional choices. For example, debugging a block-based program where a character does not move when clicked, a loop runs too many times, or an if block triggers the wrong action is appropriate. Students are not expected to debug syntax errors in text-based languages or correct complex programs involving variables, functions, or nested conditions or loops.
Pillar(s) and Practice(s)	Computational Thinking: 9. Test and refine computational artifacts.
Disposition(s)	Critical Thinking, Persistence

E5-PRO-07: Debug programs using systematic strategies.

Boundary Statement(s)	Students should apply more than one systematic debugging strategy. Strategies such as reproducing the problem, reading error messages, tracing code, changing one element at a time, or adding temporary outputs (e.g., sounds or print statements) to locate issues are appropriate. Students should focus on using clear, repeatable methods to find and fix syntax, logic, or runtime errors. Students are not expected to use professional debugging tools or automated testing frameworks.
Pillar(s) and Practice(s)	Computational Thinking: 9. Test and refine computational artifacts.
Disposition(s)	Persistence, Reflectiveness

MS-PRO-18: Use standard practices to test, debug, document, and peer-review code.

Boundary Statement(s)	Students should apply systematic testing and debugging techniques to find and fix errors, use clear comments to document their code, and give and receive constructive feedback during peer code reviews. Students are not expected to use advanced debugging software, unit testing frameworks, version control systems, or follow documentation standards beyond basic commenting practices.
Pillar(s) and Practice(s)	Inclusive Collaboration: 4. Manage computing projects. Computational Thinking: 9. Test and refine computational artifacts.
Disposition(s)	Persistence, Critical Thinking, Resourcefulness, Reflectiveness, Sense of Belonging in CS

MS-PRO-19: Refine a program based on user feedback to improve its usability and accessibility.

Boundary Statement(s)	Students should discuss with peers user feedback, consider the needs of diverse users, and identify which suggestions would most improve a program's usability or accessibility. Students should focus on developing awareness of others' needs and incremental improvement. Students are not expected to perform formal audits or implement complex accessibility frameworks.
Pillar(s) and Practice(s)	Computational Thinking: 9. Test and refine computational artifacts. Human-Centered Design: 11. Use iterative design processes.
Disposition(s)	Critical Thinking, Resourcefulness

HS-PRO-18: Evaluate a program's alignment with design specifications and responsible design values, including its correctness, effectiveness, and user experience.

Boundary Statement(s)	Students should be able to systematically evaluate a program they have created or a program created by others by applying a logical process to ensure it works as intended and meets the original design plan. A concrete example of this in a high school class would be a student creating a hangman game. They would evaluate its correctness by testing every possible input, including incorrect letters, multiple correct letters, and invalid characters, to ensure the program never crashes. In addition, they would include responsible design (ethical, social, environment, or human-centric values).They would evaluate its user experience by checking if the game responds quickly and provides clear feedback to the user and that it meets responsible design values. Finally, they would evaluate its alignment with design specifications by comparing the finished program to their initial plan, making sure all features, such as a word bank and a counter for incorrect guesses, are present and functional. Students are not expected to use formal verification methods. Students are not required to master industry-specific tools.
Pillar(s) and Practice(s)	Computational Thinking: 9. Test and refine computational artifacts. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Reflectiveness, Critical Thinking

HS-PRO-19: Refine a program based on user feedback and testing results, applying responsible design values to improve functionality, usability, accessibility, accuracy, and efficiency.

Boundary Statement(s)	Students should use testing results and user feedback to make meaningful refinements to a program's functionality, usability, and accessibility. For example, a student might add a help command, simplify instructions, or adjust text size and contrast to improve readability and accessibility. Students are not expected to correct every functional or user-related issue or achieve full optimization.
Pillar(s) and Practice(s)	Human-Centered Design: 10. Understand and involve diverse users in design decisions. Human-Centered Design: 11. Use iterative design processes.
Disposition(s)	Critical Thinking, Reflectiveness

Data Handling

EK-PRO-07: Identify everyday gestures and symbols that represent information people use to make choices.

Boundary Statement(s)	Students should recognize and explain common symbols, signs, and gestures they see in familiar environments such as home, school, or playgrounds. Students should be able to state what each symbol means or what action it signals. Symbols with clear, immediate meanings are appropriate (e.g., stop signs, traffic lights, a teacher's raised hand for quiet, thumbs up/down, or pictures on classroom labels). Students are not expected to interpret abstract or unfamiliar symbols, read written words on signs, or analyze why specific symbols were chosen. Students are not required to understand cultural variations, complex decision-making, or digital icons requiring prior background knowledge.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Computational Thinking: 7. Develop and use abstractions.
Disposition(s)	Critical Thinking

E1-PRO-08: Identify terms that refer to values that change over time in everyday life.

Boundary Statement(s)	Students should name real-world examples of information that changes, such as temperature, time, score, or age. For example, students can identify that the score of a game changes, the temperature outside changes, or their age changes each year. Students should recognize that some information changes while other information stays the same. Students are not expected to use the formal term variable or understand the programming concept of assignment.
Pillar(s) and Practice(s)	Computational Thinking: 7. Develop and use abstractions.
Disposition(s)	Critical Thinking, Curiosity

E2-PRO-08: Label different representations of information with a name and whether its value is constant or changes.

Boundary Statement(s)	Students should give logical names to pieces of information they encounter in real life and determine whether the value of each piece of information changes or stays constant. Students should focus on describing information in meaningful, age-appropriate terms. Examples from digital contexts (e.g., score in a game) and non-digital contexts (e.g., temperature in an oven, day of the week) are appropriate. Students are not expected to use programming vocabulary or write code.
Pillar(s) and Practice(s)	Inclusive Collaboration: 4. Manage computing projects. Computational Thinking: 7. Develop and use abstractions.
Disposition(s)	Critical Thinking, Reflectiveness

E3-PRO-08: Identify the variables being stored and manipulated in a program.

Boundary Statement(s)	Students should read completed programs to find variables. Students should recognize that each variable has a unique name that must be spelled consistently. Identifying variables that track a game score, user input, or collected data is appropriate. Students are not expected to assign variables specific data types (e.g., string, integer, Boolean) or write code themselves.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Computational Thinking: 7. Develop and use abstractions.
Disposition(s)	Critical Thinking, Persistence

E4-PRO-08: Trace how data flows through a program and changes variable values during execution.

Boundary Statement(s)	Students should follow short, age-appropriate programs to observe how data moves through the code and how variable values change as the program runs. Students should focus on tracing and explaining observable changes to variable values. For example, they might trace how a score counter increases with each correct answer or how a temperature tracker updates as new readings are added. Students are not expected to design variable-driven programs, work with multiple data types, or understand memory allocation, scope, or data structures.
Pillar(s) and Practice(s)	Computational Thinking: 7. Develop and use abstractions. Computational Thinking: 9. Test and refine computational artifacts.
Disposition(s)	Critical Thinking, Persistence

E5-PRO-08: Use variables to store, compare, and modify data within a program.

Boundary Statement(s)	Students should use variables in block-based or beginner text-based environments to store, compare, and update data that affects program behavior. Examples include tracking a player's score, comparing time values to trigger an event, or updating health or points after an action. Students are not expected to manage complex data types, arrays, or mathematical formulas involving multiple variables. Students are not required to understand variable scope.
Pillar(s) and Practice(s)	Computational Thinking: 7. Develop and use abstractions. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Critical Thinking

MS-PRO-20: Use appropriate data types and structures to store, modify, update, and iterate over data within a program.

Boundary Statement(s)	Students should select and use common data types (e.g., numbers, strings, Boolean values) and basic data structures (e.g., lists, arrays, simple tables) to store, organize, and process information. They should be able to modify and update data elements and iterate over collections (e.g., looping through a list of scores to find an average). Students are not expected to implement advanced data structures (e.g., trees, graphs, linked lists, or multi-dimensional arrays) or use professional programming syntax.
Pillar(s) and Practice(s)	Computational Thinking: 8. Create computational artifacts. Human-Centered Design: 11. Use iterative design processes.
Disposition(s)	Critical Thinking, Persistence, Resourcefulness, Curiosity

HS-PRO-20: Create a program that uses appropriate data structures to store, access, and manipulate data.

Boundary Statement(s)	Students should design and implement programs that use data structures such as arrays, lists, or dictionaries to organize, access, and modify data effectively. Students should focus on selecting and applying suitable structures to manage data efficiently. For example, students might create a program that stores and retrieves information about daily tasks, chores, or student names and grades. Students are not expected to implement advanced data structures like linked lists, stacks, queues, trees, or graphs.
Pillar(s) and Practice(s)	Computational Thinking: 7. Develop and use abstractions. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Critical Thinking, Persistence

HS-PRO-21: Compare fundamental data types and their uses.

Boundary Statement(s)	Students should analyze the characteristics and appropriate uses of fundamental data types (i.e., integers, floats, strings, Booleans) and compare and contrast how choosing different data types affects program behavior and accuracy. Students should focus on reasoning about data type selection for different programming needs. For example, students might determine that a login system might be created using a string for the username, an integer for the user's age, and a Boolean to check if the user is logged in. Students are not expected to understand how data types are stored in memory or perform low-level data optimization.
Pillar(s) and Practice(s)	Computational Thinking: 7. Develop and use abstractions.
Disposition(s)	Critical Thinking, Persistence

Data & Analysis

Data Fundamentals

EK-DAA-08: Demonstrate how people create and collect data to help answer questions.

Boundary Statement(s)	Students should collect data to answer a question about a grade-appropriate topic involving two to three variables. Having students wonder about the world to create questions to answer is appropriate. Collecting simple survey data and creating graphs as a class in order to answer a question is appropriate. Students are not expected to work independently. Students are not required to use digital tools or collect large sets of data.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 2. Respect others' rights when creating computational technologies. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Persistence, Curiosity

E1-DAA-09: Use multiple methods, including observation, measurement, and survey, to collect both numeric and non-numeric data.

Boundary Statement(s)	Students should use simple observation, counting, and surveys to collect data. Examples include counting objects, measuring lengths with a ruler, or asking classmates about favorite colors. Students are not expected to analyze or organize data into complex charts. Students are not required to use digital tools or formal statistical methods.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Curiosity

E2-DAA-09: Compare numeric and non-numeric types of data in terms of how they are collected and what they can tell us.

Boundary Statement(s)	Students should compare how numeric data can be used for math and measurement, while non-numeric data can be used to sort, group, and describe. Having students identify the kinds of pets the class has (non-numeric data), count the total number of pets in each category (numeric data), and make simple comparisons (e.g., more students have cats vs dogs) is appropriate. Students are not expected to perform more sophisticated data transformations or analyses, nor are students expected to identify finer-grained data types (e.g., integers and characters). Students are not required to use technology to collect or compare different types of data.
Pillar(s) and Practice(s)	Computational Thinking: 7. Develop and use abstractions.
Disposition(s)	Reflectiveness

E3-DAA-09: Evaluate numeric and non-numeric data for accuracy and completeness.

Boundary Statement(s)	Students should focus on checking for missing data and recognizing data that seems unreasonable (e.g., values that are much too large or much too small or data that is the wrong type). Students may use a survey to collect data which is recorded on a chart. Students may also use data supplied by the teacher. Students are not expected to create complex graphs or work with large datasets. Students are not required to use technology for collection, recording, or evaluation of data.
Pillar(s) and Practice(s)	Computational Thinking: 7. Develop and use abstractions. Ethics and Social Responsibility: 2. Respect others' rights when creating computational technologies.
Disposition(s)	Critical Thinking, Curiosity

E4-DAA-09: Organize collected data into tables using digital tools, with rows representing records and columns representing attributes.

Boundary Statement(s)	Students should practice organizing small sets of collected or provided data into simple tables, ensuring that each row represents one record (e.g., a survey respondent, a pet) and each column represents an attribute (e.g., age, favorite color, type of animal). Students may use paper-based tables and templates prior to using digital tools for this work. Students are not expected to design complex databases, manage large datasets, or use advanced spreadsheet functions. Students do not need to understand relational databases, formulas, or data visualization techniques.
Pillar(s) and Practice(s)	Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Critical Thinking

E5-DAA-09: Use digital tools to collect and organize different types of data.

Boundary Statement(s)	Students should demonstrate their ability to use digital tools, such as simple surveys and spreadsheets, to collect and organize data. Students should collect and distinguish between quantitative (numeric) and qualitative (descriptive and non-numeric) data. When organizing the data, students should use a computational tool to efficiently sort and group similar data (e.g., sort from highest to lowest temperature, or sort non-numeric responses alphabetically). Students are not expected to create their own data collection tools from scratch, nor are they expected to analyze, interpret, or visualize the data.
Pillar(s) and Practice(s)	Computational Thinking: 9. Test and refine computational artifacts.
Disposition(s)	Critical Thinking, Persistence

MS-DAA-21: Evaluate how different levels of precision and granularity in data collection affect accuracy, storage, and analysis.

Boundary Statement(s)	Students should explain the difference between precision and granularity in data collection and evaluate how these choices affect data accuracy, storage requirements, and interpretation. Students should identify and explain situations in which higher precision or granularity might be more important. For example, students might compare tracking daily step counts (lower granularity, less storage) versus tracking steps every minute (higher granularity, more storage), or recording ages in years (lower precision) versus years and months (higher precision). Comparing medical monitoring data (requiring high precision and granularity for patient safety) with casual fitness tracking data (where lower precision and granularity are sufficient) is appropriate. Students are not expected to collect data at varying precision or granularity levels themselves. Students are not expected to understand the technical details of data collection processes.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Inclusive Collaboration: 4. Manage computing projects.
Disposition(s)	Critical Thinking

MS-DAA-22: Explain how data and its associated metadata can be used to answer questions.

Boundary Statement(s)	Students should be able to explain how data and metadata serve different but complementary purposes when answering questions. Students should understand that metadata is data about data—it provides context that helps interpret the primary data content. For example, a photo's image data shows what was captured, while its metadata (e.g., timestamp, location, camera settings) provides context about when, where, and how the photo was taken. Students should be able to use computational tools to view metadata (e.g., viewing file properties, examining web page source code). Students should recognize that metadata can be collected automatically without their explicit knowledge (e.g., their phone adding location and time data to photos). Students are not expected to understand technical standards for metadata formats or structures. Students are not required to focus on any particular metadata tool or to manually create or edit metadata fields. Students are not expected to provide exhaustive lists of all possible metadata types for different file formats.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 2. Respect others' rights when creating computational technologies. Inclusive Collaboration: 3. Communicate effectively about computing.
Disposition(s)	Critical Thinking, Reflectiveness

HS-DAA-22: Use a digital tool to generate data that fits certain parameters for use in simulations.

Boundary Statement(s)	Students should be able to generate synthetic data using a digital tool. Students should be able to write a simple program or use a spreadsheet function to create a dataset that meets specific criteria (e.g., generate a set number of data points within a specific range of values or to create a certain distribution). For example, students could generate random initial velocities and launch angles within specified ranges to serve as inputs for a projectile motion simulation. Students should focus on creating data that can be used by a pre-existing simulation or model. Students are not expected to build the simulation from scratch. Students are also not expected to use advanced statistical methods or write complex algorithms to generate the data.
Pillar(s) and Practice(s)	Computational Thinking: 8. Create computational artifacts. Human-Centered Design: 12. Design computational technologies that empower and inform users.
Disposition(s)	Critical Thinking, Persistence, Resourcefulness

HS-DAA-23: Create a data dictionary that describes the names and types of attributes, allowable values/ranges for each attribute, and logical relationships between variables in a dataset.

Boundary Statement(s)	Students should be able to create a data dictionary for a dataset, including the names of all attributes (variables), their data types, and the range of acceptable values for each. The data dictionary should also describe the logical connections between variables, such as how a particular response on a survey question might lead to missing responses on other questions due to survey skip logic. For example, students could analyze a dataset of anonymized survey responses from a school, and then create a data dictionary that lists each question as an attribute, defines the response format (e.g., integer, text string), and specifies the valid range of responses (e.g., a rating scale might allow the numbers 1-5 and "No Response" if the question was skipped). Students are not expected to create the dataset itself or handle extremely large, complex datasets that require specialized tools or databases.
Pillar(s) and Practice(s)	Computational Thinking: 7. Develop and use abstractions. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Critical Thinking, Persistence

Data Processing

Standards do not begin until middle school.

MS-DAA-23: Use a digital tool to sort, filter, group, aggregate, and otherwise transform quantitative and qualitative data.

Boundary Statement(s)	Students should be able to access and use a spreadsheet program or other age-appropriate digital tool to apply data operations (e.g., sort, filter, group, aggregate) to a given set of data. Students can work with a pre-existing data set of a grade-level-appropriate size. Students should be able to explain why they are using each operation to transform their data and how it will help answer their data question (e.g., "I am filtering the data to show only results from 2023 so I can analyze recent trends," or "I am grouping by product type and calculating total sales to find our best-selling category.") Students do not need to collect data to fulfill this standard. Students also do not need to perform advanced data organization or cleaning. Students do not need to master any computational programs or software to transform data.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Critical Thinking

MS-DAA-24: Analyze options to address data quality issues.

Boundary Statement(s)	Students should be able to identify common data quality issues, such as missing data and incorrect formatting as well as investigate their causes. Students should also be able to develop multiple approaches for addressing data quality issues, such as deleting records with a lot of missing attributes, reformatting data that doesn't conform to expected values, or leaving the data as is. Students should evaluate the outcomes of each approach to addressing data quality and consider how removing data can introduce bias into a data set. Students can be provided with a data set to analyze. Students are not expected to choose a "correct" approach for addressing data quality. Students should be focused on analyzing and evaluating solutions. Students are not expected to use specialized software programs or understand advanced statistical corrections. Students are not expected to work with large, complex datasets nor do they need to generate their own data with errors to analyze. Students are not expected to use automation tools to identify errors. Students should be focused on the human oversight of errors in data to prepare them to understand how decisions made by humans in the development of automation tools can impact others.
Pillar(s) and Practice(s)	Computational Thinking: 9. Test and refine computational artifacts. Human-Centered Design: 11. Use iterative design processes.
Disposition(s)	Critical Thinking

HS-DAA-24: Use a digital tool to clean and organize text-based data.

Boundary Statement(s)	Students should be able to use a digital tool to prepare text-based data for analysis. This includes handling inconsistencies (e.g., variations in capitalization or spacing), correcting errors (e.g., misspellings or invalid entries), and structuring the data appropriately (e.g., separating combined fields or standardizing formats). Students should recognize when regular expressions can be used for pattern matching in text. Using a spreadsheet program or a simple programming script to convert inconsistent date formats, remove duplicate entries, or separate a single text column into multiple columns (e.g., first name and last name) within a dataset of survey responses is appropriate. Students are not expected to build cleaning tools from scratch. Students are not required to handle large-scale, unstructured data like social media feeds.
Pillar(s) and Practice(s)	Computational Thinking: 9. Test and refine computational artifacts. Human-Centered Design: 11. Use iterative design processes.
Disposition(s)	Persistence, Critical Thinking

HS-DAA-25: Evaluate different approaches to verifying consistency and compliance with expected data types, values, and ranges.

Boundary Statement(s)	Students should be able to identify and evaluate different methods for ensuring that data is clean, accurate, and ready for use. This includes examining data for expected formats, values, and ranges. For example, students could evaluate different approaches for validating a dataset of customer birthdates to ensure all entries are in the correct format (e.g., MM/DD/YYYY) and fall within a reasonable range of years. Students are not expected to write or implement scripts or use statistical models to perform data validation.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Computational Thinking: 7. Develop and use abstractions.
Disposition(s)	Critical Thinking, Persistence

Data Investigation

EK-DAA-09: Investigate a question that can be answered by collecting data in students' everyday environments.

Boundary Statement(s)	Students should pose a simple, concrete question that can be answered by gathering data in familiar contexts (e.g., "How many students ride the bus?"; "What is the most common favorite fruit in our class?"). Students are not required to use computational tools. Students are not expected to design experiments or other types of research studies, use statistics, or analyze large datasets.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems.
Disposition(s)	Curiosity

E1-DAA-10: Compare questions that can be answered with data investigations and questions that are answered through other means.

Boundary Statement(s)	Students should focus on questions that are developmentally relevant (e.g., pets, recess activities, favorite lunch items) and sort examples of questions into "data" or "non-data" categories (e.g., "How many students have pets?" vs. "What is your favorite animal?"). Creating oral explanations, drawings, or simple charts is appropriate. Connecting data/non-data questions to community norms (e.g., some data may be personal and not appropriate to collect) is appropriate. Students are not expected to work with large data sets or use computational tools. Students are not required to come up with complicated questions or do math that is beyond first grade expectations.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Curiosity

E2-DAA-10: Develop a question that can be answered with data.

Boundary Statement(s)	Students should brainstorm and refine a question that can be answered with data. Students must understand that the best way to answer some questions is not through data collection. For example, a student might start with a vague question like "What colors do kids like?" and refine it to "What is the most common favorite color in our class?" Students can develop questions related to counting and frequency, comparisons, interpretation of tally charts, bar graphs, and pictographs, and making simple predictions and inferences. Students are not required to use computational tools for data collection or creation of data representations. Students are not required to use statistical concepts beyond grade level (e.g., mean, median, mode, probability). Students are not required to create or interpret graphs with complex scales or work with large datasets.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Computational Thinking: 7. Develop and use abstractions.
Disposition(s)	Critical Thinking, Curiosity

E3-DAA-10: Investigate a data question involving relationships between multiple attributes.

Boundary Statement(s)	Students should investigate how two or more attributes in a dataset relate to each other. Investigating the relationship between sunlight and water on the height of a plant, where students collect data on all three attributes and examine how height changes with different combinations of sunlight and water, is appropriate. Students are not expected to perform statistical analysis. Students are not expected to work with more than two or three attributes. Students are not required to use digital equipment to collect or analyze data.
Pillar(s) and Practice(s)	Computational Thinking: 7. Develop and use abstractions.
Disposition(s)	Critical Thinking, Curiosity

E4-DAA-10: Write a brief narrative that includes at least one data visualization to report the process and results of a data investigation, using computing tools.

Boundary Statement(s)	Students should create a short narrative and data visualization through the use of accessible digital tools (e.g., spreadsheets, slides, age-appropriate data visualization applications). A narrative that explains how they collected data and what the data shows, using both text and a bar chart or other basic chart or graph, is appropriate. Students are not expected to produce statistical analyses, interactive dashboards, or complex data visualizations. Students are not required to have knowledge of data visualization design principles beyond choosing a graph type that is appropriate for the data being presented.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Creativity, Reflectiveness

E5-DAA-10: Analyze a dataset to identify the nature and possible sources of variability in the data.

Boundary Statement(s)	Students should identify patterns and variations in a dataset and distinguish between typical values and outliers. Students should look for trends or regularities in data and identify relationships between different variables. Understanding that data can vary from one observation to another and distinguishing between typical values and outliers is appropriate. Students are not required to calculate statistical measures (e.g., mean, median, standard deviation) or perform statistical tests.
Pillar(s) and Practice(s)	Computational Thinking: 7. Develop and use abstractions.
Disposition(s)	Critical Thinking, Reflectiveness

MS-DAA-25: Use computational tools to identify relationships among variables in a dataset and make classifications or predictions.

Boundary Statement(s)	Students should use spreadsheets or programming tools to identify relationships between different variables in a large dataset. Once a relationship is identified it can be used to make a prediction. Students should critically analyze their predictions to see if they make sense for unknown conditions or input. Students should recognize that many artificial intelligence systems work by identifying relationships and making predictions. Students are not required to understand statistics beyond what is covered in sixth grade math standards.
Pillar(s) and Practice(s)	Inclusive Collaboration: 4. Manage computing projects. Computational Thinking: 7. Develop and use abstractions.
Disposition(s)	Reflectiveness, Critical Thinking

MS-DAA-26: Create data visualizations to show how different design choices can impact the interpretation of the same data.

Boundary Statement(s)	Students should generate multiple visualizations for the same data to highlight a key insight they discovered. Students should critically analyze whether the different visualizations are equally clear and accurate, visually appealing, and accessible to all people, including those with visual or auditory disabilities. Students should also critically analyze the visualizations to see if they convey the same message despite different design choices. Students can use representations other than visual to communicate insights from data. Students can work with pre-existing data. Students are not required to use computational tools, though they may be helpful.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 2. Respect others' rights when creating computational technologies. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Creativity, Curiosity

MS-DAA-27: Summarize a data investigation process by describing the question, the data collection and analysis methods, potential biases and limitations, and the evidence supporting the conclusion.

Boundary Statement(s)	Students should describe a data investigation process they completed themselves or with a team, or interview someone else and report on their process. Students should describe the motivation for the data investigation, all decisions made in the process, whether the results are satisfactory or as expected, and any limitations of the investigation. Students are not expected to collect actual data but rather reflect on a process that is already completed. Students are not required to use computational tools but they may find them helpful.
Pillar(s) and Practice(s)	Inclusive Collaboration: 5. Act responsibly in computing collaborations. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Reflectiveness

HS-DAA-26: Use computational tools to create data visualizations of multivariate data sets to answer a question, classify, or make predictions.

Boundary Statement(s)	Students should be able to use computational tools (e.g., Excel, CODAP) to create data visualizations (e.g., scatter plots, line graphs, stacked bar charts, heat maps) that represent the results of their data investigations. For example, to represent the school's recycling habits, students could create a stacked bar chart showing the distribution of recycling material type (paper, plastic, metal) by school location (cafeteria, gym, classrooms) or time of day. Students are not expected to use text-based programming or implement an algorithm or model to generate their visualization or make predictions. Students are also not required to conduct statistical analyses.
Pillar(s) and Practice(s)	Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Critical Thinking, Resourcefulness

Data & Analysis
Data Fundamentals
Data Processing
Data Investigation
Impacts of Data Science

HS-DAA-27: Evaluate the results of data simulations and data visualizations to help answer data questions and to inform decision-making, including identifying limitations.

Boundary Statement(s)	Students should be able to analyze the results presented in data visualizations and data simulations to determine their validity and usefulness in answering a question. Students should be able to identify potential biases, misleading representations, or limitations in the data or the model that could lead to different conclusions. For example, students could analyze a line graph showing daily air quality index values and identify whether the y-axis starts at zero or at a higher value, which could exaggerate changes in air quality. Students should focus on evaluating existing data products rather than creating them. Students are not expected to create their own complex data simulations. Students are not required to rewrite or debug the code that generated the data visualizations or simulations.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact.
Disposition(s)	Critical Thinking, Reflectiveness

Impacts of Data Science

EK-DAA-10: Investigate how data can help a person make informed decisions in everyday life.

Boundary Statement(s)	Students should explore everyday situations involving things they can count, observe, sort, and touch. Examples include household objects, classroom materials, and weather attributes. Students should compare quantities (e.g., less than, more than, or equal to) and use patterns to make simple decisions based on data. Students are not expected to work with large datasets. Students are not required to use statistical concepts. Students should not make decisions that are subjective and not based on data.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact.
Disposition(s)	Critical Thinking, Reflectiveness

E1-DAA-11: Examine a variety of data questions that address the needs of a person or community.

Boundary Statement(s)	Students should examine data questions that are relevant to their lives (e.g., "How many students bring lunch versus buy lunch?", "What are the most common ways students get to school?"). Students should consider how a given data question might be relevant to or impact different people and communities and how minor changes to wording can affect how we interpret and answer the question (e.g., "How many first-graders read at home?" vs. "How many first-graders have books at home?"). Students are not expected to generate data questions or answer the questions being examined. Students are not required to collect or analyze data.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Resourcefulness

Data & Analysis
Data Fundamentals
Data Processing
Data Investigation
Impacts of Data Science

E2-DAA-11: Distinguish between data collection approaches, including those that may lead to inaccurate or biased data.

Boundary Statement(s)	Students should articulate how different ways of gathering data could affect different people or communities they are familiar with (e.g., classroom, family, school, neighborhood). Students should identify which communities might be unfairly represented or left out by specific data collection methods. Students should compare similar approaches to describe how they might produce different results. Students should connect these differences to real-world situations where biased data collection could lead to unfair decisions for their community. Students are not expected to analyze data at this stage. Students are not required to collect data.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Curiosity

E3-DAA-11: Design a data collection process that addresses the needs of people from different backgrounds or groups.

Boundary Statement(s)	Students should consider different data collection plans and how they would affect accuracy and representation. Students' plans should include what data to gather, what tools to use, and what sources to access for an investigation. Tools like class surveys or tally charts marking observations are appropriate. Students should identify different groups and describe how they would be affected by different data collection strategies. Students can obtain consent by including an opt out box for the survey a student is designing. Students should learn that they have a right to tell authority figures (e.g., teachers, website, etc.) they don't want to share their information. Students are not required to use advanced data collection techniques or statistics. Students are not required to consider the ethics associated with data collection or design a process for more than two or three variables.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Human-Centered Design: 12. Design computational technologies that empower and inform users.
Disposition(s)	Critical Thinking, Resourcefulness

Data & Analysis
Data Fundamentals
Data Processing
Data Investigation
Impacts of Data Science

E4-DAA-11: Investigate how data collected about people may affect individuals and groups.

Boundary Statement(s)	Students should investigate simple scenarios to identify who is affected by data collection and what the effect is, including concerns about privacy and fairness. Students should ask clarifying questions (e.g., "Who is collecting this information?", "What information are they collecting?", and "Why do they need it?"). Students should identify who is affected, distinguishing between individuals and groups (e.g., the whole class). Students should describe what the effect is for the individuals and groups using simple, descriptive terms (e.g., "helpful", "harmful", or "unfair"). Gathering facts and identifying clear effects rather than weighing pros and cons is appropriate. Students are not expected to conduct formal research projects or make evaluative judgments about whether the data collection is overall good or bad. Students are not expected to understand the technical aspects of how data is collected online.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Ethics and Social Responsibility: 2. Respect others' rights when creating computational technologies.
Disposition(s)	Critical Thinking, Reflectiveness

E5-DAA-11: Analyze the benefits and risks of computing technology that uses collected data.

Boundary Statement(s)	Students should examine everyday technologies that rely on collected data (e.g., weather apps, fitness trackers, or AI tools like voice assistants or recommendation systems) and identify both benefits (e.g., convenience, personalization, or improved predictions) and risks (e.g., privacy concerns, inaccurate results, or bias). Students are not expected to understand technical details of algorithms, machine learning model training, or data storage infrastructure. Students are not required to evaluate statistical methods, encryption, or policy frameworks.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact.
Disposition(s)	Critical Thinking, Reflectiveness

Data & Analysis
Data Fundamentals
Data Processing
Data Investigation
Impacts of Data Science

MS-DAA-28: Explain the benefits and risks of allowing personal data and metadata to be collected and incorporated into datasets, including data ownership, privacy, and sovereignty.

Boundary Statement(s)	Students should be able to explain both benefits and risks of allowing personal data and metadata to be collected by websites, apps, and other computing technologies. Students should understand that benefits might include personalized recommendations (e.g., suggested videos or products), improved services based on user feedback data, or free access to platforms in exchange for data collection. Students should recognize risks including loss of privacy through location tracking or browsing history collection, unauthorized sale or sharing of personal data, and loss of control over how their information is used (data ownership). Students should understand basic concepts of data sovereignty, such as how data collected in one country might be stored or governed under another country's laws. Students are not expected to understand technical implementations of data protection (e.g., encryption methods, secure protocols) or to analyze specific legal frameworks (e.g., GDPR, COPPA). Students are not required to create technical explanations of cybersecurity systems or to evaluate the technical adequacy of privacy policies.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Inclusive Collaboration: 5. Act responsibly in computing collaborations.
Disposition(s)	Critical Thinking

Data & Analysis
Data Fundamentals
Data Processing
Data Investigation
Impacts of Data Science

MS-DAA-29: Analyze how decisions made during data collection, data processing, data analysis, and data presentation can lead to biased data, misleading conclusions, and compromised AI models.

Boundary Statement(s)	Students should be able to recognize that every stage of working with data involves human choices, and those choices can affect the results, often unintentionally. For example, if a survey only includes certain groups of people, the data might be biased even if the researchers did not intend to exclude anyone. If graphs are created with misleading scales, the conclusions can be distorted. If an AI model is trained with incomplete or unfair data, its predictions may also be biased. Students should examine these issues in simple examples and explain the limitations of the data and how those limitations affect interpretations. Students are not required to learn advanced statistical methods, the technical details of AI training algorithms, or programming-level solutions to bias. Students are not required to have deep knowledge of machine learning mathematics or data science processes.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Inclusive Collaboration: 4. Manage computing projects.
Disposition(s)	Critical Thinking

Data & Analysis
Data Fundamentals
Data Processing
Data Investigation
Impacts of Data Science

HS-DAA-28: Evaluate the societal, environmental, and ethical implications of large-scale data collection and processing, including AI applications.

Boundary Statement(s)	Students should be able to analyze and articulate the societal, environmental, and ethical trade-offs associated with large-scale data collection and processing. This includes evaluating issues such as individual privacy and consent, data sovereignty, algorithmic bias, the spread of misinformation, and the environmental impact of data centers. For example, students could evaluate a social media platform's data collection policy and discuss the ethical implications of using that data for targeted advertising. Students are not expected to perform a formal legal or economic analysis of data policies or otherwise focus on the technical or legal specifics.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 2. Respect others' rights when creating computational technologies. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Reflectiveness, Sense of Belonging in CS

HS-DAA-29: Debate the efficacy of policies and regulations to ensure responsible data use.

Boundary Statement(s)	Students should be able to analyze and evaluate the effectiveness of existing data policies and regulations. This could look like students researching a specific data privacy issue and preparing arguments for a classroom debate on whether a new law is needed to protect consumer data. Students should include the efficacy of social approaches (social movements, political resistance, personal behavioral change) aimed at shaping the design and use of computing systems. Students are not expected to read the text of laws.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Reflectiveness, Persistence

Systems & Security

Hardware and Software

EK-SAS-11: Examine the use of tools to accomplish tasks or solve problems for different users.

Boundary Statement(s)	Students should explore and talk about how everyday tools, both digital and non-digital, help people complete activities. Students should also recognize that different tools are designed to solve specific problems for different people. Age-appropriate examples include: using a pencil or pen to write, using a paintbrush or drawing app to create a picture, using a magnifying glass to see small things, or using an app on a tablet or physical book to read a story. Students are not expected to explain technical details about how the tools work or compare them based on efficiency. Students do not need to develop knowledge of specialized hardware, software, or design processes.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Curiosity

E1-SAS-12: Describe the purpose of basic hardware components of a computing system, using accurate terminology.

Boundary Statement(s)	Students should use correct academic language to both identify the name and explain the basic function of external computing system hardware (e.g., laptop computer, tablet device, robot, keyboard, mouse, and other peripherals). Expressing that a laptop or tablet is used to run apps to complete school work or relating that headphones are used to hear sounds on a device would be appropriate. Students are not expected to identify or describe the function of internal computer system hardware. Students do not need to determine whether peripherals serve as input or output for a computing system.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Computational Thinking: 6. Define computational problems.
Disposition(s)	Curiosity, Critical Thinking

E2-SAS-12: Explain how the basic hardware components of a computing system, including sensors, work together to perform input and output (I/O) operations.

Boundary Statement(s)	Students should use academic language to name components of a computing system, identify whether they are input or output components, and explain how the components work together to take in information from the real world and produce an action that can be seen or heard by people. Explaining input and output operations of components like keyboards, monitors/screens, microphones, headphones, speakers, webcams/cameras on a robot, printers, light sensors, sound sensors, and distance sensors is appropriate. Students are not expected to explain the technical details (e.g., I/O ports and buses, data digitization) of how inputs and outputs work. Students do not need to explain processing and storage.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Curiosity

E3-SAS-12: Describe the role of software in a computing system to accomplish tasks or solve problems.

Boundary Statement(s)	Students should describe how software applications help with common tasks (e.g., writing a story, creating an image, playing a game). Students do not need to learn about operating system structures, cloud storage administration, or version control systems.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Resourcefulness

E4-SAS-12: Apply basic troubleshooting processes to identify and fix common hardware and software issues.

Boundary Statement(s)	Students should focus on following a consistent set of steps (e.g., define, plan, try, reflect) to address common hardware and software issues such as a device not responding, no network access, or an app crashing. Students should apply basic troubleshooting strategies such as restarting the device, checking connections, or closing and reopening applications. Students are not expected to solve all hardware and software issues they encounter. Students do not need to open devices to diagnose or troubleshoot internal hardware problems.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Human-Centered Design: 11. Use iterative design processes.
Disposition(s)	Critical Thinking, Resourcefulness

E5-SAS-12: Explain how hardware and software components of a computing system work together to perform input and output (I/O), processing, and storage.

Boundary Statement(s)	Students should focus on everyday examples of how hardware and software interact to complete a task (e.g., pressing a key on a keyboard sends input to the computer, which is stored in memory and processed with software to display a letter on the screen as output). Age-appropriate models, such as flow diagrams, role-play activities (students acting as a computer with input, processor, memory, and output), and creating physical computing projects that use sensors are appropriate. Students are not expected to design or analyze actual logic gate diagrams, understand machine-level binary representations, or study complex computer architecture. Students do not need to develop knowledge of circuitry or number base systems.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Computational Thinking: 6. Define computational problems.
Disposition(s)	Reflectiveness, Critical Thinking

MS-SAS-30: Examine differences between computing systems based on user needs, system requirements, and potential societal, environmental, and ethical impacts.

Boundary Statement(s)	Students should compare multiple computing systems based on features and system requirements (e.g., storage capacity, connectivity, accessibility features). Students should identify user needs (e.g., portability, cost, battery life) and provide reasoning for system choices. Students should connect at least one societal, environmental, or ethical impact to system choice (e.g., digital divide due to cost, energy consumption and e-waste, privacy and accessibility concerns). Students are not expected to explain detailed technical specifications (e.g., processor clock speeds, low-level architecture).
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 2. Respect others' rights when creating computational technologies. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Resourcefulness, Reflectiveness, Curiosity

MS-SAS-31: Describe computing devices used in industry, their basic functions, and how they are used to accomplish tasks and/or solve problems.

Boundary Statement(s)	Students should learn about various computing devices used in industry and describe their basic functions. Devices might include robots, electronic control units in vehicles, medical imaging devices, automated manufacturing equipment, and agricultural sensors. Students should identify how these devices use input, output, processing, storage, and (when applicable) mechanical interactions to accomplish specific tasks or solve problems within industries such as medicine, agriculture, or manufacturing. Students should have agency in selecting industries and devices to study. Students are not expected to build or program devices. Students do not need to demonstrate advanced technical fluency in hardware or software integration.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Reflectiveness, Resourcefulness, Curiosity

HS-SAS-30: Differentiate operating systems as a special type of software that manages both the hardware and other software components of a computing system, including handling memory and peripherals.

Boundary Statement(s)	Students should be able to differentiate an operating system from other types of software by identifying its role in managing hardware and software resources. For example, students might compare a computer's operating system to a video game, explaining that the operating system manages the computer's basic functions (like allocating memory and connecting to peripherals) while the video game is an application that relies on the operating system to run. Students are not expected to understand the intricacies of how an operating system is coded or to perform advanced tasks like configuring system drivers.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Computational Thinking: 7. Develop and use abstractions.
Disposition(s)	Critical Thinking

HS-SAS-31: Apply a physical or simulated computing device to address a real-world task or problem, demonstrating understanding of its capabilities and limitations.

Boundary Statement(s)	Students should be able to select and use a computing device, like a smartphone, single-board computer, or tablet, to solve a practical problem. For example, a student could use a tablet to manage project deadlines and task assignments for a group project, using its apps and cloud synchronization features to coordinate with team members, while also identifying limitations such as battery life or screen size. Students are not expected to build their own computing device, understand the internal architecture of the device, or write code that directly controls the device's hardware components.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Human-Centered Design: 12. Design computational technologies that empower and inform users.
Disposition(s)	Persistence, Critical Thinking, Resourcefulness

Security

EK-SAS-12: Differentiate between public and private information

Boundary Statement(s)	Students should focus on being able to describe the differences between information that can be safely made public (e.g., favorite food, favorite animal) and information that should be kept private (e.g., home address, family photos). Real-life scenarios, like deciding what is okay to share with a friend versus a stranger, are appropriate. Students are not expected to use technical vocabulary like "confidential". Students do not need to understand detailed internet safety rules or the technical aspects of data privacy or cybersecurity.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 2. Respect others' rights when creating computational technologies. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Sense of Belonging in CS, Critical Thinking

E1-SAS-13: Describe how to keep online accounts safe from unauthorized access.

Boundary Statement(s)	Students should focus on how authentication (e.g., passwords, QR codes for signing in) and good habits like signing out/logging off help keep their information protected online. Demonstrating how they sign out of accounts and keep their authentication details secret from others would be appropriate, if students have online accounts that require authentication. Students are not expected to create their own passwords or other authentication methods, or to describe more complex authentication methods (e.g., multi-factor authentication, passkeys, biometrics). Students do not need to have online accounts that require authentication; students can learn and describe these concepts without online accounts.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 2. Respect others' rights when creating computational technologies. Inclusive Collaboration: 5. Act responsibly in computing collaborations.
Disposition(s)	Critical Thinking, Reflectiveness

E2-SAS-13: Explain how online actions have real-world consequences and that laws and rules may also apply when online.

Boundary Statement(s)	Students should focus on age-appropriate examples that connect how their online actions (e.g., sharing information, posting comments, using a device) can have effects in the real world (e.g., hurting someone's feelings, compromising personal safety). Writing a story or drawing a picture to show how a specific online behavior (e.g., sharing a picture without asking) can lead to a consequence (e.g., making a friend angry) is appropriate. Discussing the consequences of copying someone's work online is also appropriate. Students are not expected to identify specific laws (e.g., CIPA, COPPA) or legal frameworks like copyright or intellectual property law. Students do not need to have online accounts; students can learn and describe these concepts without online accounts.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 2. Respect others' rights when creating computational technologies. Inclusive Collaboration: 3. Communicate effectively about computing.
Disposition(s)	Sense of Belonging in CS, Reflectiveness

E3-SAS-13: Distinguish between authentication and authorization in protecting devices and private information.

Boundary Statement(s)	Students should focus on authentication and authorization as two distinct but related concepts for protecting private information and devices. Using an analogy of a house is appropriate: authentication is proving you have the key to get in the door, while authorization is a parent telling you which rooms in the house you're allowed to enter once you're inside. Students are not expected to learn about different types of authentication (e.g., biometrics, multi-factor authentication) or technical details of how authentication or authorization work. Students do not need to use a variety of authentication methods.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing.
Disposition(s)	Critical Thinking, Reflectiveness

E4-SAS-13: Evaluate how sharing information online might reveal personally identifiable information (PII) and other details to people other than the intended recipients.

Boundary Statement(s)	Students should focus on developing skills to help them recognize risks in online situations and assess what and when to share information online. Activities using age-appropriate real-world situations (e.g., sharing their favorite book with a classmate, providing their address in response to a pop-up on a website saying they won a prize) are appropriate. Students should practice identifying how seemingly harmless information shared in contexts like online gaming (e.g., screen names, team names, game schedules) can unintentionally reveal PII to unintended recipients. Students are not expected to evaluate all risky online situations, including things like catfishing and data breaches. Students do not need to participate in situations where they could reveal their personally identifiable information.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 2. Respect others' rights when creating computational technologies.
Disposition(s)	Critical Thinking, Reflectiveness

E5-SAS-13: Describe the concepts of the CIA (Confidentiality, Integrity, Access) Triad and how each part is important in protecting information.

Boundary Statement(s)	Students should be able to explain the CIA Triad at a conceptual level and describe how each element (Confidentiality, Integrity, and Availability) helps protect information. For example, students might describe confidentiality as keeping a password secret and not sharing personally identifiable information online. They might describe integrity as ensuring that data is accurate and unchanged (e.g., checking whether a photo is real or AI-generated, or ensuring that grades in a gradebook haven't been improperly changed). They might describe availability as ensuring they can access their accounts, devices, apps, or gaming systems when they need them (e.g., the system isn't down or blocked). Students are not expected to implement technical security measures such as encryption or access control lists. Students do not need to understand cybersecurity systems or networks beyond how they affect students' daily lives.
Pillar(s) and Practice(s)	Inclusive Collaboration: 4. Manage computing projects. Human-Centered Design: 12. Design computational technologies that empower and inform users.
Disposition(s)	Critical Thinking, Reflectiveness

Systems & Security
Hardware and Software
Security
Networks
Impacts of Computing Systems

MS-SAS-32: Explain the effects of failing to use the CIA (Confidentiality, Integrity, Access) Triad.

Boundary Statement(s)	Students should recognize and explain what happens when each element of the CIA Triad is not maintained. When confidentiality is compromised, sensitive data is exposed (e.g., leaked passwords, shared personal information). When integrity is compromised, data becomes corrupted or altered (e.g., incorrect grades in a school database). When availability is compromised, systems become inaccessible (e.g., a denial-of-service attack blocking a website). Students should apply the CIA Triad to age-appropriate, relatable contexts (e.g., school accounts, social media, or online gaming, demonstrating understanding of how failures connect to real-world impacts). Students are not expected to conduct professional-level security audits or design encryption protocols. Students do not need to develop deep cryptography knowledge.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Inclusive Collaboration: 5. Act responsibly in computing collaborations.
Disposition(s)	Critical Thinking, Reflectiveness

MS-SAS-33: Evaluate common types of cyber attacks, including social engineering and malware, and preventions.

Boundary Statement(s)	Students should identify and describe common types of cyber attacks such as phishing, malware, ransomware, and social engineering. They should be able to evaluate how these attacks exploit human behavior and system vulnerabilities. Students could also explore real-world examples of cyber incidents and their consequences. Students should learn to apply basic prevention strategies, such as using strong passwords, enabling multi-factor authentication, and recognizing suspicious activity. Students are not expected to perform penetration testing or advanced threat modeling. Students do not need to write code, analyze malicious code, or implement enterprise-level security systems or protocols.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Resourcefulness, Reflectiveness, Curiosity

HS-SAS-32: Identify different types of cybersecurity and physical security measures and the trade-offs for users, data, and devices.

Boundary Statement(s)	Students should be able to identify and describe different types of cybersecurity and physical security measures, as well as their associated trade-offs. They should be able to articulate how a security measure may benefit one aspect (e.g., data protection) while posing a challenge to another (e.g., user convenience or cost). For example, students could identify how multi-factor authentication (MFA) enhances security for a user's account but may be less convenient for the user to access. Students should focus on conceptual understanding of security principles and their practical implementation. Students are not expected to implement these security measures or demonstrate technical expertise.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 2. Respect others' rights when creating computational technologies. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Reflectiveness

HS-SAS-33: Classify the causes and impacts of security breaches and social engineering attacks for individuals, industries, communities, and governments.

Boundary Statement(s)	Students should be able to research and categorize the consequences of a security breach or social engineering attack, differentiating between the effects on various stakeholders. The classification should go beyond a simple list of impacts and demonstrate an understanding of how the same breach can have different types of consequences (e.g., financial, reputational, legal, operational) for different groups. For example, a student could classify the impacts of a data breach at a hospital, analyzing the effects on patients (individuals), the hospital (industry), the local healthcare system (community), and regulations (government). Students should focus on understanding the broad, interconnected impacts of an attack. Students are not expected to conduct a forensic investigation of a real-world breach or to use professional cybersecurity tools for their analysis. Students do not need to focus on the technical details of the attack itself.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 2. Respect others' rights when creating computational technologies. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Reflectiveness

HS-SAS-34: Formulate a solution to a security flaw in a given system.

Boundary Statement(s)	Students should be able to analyze a simulated or described scenario involving a security flaw and propose a high-level solution. The proposed solution should explain the identified vulnerability, the potential consequences, and the recommended steps for mitigation. For example, a student could be given a case study of a school's network being compromised by a phishing attack and be asked to propose a solution that includes both technical measures (e.g., email filtering that detects suspicious links, multi-factor authentication) and non-technical measures (e.g., user training, security policies) to prevent future attacks. Students are not expected to implement the solution.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Resourcefulness

Networks

Standards do not begin until Grade 3.

E3-SAS-14: Explain how people access the Internet to gain information and communicate with each other.

Boundary Statement(s)	Students should be able to describe, in age-appropriate terms, how people use different devices to connect to the Internet and communicate. Examples of devices that can be used to access the Internet include computers, cell phones, and tablets. Examples of accessing information include using a search engine to look up facts for a report. Examples of communicating through technology include using video chat to talk with people across long distances. Students are not expected to describe technical infrastructure such as IP addressing, routing protocols, domain name systems, or data packet transmission. Students do not need to consider network configuration or errors.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Inclusive Collaboration: 3. Communicate effectively about computing.
Disposition(s)	Resourcefulness

E4-SAS-14: Critique the ways computing devices connect to the Internet, wired or wireless.

Boundary Statement(s)	Students should focus on comparing the advantages and disadvantages of different ways of connecting to the Internet. Important factors include distance, speed, convenience, and safety considerations (e.g., connecting to trusted networks at home or school versus unknown public networks). Tasks where students determine when it is advantageous to have either a wired connection or wireless connection are appropriate. Students are not expected to explore other ways of connecting between devices to share information (e.g., Bluetooth) or the protocols used to connect devices (e.g., TCP/IP). Students do not need direct access to routers or hubs.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing.
Disposition(s)	Critical Thinking, Reflectiveness

E5-SAS-14: Investigate the components of wired and wireless networks.

Boundary Statement(s)	Students should explore and describe the basic components of networks, such as devices (computers, tablets, phones), connecting hardware (routers, switches, cables), and wireless connections (Wi-Fi signals). Students are not expected to understand detailed networking protocols, IP addressing, or the technical differences between LANs, WANs, and the Internet. Students do not need to understand packet structures, routing tables, or advanced network design.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Computational Thinking: 7. Develop and use abstractions.
Disposition(s)	Reflectiveness, Curiosity

Systems & Security
Hardware and Software
Networks
Security
Impacts of Computing Systems

MS-SAS-34: Model how information in a network is broken down into packets, transmitted between devices, and reassembled.

Boundary Statement(s)	Students should be able to model how data is split into packets, encoded into binary, transmitted independently across a network, and reassembled at the destination. Students should also identify common issues such as packet loss, explain possible causes (e.g., congestion, interference, or faulty hardware), and describe basic strategies to reduce or prevent packet loss. Students are not expected to explore networking protocols in detail (e.g., TCP/IP) or to configure real-world networking hardware (routers, switches, servers).
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Computational Thinking: 7. Develop and use abstractions.
Disposition(s)	Critical Thinking, Curiosity

MS-SAS-35: Explain how the resilience of the Internet depends on the interconnected devices, including their roles and functions within the network.

Boundary Statement(s)	Students should be able to explain how the Internet is made up of interconnected devices that each serve specific roles. For example, routers direct traffic between networks, servers host information, clients (e.g., laptops or phones) request and use data, and switches connect devices within a local network. Students should also describe how redundancy (multiple pathways between devices) supports resilience, allowing communication to continue even if some devices or connections fail. Students are not expected to demonstrate technical knowledge of networking protocols (e.g., TCP/IP) or to configure or analyze real-world enterprise-level networking systems.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Computational Thinking: 6. Define computational problems.
Disposition(s)	Reflectiveness, Resourcefulness

Systems & Security
Hardware and Software
Networks
Security
Impacts of Computing Systems

HS-SAS-35: Diagram a network of computing systems, including hardware and software.

Boundary Statement(s)	Students should be able to create a diagram of a computing systems network that includes both hardware (e.g., servers, routers, storage, and user devices) and software (e.g., operating systems and applications). Students should focus on recognizing and representing the concept of a network and the role of each component in the network. Students should show how these components connect and interact to process information, accomplish tasks, or solve problems. Students are not expected to build a network using hardware or software.
Pillar(s) and Practice(s)	Computational Thinking: 7. Develop and use abstractions. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Creativity, Critical Thinking

HS-SAS-36: Analyze how the Internet functions as a network of networks, including similarities and differences between the Internet and other types of networks in terms of structure, protocols, and scalability.

Boundary Statement(s)	Students should be able to conceptually analyze the Internet's layered structure and its role as a global network connecting smaller, local networks. This includes identifying key components such as routers, protocols like TCP/IP, and the client/server model. For example, a student could compare a simple local area network (LAN) in a classroom to the global structure of the Internet, explaining how the principles of each differ in terms of scale, addressing, and data transfer. Students should focus on developing a high-level, conceptual understanding of networking principles. Students are not expected to implement specific networks, configure or troubleshoot professional-grade networking equipment, write network protocols, or perform packet analysis.
Pillar(s) and Practice(s)	Computational Thinking: 7. Develop and use abstractions.
Disposition(s)	Critical Thinking, Resourcefulness

Impacts of Computing Systems

EK-SAS-13: Identify an individual's role in responsibly using computing systems and tools.

Boundary Statement(s)	Students should be able to identify the responsible use of hardware and software in terms of tangible, classroom-level expectations including caring for hardware (e.g., handling devices gently, keeping food and drink away from devices), making safe choices (e.g., asking permission before using a device, only using the app specified), and being a good citizen in a shared technology environment (e.g., not using someone else's account, using headphones instead of playing sound out loud). Students should focus on understanding that computing systems are used as tools for learning and should be used appropriately in the same way other tools are used in the classroom. Students are not expected to understand complex cybersecurity or digital citizenship topics.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Inclusive Collaboration: 3. Communicate effectively about computing.
Disposition(s)	Reflectiveness

E1-SAS-14: Describe an individual's role in responsibly using computing systems and tools.

Boundary Statement(s)	Students should focus on responsible use in terms of tangible, classroom-level expectations including caring for hardware (e.g., handling devices gently, keeping food and drink away from devices), making safe choices (e.g., asking permission before using a device), and being a good citizen in a shared technology environment (e.g., not using someone else's account, using headphones instead of playing sound out loud). Students are not expected to describe complex digital citizenship topics. Students do not need to understand intellectual property (e.g., copyright), plagiarism, how to identify misinformation, or the nuances of a digital footprint.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Inclusive Collaboration: 3. Communicate effectively about computing.
Disposition(s)	Reflectiveness

Systems & Security

Hardware and Software

Networks

Security

Impacts of Computing Systems

E2-SAS-14: Describe the benefits and harms that arise from an individual's use of computing technology.

Boundary Statement(s)	Students should describe benefits and harms in the context of their own personal experiences, feelings, and well-being. The concepts should be simple and concrete. Describing benefits such as learning new things (e.g., watching a video about animals), having fun (e.g., playing a creative game), and connecting with family and friends (e.g., video calling a relative) would be appropriate. Describing harms such as physical effects of too much screen time (e.g., tired eyes, feeling grumpy), feeling sad or left out by something they see, or accidentally seeing something confusing or scary would also be appropriate. Students are not expected to describe complex or large-scale societal harms. Students do not need to understand issues like the digital divide, the spread of misinformation, or data privacy.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing.
Disposition(s)	Critical Thinking, Reflectiveness

E3-SAS-15: Describe how widely used computing technologies may impact an individual's life and community.

Boundary Statement(s)	Students should be able to describe both benefits and harms (e.g., effects on safety, privacy, economy, and culture) of computing technology in wide use. They should be able to reason about specific scenarios and identify which individuals or communities (e.g., themselves, their parents, children, elderly people, teachers) are impacted by a particular benefit or harm and describe the nature of the impacts. Students are not expected to describe the benefits and harms of computing technologies or the impacts on communities without first learning about these technologies and communities. Students do not need to memorize a list of technologies and their impacts on specific communities.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 2. Respect others' rights when creating computational technologies. Human-Centered Design: 12. Design computational technologies that empower and inform users.
Disposition(s)	Reflectiveness, Resourcefulness

Systems & Security
Hardware and Software
Networks
Security
Impacts of Computing Systems

E4-SAS-15: Investigate the impacts for widely used computing technologies on natural resources and the environment.

Boundary Statement(s)	Students should investigate the environmental effects of widely used technology (e.g., artificial intelligence, computers, tablets, monitors). They should designate the effects of computing technology as harmful or beneficial to the Earth's environment and its natural resources. Students can consider large sources of impact (e.g., e-waste, burning fossil fuels for power, mining of materials, modeling of solutions using technology, conserving energy with "smart" systems) and more individual sources of impact (e.g., water and energy usage, frequent device upgrades, recycling/reuse of computing systems and components, information availability about the environment, reducing paper usage). Students are not expected to investigate the impacts of technology that is not in wide use (e.g., quantum computing). Students do not need specialized equipment or knowledge of science concepts beyond what is grade-level appropriate.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Ethics and Social Responsibility: 2. Respect others' rights when creating computational technologies.
Disposition(s)	Critical Thinking, Reflectiveness

Systems & Security
Hardware and Software
Networks
Security
Impacts of Computing Systems

E5-SAS-15: Examine how computing technologies impact culture and the ways people live and work.

Boundary Statement(s)	Students should examine the cultural impact of existing and emerging computing technologies that both help and hinder societal needs. Examples of appropriate topics include the different roles social media plays in our culture and society and how generative AI is changing the way people work. Students are not expected to identify or reference specific ethical frameworks (e.g., utilitarianism) or sociological concepts (e.g., functionalism). Students do not need to access or use the technologies being discussed (e.g., social media, generative AI).
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 2. Respect others' rights when creating computational technologies. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Reflectiveness

MS-SAS-36: Collaboratively improve the design of a computing system so it can be better used by people with different needs, abilities, and ways of thinking.

Boundary Statement(s)	Students should recognize that computing systems need to be designed with accessibility and inclusivity in mind. Examples include providing options for closed captions, adjusting font sizes or colors for readability, providing alternative input devices (like voice control or adaptive keyboards), or features that support neurodiverse user needs. Students should collaborate in teams to propose and refine designs that make a system more inclusive. Students may conduct simple peer testing to evaluate design improvements. Students are not expected to design hardware from scratch or write complex code for accessibility features. Students do not need to perform professional-level usability testing.
Pillar(s) and Practice(s)	Inclusive Collaboration: 5. Act responsibly in computing collaborations. Human-Centered Design: 12. Design computational technologies that empower and inform users.
Disposition(s)	Creativity, Sense of Belonging in CS

MS-SAS-37: Examine differences in access to computing systems, based on personal and social factors, including physical ability, geographic location, socioeconomic status, and age.

Boundary Statement(s)	Students should investigate how access to computing systems varies across groups and regions, considering factors like income, physical accessibility, community resources, and infrastructure. For example, students may compare access to broadband Internet in urban vs. rural areas, or explore how adaptive technologies support people with disabilities. Students should analyze examples of digital divides at personal (home access), school (device availability), and community (libraries, Wi-Fi hotspots) levels. Students may discuss the role of government or nonprofits in expanding access. Students are not expected to perform statistical analyses of large national datasets, but they may use pre-curated datasets, maps, or local surveys. Students do not need to perform advanced policy evaluation.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Curiosity

Systems & Security
Hardware and Software
Networks
Security
Impacts of Computing Systems

HS-SAS-37: Evaluate the rationales behind laws and policies governing the design and use of computing systems.

Boundary Statement(s)	Students should be able to analyze and articulate the reasoning (technical, legal, and social) behind various policies related to the design and use of computing systems. Students should consider the rationale for technical approaches (e.g., responsible design practices, policies requiring two-factor authentication for sensitive data), legal approaches (e.g., laws like the Children's Online Privacy Protection Act (COPPA)) and social approaches (e.g., social movements, political resistance, personal behavioral change) to shaping the design and use of computing systems. Students should focus on understanding why these policies exist, what problems they aim to solve, and their intended and unintended consequences. Students do not need to read and interpret the text of laws or other legal texts.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Reflectiveness

HS-SAS-38: Investigate how computing systems and infrastructure impact society and the environment, identifying who is affected and why.

Boundary Statement(s)	Students should be able to research, analyze, and articulate the societal and environmental impacts of computing, including the physical components and their lifecycle. This includes, but is not limited to, the energy consumption of data centers, the ethics of computing waste storage and disposal, and the social effects of mobile devices. For example, a student could investigate the supply chain of a mobile device including its human and environmental costs. Students are not expected to propose novel technical solutions to these large-scale problems, but rather to analyze and communicate the problems themselves.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact.
Disposition(s)	Critical Thinking, Reflectiveness

Computing & Society

History of Computing

EK-CAS-14: Identify computing technologies used in daily life that have changed over time.

Boundary Statement(s)	Students should recognize and recall basic visual differences of computing technologies used in school or by their own family members that have changed from older versions. Examples are identifying that the current family phone is smaller and flatter than an older phone a parent or grandparent used, or that the family watches movies on a tablet now instead of using a TV would be appropriate. Students are not expected to understand the internal technological advancements (e.g., faster processors, memory capacity) that caused the changes. Students are not required to know specific historical dates, names of inventors, or the technical functions of the different generations of devices.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing.
Disposition(s)	Reflectiveness, Curiosity

E1-CAS-15: Compare how one familiar daily activity was done before and after the introduction of a specific computing technology.

Boundary Statement(s)	Students should compare observable differences in how people completed a familiar task before and after a computing technology was introduced. Concrete, everyday examples, such as how people communicate (writing letters vs. sending text messages) or find information (using library books vs. searching online) are appropriate. Students are not expected to understand the technical details of how technologies work or their historical timelines. Students are not required to analyze societal shifts or economic impacts.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Inclusive Collaboration: 3. Communicate effectively about computing.
Disposition(s)	Reflectiveness, Curiosity

E2-CAS-15: Analyze the ways that people from different cultures, backgrounds, and time periods have designed computing technologies to help them solve problems and express themselves.

Boundary Statement(s)	Students should explore how computing technologies have emerged from people's efforts to meet specific needs and express ideas within their cultural and historical contexts. An analysis focused on how computing technologies have supported individuals with different abilities throughout history or changed the modes and frequency of communication is appropriate. Students are not expected to conduct detailed history on the evolution of computing as a whole. Students are not required to compare complex cultural or societal impacts of technology across time periods.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Sense of Belonging in CS, Reflectiveness

E3-CAS-16: Examine how computing innovations have changed the ways people live, work, or communicate over time.

Boundary Statement(s)	Students should explore and explain how computing innovations across different time periods have progressively changed daily life, work, or communication, focusing on concrete, observable changes in human behavior. Examining how finding information, communicating verbally and in writing, enjoying entertainment, shopping, doing tasks at school and work have evolved over time due to computing innovations are appropriate. Students are not expected to understand the technical details of how computing innovations work or to create timelines of all computing innovations. Students are not required to examine complex societal impacts of computing innovations.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Reflectiveness

Computing & Society
History of Computing
Emerging Technologies
Humans and Computing
Career Exploration

E4-CAS-16: Investigate the contributions of diverse and often overlooked individuals and communities in the history of computing.

Boundary Statement(s)	Students should explore stories and achievements of individuals and communities whose contributions to computing are often underrepresented in mainstream accounts. Students should focus on understanding the concept of diversity or lack thereof in computing history. Researching and explaining the contributions of women, people of color, and innovators from global communities is appropriate. Students are not expected to explore the factors that led to historical underrepresentation or exclusion in computing. Students are not required to memorize a comprehensive list of every historical figure.
Pillar(s) and Practice(s)	Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Sense of Belonging in CS, Curiosity

E5-CAS-16: Analyze how the inclusion or exclusion of diverse and often overlooked individuals and communities has shaped the design, development, and societal impact of computing technologies.

Boundary Statement(s)	Students should analyze historical and contemporary examples of computing innovations, identifying how inclusion or exclusion of certain groups influenced technological progress and access. Analyzing the development of early programming languages or accessibility tools would be appropriate. Students are not expected to conduct independent historical research or examine complex sociopolitical movements in depth. Students are not required to know specific dates, laws, or named individuals beyond those directly connected to major computing milestones.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Curiosity

MS-CAS-38: Compare the roles of individuals, communities, organizations, and governments in shaping computing technologies across major eras in computing history.

Boundary Statement(s)	Students should identify and describe examples of historical and contemporary individuals, communities, organizations, and governments that have influenced the development of computing technologies. Students explain the motivations of these individuals and groups in advancing computing. Students also recognize how these different actors have interacted with one another. Students should be able to recognize both positive and negative impacts of technological decisions. Students are not expected to memorize timelines, technologies, or related facts. They are not expected to conduct deep ethical or social analyses. Students are also not expected to evaluate or rank individuals or groups by their relative importance.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Curiosity

MS-CAS-39: Analyze intended and unintended impacts of historical computing technologies on society and the environment.

Boundary Statement(s)	Students should distinguish between the intended purpose of a computing technology and its unintended consequences. Students identify and describe clear societal and environmental impacts, such as changes in communication, employment, privacy, e-waste, and energy consumption. They make simple cause-and-effect connections between computing technologies and their outcomes. For instance, recognizing that increased use of artificial intelligence can raise energy demands and contribute to higher power costs for consumers Students are not expected to conduct cost-benefit analyses or perform complex calculations to determine environmental impacts. They are also not expected to make detailed predictions about future technologies or develop solutions to the unintended effects they identify.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Inclusive Collaboration: 5. Act responsibly in computing collaborations.
Disposition(s)	Critical Thinking, Reflectiveness

HS-CAS-39: Analyze the historical trajectory of specific computing technologies and how their development is linked to social, political, environmental, and economic factors.

Boundary Statement(s)	Students should be able to investigate a computing technology (e.g., the internet, a social media platform, or a specific type of integrated sensor technology) from its inception to its current state, identifying key moments where non-technical forces directly influenced its design, adoption, or ethical challenges. They should use reliable secondary sources to draw and support their analytical conclusions. For example, high school students could research the rise of mobile computing, analyzing how shifts in economic factors (e.g., globalization of supply chains lowering hardware cost) and social trends (e.g., demand for instant communication) drove its rapid adoption and evolution. Students are not expected to conduct or synthesize research on the primary source documents of these technologies, nor are they required to master economic models or complex legal frameworks related to technology regulation.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Inclusive Collaboration: 3. Communicate effectively about computing.
Disposition(s)	Critical Thinking, Reflectiveness

HS-CAS-40: Propose modifications to existing policies and legislation that encourage ethical innovation and minimize societal risks associated with technology.

Boundary Statement(s)	Students should be able to articulate the rationale behind policies and legislation that guide technological development. This includes explaining how specific laws, like data privacy regulations or rules about algorithmic transparency, are designed to address societal concerns such as discrimination, a loss of privacy, or job displacement. Students should focus on understanding the why behind technology policy rather than the how of lawmaking. For example, a student could justify the need for legislation like the European Union's General Data Protection Regulation (GDPR) by explaining how it gives individuals more control over their personal data, thereby minimizing the risk of privacy breaches and misuse. Students are not expected to draft legal documents, interpret complex legal jargon, or have a deep understanding of the legislative process itself.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Ethics and Social Responsibility: 2. Respect others' rights when creating computational technologies.
Disposition(s)	Critical Thinking, Reflectiveness

Emerging Technologies

Standards do not begin until Grade 3.

E3-CAS-17: Describe how new technologies create both benefits and risks in personal and family life.

Boundary Statement(s)	Students should describe how new technologies (both that they interact with on a daily basis and those they may have learned about in other ways) may not have an exclusively positive or negative impact on their personal or family life. They should consider examples of both benefits and risks. Examples such as video calls with relatives, online learning, generative AI, or using smart devices at home are appropriate. Students are not expected to describe complex social or economic impacts nor complex new technologies (e.g., quantum computing).
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Inclusive Collaboration: 3. Communicate effectively about computing.
Disposition(s)	Reflectiveness

E4-CAS-17: Analyze how the limitations of existing technologies can lead to emerging technologies.

Boundary Statement(s)	Students should be able to identify a limitation in an older or existing technology (e.g., a weakness, something that doesn't work well, or a problem) and logically explain how overcoming that limitation could motivate the creation of a new, emerging technology, focusing on cause-and-effect reasoning in innovation. Analyzing both past (e.g., Mainframe computers to laptops, older mobile phones to smartphones) and more current emerging technologies (e.g., auto complete to generative AI, older model cars to self-driving cars) is appropriate. Students are not expected to research the actual motivations for the creation of technologies.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Reflectiveness, Curiosity

E5-CAS-17: Examine how people decide whether or not to use emerging technologies.

Boundary Statement(s)	Students should analyze the factors people consider when making personal decisions about using, avoiding, or rejecting emerging technologies, including personal needs and values, benefits, concerns, and accessibility. For example, students might examine why a person would choose to use or reject a smartwatch or voice assistant at home by considering convenience, cost, privacy concerns, and accessibility for family members with varying abilities. Students are not expected to do research studies or understand the technical aspects of how the emerging technologies examined work. Students are not required to make value judgments about whether people's decisions are "right" or "wrong."
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Inclusive Collaboration: 3. Communicate effectively about computing.
Disposition(s)	Reflectiveness, Curiosity

MS-CAS-40: Evaluate when it is appropriate to use AI and other emerging technologies to solve a problem based on their capabilities, limitations, and environmental impacts.

Boundary Statement(s)	Students should be able to identify what emerging technologies (e.g., AI, robotics, virtual reality, quantum computing) can and cannot currently do, and determine whether they are appropriate for addressing a specific problem. They should consider factors such as technical capabilities, reliability, infrastructure requirements, environmental impacts, and the maturity or readiness of the technology. For example, students might evaluate whether robotics is suitable for automating a farm task or whether quantum computing is realistic for improving classroom security systems. Students are not expected to understand the underlying mechanics of these technologies, predict future breakthroughs, or design functioning prototypes. They are also not expected to engage in ethical debates about regulation or human rights implications.
Pillar(s) and Practice(s)	Inclusive Collaboration: 4. Manage computing projects. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Reflectiveness, Critical Thinking

MS-CAS-41: Evaluate how design decisions in emerging technologies influence user experiences differently across different communities.

Boundary Statement(s)	Students should be able to evaluate how design decisions in new and emerging technologies affect people's experiences across different communities. They explore how choices in areas like accessibility, language, cost, or connectivity shape who can use and benefit from a technology. Students examine examples of technologies that may unintentionally exclude certain users and suggest ways to make them more inclusive. Teachers might observe students comparing real products or platforms and discussing which design choices create barriers or open access. These conversations help students see how thoughtful, human-centered design can make computing more equitable and responsive to the needs of all people. Students are not expected to conduct full usability studies or design professional prototypes.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 2. Respect others' rights when creating computational technologies. Inclusive Collaboration: 3. Communicate effectively about computing.
Disposition(s)	Reflectiveness, Resourcefulness

MS-CAS-42: Debate ways an emerging technology impacts the social, cultural, and ecological issues in their communities.

Boundary Statement(s)	Students should be able to provide a list of issues at the community level that have benefited or could benefit from an emerging technology. Focusing on a specific emerging technology, students should be able to construct arguments with references for a debate on positive and negative impacts of that technology. Students could either debate which applications of an emerging technology have the greatest or least impact, or argue either for or against a particular application. Students should attend to who is impacted and how they are impacted as well as costs and harms to the community and environment. Students are not expected to understand all the technical details of emerging technologies.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Creativity, Persistence, Resourcefulness, Critical Thinking

HS-CAS-41: Hypothesize how AI or another emerging technology could lead to enhancements or alternative approaches for an existing computing system or device.

Boundary Statement(s)	Students should be able to propose a plausible idea for how a new or developing technology could be integrated into an existing computing system or device to improve its performance or offer a new way of accomplishing a task. This requires a basic understanding of the emerging technology's function and the existing system's purpose. For example, a student could hypothesize that a virtual reality (VR) system could replace a traditional web browser by creating an immersive, three-dimensional space for searching and interacting with information, which would be an alternative approach to a two-dimensional screen. Students are not expected to have the technical knowledge to build or program the new system, nor are they required to understand the complex science behind the emerging technology.
Pillar(s) and Practice(s)	Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Creativity, Critical Thinking

HS-CAS-42: Evaluate the societal and environmental impacts of emerging technologies, including those that lead to inequities in access and outcomes.

Boundary Statement(s)	Students should be able to critically analyze an emerging technology by identifying and assessing its potential positive and negative effects on society and the environment. This includes considering how the technology might create or worsen social inequities in access and outcomes. Students should focus on developing a well-reasoned, conceptual argument about an emerging technology's broader impacts. For example, a student could evaluate the use of generative AI in education. They would identify the potential positive impacts, like providing personalized tutoring, but also the negative ones, such as the potential for academic dishonesty and the risk of widening the digital divide for students without access to these tools. Students are not expected to conduct a formal, data-driven sociological or environmental study.
Pillar(s) and Practice(s)	Human-Centered Design: 10. Understand and involve diverse users in design decisions. Human-Centered Design: 12. Design computational technologies that empower and inform users.
Disposition(s)	Critical Thinking, Reflectiveness

HS-CAS-43: Design a conceptual or prototype solution to a real-world problem using an emerging technology, supported by credible research and an ethical analysis of its potential benefits and harms to people and the environment.

Boundary Statement(s)	Students should be able to design a conceptual or tangible solution to a real-world problem by applying an emerging technology. This process must be supported by credible research on both the problem and the technology's capabilities. The solution should also be mindful of ethical implications, with students analyzing the potential benefits and harms to different groups of people and the environment. For example, a student could create a design document for a system that uses AI to analyze satellite imagery to help track deforestation, addressing a real-world environmental problem. Their work would include research to support their claim and a discussion of the ethical considerations, such as the potential for misuse of the data. Students are not expected to build a fully functional, production-ready version of their solution. Students are not expected to engage in complex engineering.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Computational Thinking: 6. Define computational problems.
Disposition(s)	Creativity, Critical Thinking

Humans and Computing

EK-CAS-15: Describe that people design and develop computing technologies.

Boundary Statement(s)	Students should recognize that people make technology and understand that they can be creators too. Examples of diverse creators (e.g., people of different ages, genders, and backgrounds) who make simple tools or technologies are appropriate. Students are not expected to understand the technical details of how technologies are built. Students are not required to build their own functioning electronic computing technologies.
Pillar(s) and Practice(s)	Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Sense of Belonging in CS

E1-CAS-16: Differentiate between activities that humans do well and activities that computing technologies do well.

Boundary Statement(s)	Students should categorize and explain simple, observable differences in what humans and computing technologies do best for various tasks. Examples of human strengths such as creativity, understanding feelings, and solving new problems are appropriate. Examples of computing technology strengths such as repeating tasks without mistakes, calculating quickly, and remembering lots of information are appropriate. Students are not expected to understand the cognitive science behind human intelligence or the technical details of how computing technologies perform certain tasks. Students are not required to analyze ethical or societal consequences that arise from humans or machines performing specific tasks.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Curiosity

Computing & Society
History of Computing
Emerging Technologies
Humans and Computing
Career Exploration

E2-CAS-16: Investigate situations where humans have created computing technologies to solve problems.

Boundary Statement(s)	Students should investigate familiar, concrete examples of how people have designed computing technologies to address everyday problems they can relate to. Technologies with clear, observable purposes are appropriate, such as traffic lights that help cars and pedestrians move safely, barcode scanners that help stores track inventory, automatic doors that help people enter buildings, weather apps that help families plan activities, or classroom timers that help manage transitions. Students might investigate how these technologies were created to solve specific problems like "cars were crashing at intersections" or "it was hard to keep track of library books." Students are not expected to understand how computing technologies work internally or to explain programming, engineering, or data processes. Students are not required to analyze historical development, compare technologies, or address advanced issues such as data privacy, cybersecurity, artificial intelligence, or automation.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Sense of Belonging in CS, Critical Thinking

E3-CAS-18: Examine why people design and build computing technologies, including AI.

Boundary Statement(s)	Students should explore the basic reasons computing technologies were created and communicate the needs or problems these technologies were likely intended to address, focusing on the human intent or goal. Examining how computing technologies are designed to meet needs, solve specific problems, make tasks easier, or provide entertainment is appropriate. Students are not expected to understand the technical details of how computing technologies work. Students are not required to compare multiple technologies to determine which best meets a particular need.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Reflectiveness

E4-CAS-18: Distinguish between the ways humans learn and the ways computing technologies learn.

Boundary Statement(s)	Students should compare and contrast basic characteristics of human learning and machine learning using simple, observable examples. For instance, humans learn by making sense of experiences and drawing on prior knowledge, while computing technologies learn by detecting patterns or correlations in data. Students are not expected to understand mathematical modeling, neural networks, or algorithmic training processes in technical detail nor are they expected to understand human cognition in detail. Students are not required to use or code actual AI models or apply formal data training procedures.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking

E5-CAS-18: Evaluate when it is appropriate to use or not use computing technologies to solve a problem.

Boundary Statement(s)	Students should evaluate and justify whether computing technologies are the best tools for solving a specific problem by considering the context, potential benefits, and potential drawbacks. These potential benefits and drawbacks can include technical, ethical, or environmental considerations. Students should focus on analyzing the feasibility of whether or not the solution requires computing. Deciding that AI is helpful for quickly translating a simple sentence but is not appropriate or helpful for deciding who should receive the last slice of pizza, which requires human judgment and social skills, is appropriate. Students are not expected to understand ethical frameworks, legal implications, or technical performance metrics (e.g., error rates, training data bias) used by professional engineers to make these decisions. Students are not required to solve the problem.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Resourcefulness

MS-CAS-43: Analyze how the decisions humans make when using computing technologies influence ethical and social outcomes.

Boundary Statement(s)	Students should analyze how human choices in the design and use of computing technologies influence fairness, equity, privacy, and social well-being. They should investigate how decisions such as what data to collect, which algorithms to use, or how results are interpreted can lead to different outcomes for individuals and communities. Examples might include analyzing how recommendations on social media affect what people see online, how automated grading or hiring systems might reinforce bias, or how accessibility features can expand inclusion. Students should explain how intentional human decisions, rather than the technologies alone, shape these ethical and social impacts and propose ways to use computing more responsibly. Students are not expected to design or program full computing systems, perform advanced algorithmic audits, or resolve complex ethical dilemmas. Students are not required to take positions on philosophical questions about artificial intelligence, consciousness, or rights.
Pillar(s) and Practice(s)	Inclusive Collaboration: 4. Manage computing projects. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Reflectiveness

HS-CAS-44: Evaluate how human choices in using, designing, deploying, and regulating computing technologies influence their risks, benefits, and long-term impacts.

Boundary Statement(s)	Students should evaluate the ethical and societal implications of computing technologies by examining how choices made by developers, users, and regulators shape their outcomes. Students should be able to analyze a case study, such as the use of an algorithmic system in a loan approval process, to identify the human choices (e.g., data selection, model tuning, deployment criteria) that could introduce bias and discuss the resulting risks (e.g., discriminatory outcomes) and benefits (e.g., increased efficiency). Students are not expected to train a complex machine learning model from scratch or perform a deep mathematical analysis of different AI algorithms (e.g., backpropagation in neural networks). Students should not focus on the underlying code or mathematics.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Inclusive Collaboration: 3. Communicate effectively about computing.
Disposition(s)	Critical Thinking, Reflectiveness

HS-CAS-45: Debate perspectives on the necessary differences between human and artificial intelligence, including implications for sentience, consciousness, ethics, rights, and societal responsibilities.

Boundary Statement(s)	Students should debate the philosophical and ethical boundaries between human intelligence (HI) and artificial intelligence (AI), focusing on core concepts like Artificial General Intelligence (AGI) and the resulting implications for an AI's ethical status or rights. Students should be able to analyze and articulate arguments for and against the possibility of creating an AGI and discuss what societal responsibilities (e.g., accountability, legal personhood) might arise if such an event occurred. Students are not expected to deeply analyze the neurological or biological mechanisms of human consciousness or the mathematical frameworks of complex AI models (e.g., specific neural network architectures).
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Reflectiveness

Career Exploration

EK-CAS-16: Identify how people use digital devices in their homes, schools, and work.

Boundary Statement(s)	Students should recognize, label, and state ways people use digital devices for daily work, learning, routines, and play. Examples such as using a digital thermometer to check temperature before going outside or parents checking work emails on a phone are appropriate. Students should participate in classroom discussions and activities where digital devices are used for communicating, learning, or helping with a task. Examples such as using manipulatives, role-play, or simple digital tools are appropriate. Students are not expected to name all device types or explain technical details. Students are not required to operate digital devices independently or using advanced features of digital devices.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Computational Thinking: 6. Define computational problems.
Disposition(s)	Sense of Belonging in CS

E1-CAS-17: Describe how computing is used by people in your life at home, school, and work.

Boundary Statement(s)	Students should identify and describe simple, familiar examples of how people use computing devices every day at home, school, or in the community. Examples such as a teacher using a tablet to read a story, a parent using a phone to call family, or a sibling using a computer to do schoolwork are appropriate. This practice of observing people's current use of computing aligns with human centered design practices and can assist students in understanding users' needs. Students are not expected to describe the internal parts of computers or the complex inner workings of the computing systems technology. Students are not required to use the computing devices themselves or to know specific technical terms.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Reflectiveness

Computing & Society
History of Computing
Emerging Technologies
Humans and Computing
Career Exploration

E2-CAS-17: Investigate how your personal interests connect to computing in different industries and careers.

Boundary Statement(s)	Students should explore and describe, in simple terms, how their passions and hobbies connect to computing. They should also investigate how computing is used in different jobs and industries that relate to those interests. For example, a student who loves drawing could learn how artists create digital illustrations or a student who enjoys sports exploring how coaches use computers to track scores and performance are appropriate. Students are not expected to understand technical details about how computing systems work, specific job qualifications, or the exact tools used in various careers. Students are not required to conduct in-depth career research or simulate real-world industry work.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Computational Thinking: 6. Define computational problems.
Disposition(s)	Sense of Belonging in CS, Curiosity

E3-CAS-19: Explain how people in different industries use computing technologies and skills to accomplish their work.

Boundary Statement(s)	Students should describe how people in different fields (e.g., healthcare, transportation, entertainment) use computers and basic computing skills (e.g., finding information, storing data, communicating, creating media) to do their jobs. Examples such as a bus driver following GPS routes, an animator creating cartoons using digital tools, or a shop owner tracking inventory on a laptop are appropriate. Students are not expected to understand how the identified technologies work or to demonstrate the referenced computing skills. Students are not required to research industries or careers in depth or understand training requirements.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Resourcefulness, Curiosity

Computing & Society
History of Computing
Emerging Technologies
Humans and Computing
Career Exploration

E4-CAS-19: Investigate how the workforce adopts new computing technologies and continues to update their computing skills.

Boundary Statement(s)	Students should examine and analyze how people in a variety of careers have learned and updated computing skills to adapt to changes in their work. Examples such as doctors learning to use digital records, teachers adopting video conferencing tools to teach remotely, farmers using GPS technology to plant and cultivate crops more precisely, or programmers applying AI tools to shorten development and debugging time are appropriate. Students are not expected to conduct formal research or interviews with the workforce. Students are not required to demonstrate the computing skills or use the tools used by the workforce.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Persistence, Curiosity

E5-CAS-19: Examine how professionals collaborate while using computing technologies to solve problems.

Boundary Statement(s)	Students should focus on being able to describe ways different professionals (e.g., scientists, doctors, artists, engineers) work together on a task using common digital tools for communication and sharing information. Real-life scenarios, like doctors sharing digital X-rays to diagnose a patient or architects using shared design software to plan a building, are appropriate. Students are not expected to understand the technical details of how the network or collaborative software functions. Students are not required to engage in complex collaborative projects or to use professional tools.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Reflectiveness, Curiosity

Computing & Society
History of Computing
Emerging Technologies
Humans and Computing
Career Exploration

MS-CAS-44: Analyze how workers in different careers use computational thinking to solve real-world problems.

Boundary Statement(s)	Students should investigate and describe examples of computational thinking across a range of professions. For example, they might analyze how a game designer uses algorithms to balance gameplay, how an environmental engineer uses data models to predict flooding, or how a social media manager applies pattern recognition to understand audience engagement. Students should be able to connect computational thinking concepts (such as abstraction, decomposition, or pattern recognition) to specific job functions, identifying parallels between how they think when coding and how professionals think when solving domain-specific problems. Students are not expected to explore specialized programming frameworks or conduct domain-specific simulations that require advanced mathematics, physics, or proprietary software.
Pillar(s) and Practice(s)	Inclusive Collaboration: 4. Manage computing projects. Computational Thinking: 6. Define computational problems.
Disposition(s)	Sense of Belonging in CS, Creativity

MS-CAS-45: Evaluate how automation in technology can create or replace jobs and change how people work.

Boundary Statement(s)	Students should explore how automation technologies (e.g., robotics, artificial intelligence, and machine learning) have changed or are changing the nature of work across industries, and analyze the social and economic trade-offs of automation. Examples where automation improves efficiency, accuracy, and safety, as well as cases where it disrupts or replaces jobs, are appropriate. Students should also consider how automation affects job tasks and skill requirements, raises ethical and human-centered questions, and prompts workers and industries to adapt through upskilling, reskilling, or new roles. Students are not expected to design or program automated systems, conduct in-depth economic analyses, or engage in labor-market forecasting.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Curiosity

Computing & Society
History of Computing
Emerging Technologies
Humans and Computing
Career Exploration

HS-CAS-46: Analyze narratives about how diverse teams of people used computational thinking and technologies to solve problems.

Boundary Statement(s)	Students should analyze professional narratives to understand the real-world application of computer science, including how CS is used at work, the kinds of challenges that practitioners face, and how they contribute to a more inclusive field. A concrete example of this could be having students analyze a podcast or video series featuring computer science professionals from diverse backgrounds and then identifying how they use their skills and have navigated workplace barriers. Students are not expected to conduct formal, academic research on these topics or to interview professionals themselves, but rather to analyze existing, publicly available narratives. Students are not required to solve complex social problems, but rather to understand and articulate the issues and solutions presented in the narratives.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Reflectiveness

HS-CAS-47: Connect computing knowledge and skills acquired to students' personal goals and career aspirations.

Boundary Statement(s)	Students should evaluate how their personal interests and career aspirations connect to computing knowledge and skills. This involves critically reflecting on their own strengths and passions and researching how these can be applied in various computing-related fields. Students are not expected to make a final career choice or to pursue an internship or work-based learning experience. Students are not expected to make a final commitment to a specific career path.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Reflectiveness, Resourcefulness

Specialty Standards for High School

What are Specialty Standards?

Specialty Standards define **advanced, domain-specific learning** beyond foundational PK–12 CS content. The Standards are organized into two tiers (Specialty I and Specialty II) across six high school specialty areas identified through the *Reimagining CS Pathways* project:

- Software Development
- Cybersecurity
- Data Science
- Physical Computing
- Artificial Intelligence (AI)
- Game Development

Specialty I standards cover the introductory knowledge and skills essential to a chosen specialty area, serving as the first dedicated learning experience in that domain. **Specialty II** standards describe advanced study within the specialty area, designed to prepare students for college-level coursework or industry-level certifications.

These specialty areas are intended for students who have a particular interest in one or more specific fields, including those who wish to pursue computing-intensive postsecondary education.

Additionally, one level of **X+CS Standards** has been developed to guide the integration of foundational high school CS content into other subject areas, like Journalism or Biology.

How do Specialty Standards Differ From Foundational Standards?

The Foundational Standards aim to prepare *every* student for a world powered by computing, while the Specialty Standards delineate learning outcomes beyond foundational CS that are aligned with particular specialty areas. These learning outcomes are intended to inform the development of learning experiences (including but not limited to **courses** and **pathways**) for students who choose to continue their study of CS beyond the foundational standards. The core differences between Foundational and Specialty Standards lie in the target audience, goal, and scope:

	Foundational Standards	Specialty Standards
Target Audience	**All** PK–12 students.	High school students who pursue continued CS learning **beyond** the foundational PK–12 standards.
Primary Goal	Develop CS knowledge, skills, and dispositions for **informed participation in society**, critical content consumption, responsible creation, and general problem-solving.	Develop **deeper, domain-specific knowledge, skills, and dispositions** that align with student interests and related pathways to college and/or career.
Scope	**Broad content** across five concepts: Algorithms & Design, Programming, Data & Analysis, Systems & Security, and Computing & Society.	**Focused, advanced** content progressions tailored to a CS subdiscipline (e.g., software development, artificial intelligence) or a CS-intensive discipline (e.g., data science, game design, computational art).

How to Implement Specialty Standards

Specialty Standards are flexible guidelines that can be implemented in a variety of ways depending on local context, resources, and student interest. Like the Foundational Standards, they can be implemented through standalone experiences, integrated with other disciplines, or a combination of the two.

1. **Develop Sequential Courses/Experiences:** Package the standards into discrete, sequential course pathways. These could focus on one specialty area or combine multiple areas. Samples are in the following table:

Strategy	Course 1	Course 2
Exploratory Specialty Pathway	**Course Title:** Programming the Future **Standards:** Subsets of Specialty I standards across Software Development, Artificial Intelligence, Cybersecurity, and Data Science **Pre-requisites:** Foundational CS **Sample rationale for offering:** Many students enjoyed their foundational CS experience and want to continue their learning. They want to maintain a broad view of computer science before they dive more deeply into a singular subdomain.	**Course Title:** Information and Network Security **Standards:** Cybersecurity Specialty II (and any Cybersecurity Specialty I standards that were not covered in Programming the Future) **Pre-requisites:** Programming the Future **Sample rationale for offering:** After offering Programming the Future, student interest seemed to favor cybersecurity, so an additional course opportunity was created in this area.
Focused Specialty Pathway	**Course Title:** AI Fundamentals **Standards:** Artificial Intelligence Specialty I **Sample rationale for offering:** The local community, including parents and employers, prioritize learning opportunities to build specialized knowledge in AI.	**Course Title:** Developing AI Applications **Standards:** Artificial Intelligence Specialty II **Sample rationale for offering:** Students developed their skills and confirmed their interest in AI through AI Fundamentals and wanted to continue to learn even more about how AI works and its applications.

2. **Augment a Foundational Course to Emphasize One or More Specialty Areas:** Incorporate content from a specialty area into foundational CS learning experiences to increase early exposure to specialty areas and/or create thematic foundational experiences. This might include using AI Specialty Standards to develop an AI unit that is incorporated into a foundational CS course or using cybersecurity Specialty Standards to dive deeper into Systems & Security concepts within a foundational experience.

3. **Guide Integrated Courses/Experiences:** Use the Specialty Standards to structure interdisciplinary courses/experiences that blend CS with other subjects (e.g., *Computational Biology* or *Computational Journalism*), ensuring the CS depth goes beyond the foundational level. This approach requires co-requisite knowledge in the non-CS discipline (X). X+CS specialty standards could be used across any discipline, however other specialty area standards can inform how CS concepts might be integrated into other disciplines as well (e.g., leverage Data Science Specialty Standards to integrate CS into math coursework). The table below shows a course progression that infuses CS and journalism:

Foundational Courses	X+CS Course
Computer Science Foundations Computer Science Foundations supports all high school students, regardless of postsecondary goals, in developing the knowledge, skills, and dispositions necessary to navigate and understand the technology-driven world in which they live. Course content, organized into five concepts: Algorithms & Design, Programming, Data & Analysis, Systems & Security, and Computing & Society. **Introduction to Journalism** Introduction to Journalism includes the fundamentals of gathering, writing, and reporting news stories across various media platforms. Students learn essential skills including conducting interviews, fact-checking, understanding media ethics and law, and recognizing different story formats such as news articles, features, and opinion pieces. The course typically emphasizes critical thinking, effective communication, and the role of journalism in a democratic society.	**Computational Journalism** Designed for students who have completed a foundational computing course as well as a foundational journalism course, this class exposes students to computational techniques and issues related to journalism, including: • Ethical issues, including data privacy and security • Language processing and text analysis • Reporting on technology and the technology industry • Data journalism, including data visualization • AI and its impact on the field of journalism

The Relationship Between Specialty Standards and Career and Technical Education (CTE)

Specialty Standards and CTE Frameworks serve different, yet compatible, purposes. Specialty Standards detail domain-specific learning for high school students who have already completed a foundational CS learning experience, while CTE typically includes a specific emphasis on workplace readiness and industry-recognized credentials. For those working in CTE spaces, Specialty Standards can:

- Align with existing CTE pathways,
- Supplement CTE courses and pathways, and
- Inform the refinement of CTE standards and curricula to increase the depth and/or breadth of CS content.

Specialty Standards are not exclusively intended to complement CTE experiences—they may also inform the development of academic courses and pathways, integrated studies, and informal or non-traditional learning experiences (e.g., out-of-school time).

Naming Conventions for Specialty Standards

Each of the identifiers for specialty standards follows a naming convention similar to the one for foundational standards:

Level		Focus area		Number
S#	-	YYY	-	##

The first two characters indicate Specialty I (S1) or Specialty II (S2) standards. The three characters in the next section indicate the focus area:

Abbreviation	Focus Area
SWD	Software Development
CYB	Cybersecurity
AIN	Artificial Intelligence
PHY	Physical Computing
DSC	Data Science
GMD	Game Development
XCS	X + CS

The last two digits are the standard number. Numbering begins with 01 for Specialty I standards in each focus area. Specialty II standards in each focus area begin with 01 as well.

Focus Area

Software Development

Cybersecurity

Artificial Intelligence

Physical Computing

Data Science

Game Development

X + CS

Software Development

Specialty I

S1-SWD-01: Apply linear data structures to organize and access collections of data when solving computational problems.

Boundary Statement(s)	Students should be able to analyze a computational problem and select appropriate linear data structures to store and manage data. Students should use arrays for fixed-size collections accessed by position, dynamic arrays (lists) for collections that grow or shrink, linked lists for efficient insertion and deletion, stacks for last-in-first-out access, and queues for first-in-first-out access. For example, students could use a dynamic array to store temperature readings collected over time, a stack to implement an undo feature in an application, or a queue to manage print jobs in the order they were submitted. Students should use built-in or library implementations and their associated methods. Students are not expected to implement these data structures or their operations (e.g., insert, delete, search) from scratch. Students do not need to use dictionaries, trees, graphs, or heaps.
Pillar(s) and Practice(s)	Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Critical Thinking, Persistence

Focus Area

- **Software Development**
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- Data Science
- Game Development
- X + CS

S1-SWD-02: Design software that accounts for scalability and manages complexity through abstraction.

Boundary Statement(s)	Students should apply design principles that promote maintainable and extensible software. Students should make intentional decisions about program structure, data organization, and relationships between components. They should consider how their software will handle growth (e.g., more users, more data, more features) and apply abstraction to manage complexity. For example, students could design a social media application that organizes data efficiently to support adding more users and posts without rewriting core functionality. Students do not need to handle scalability challenges requiring specialized infrastructure (e.g., multiple servers, distributed databases). Students are not expected to perform algorithm complexity analysis or optimize code for speed and memory usage.
Pillar(s) and Practice(s)	Computational Thinking: 7. Develop and use abstractions. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Persistence, Resourcefulness, Critical Thinking

S1-SWD-03: Use integrated development environment (IDE) features to streamline software development, including code editing, debugging, version control, and project management.

Boundary Statement(s)	Students should be able to use the core features of an IDE to increase their efficiency and effectiveness in developing software. This includes utilizing the IDE's built-in tools for syntax highlighting, autocomplete, and real-time error detection while writing code. Students should be able to use the integrated debugger to set breakpoints, step through code, and inspect variables to diagnose and fix logical errors. Students should also be able to interact with a version control system directly within the IDE to manage changes to their code, commit new versions, and collaborate on projects. The IDE's project management features, such as organizing files and folders and managing dependencies, should also be used to maintain a structured project. Students are not expected to deeply understand the underlying mechanisms of these tools (e.g., how the debugger works at a machine level), set up complex IDEs from scratch, or configure custom build processes. Students should focus on practical application and using the IDE as a powerful tool for a streamlined development workflow.
Pillar(s) and Practice(s)	Computational Thinking: 8. Create computational artifacts. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Resourcefulness

S1-SWD-04: Refine a user interface design using accessibility and responsive design tools.

Boundary Statement(s)	Students should be able to evaluate the effectiveness of user interface design choices in a digital product based on established principles such as contrast, alignment, proximity, and readability. For example, students could analyze a mobile app's user interface and provide a written critique of how element alignment and color contrast affect readability for users with low vision. Students are not expected to design graphics. Students are not expected to create design systems that specify detailed guidelines for all visual and interaction elements across a product.
Pillar(s) and Practice(s)	Human-Centered Design: 10. Understand and involve diverse users in design decisions. Human-Centered Design: 12. Design computational technologies that empower and inform users.
Disposition(s)	Critical Thinking, Reflectiveness

Focus Area

- **Software Development**
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- Data Science
- Game Development
- X + CS

S1-SWD-05: Apply the user experience design principles to create software that serves intended users and contexts.

Boundary Statement(s)	Students should be able to apply user experience design principles to evaluate and improve software for intended users and contexts. Students should conduct heuristic reviews, or systematic evaluations of software against established UX principles, using criteria such as usability, consistency, accessibility, and user control. Students should identify how well the software serves its intended users and contexts, considering factors like diverse abilities, linguistic backgrounds, and cultural contexts. Students should make design decisions or recommendations based on UX principles. For example, students might review an educational app and identify that it lacks keyboard navigation for users who cannot use a mouse, uses low color contrast making it difficult for users with visual impairments, and includes idioms that may not translate well across cultures. Students would then redesign or recommend changes to address these issues using established accessibility guidelines. Students are not expected to conduct user testing or collect feedback from actual users. Students are not expected to create accessibility solutions beyond applying established guidelines (e.g., WCAG standards).
Pillar(s) and Practice(s)	Inclusive Collaboration: 4. Manage computing projects. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Reflectiveness, Critical Thinking, Creativity

Focus Area

- **Software Development**
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- Data Science
- Game Development
- X + CS

S1-SWD-06: Evaluate AI-assisted test case recommendations to identify and address gaps in test coverage.

Boundary Statement(s)	Students should be able to critically evaluate test cases from both human and AI sources to identify coverage gaps, particularly edge cases and complex boundary conditions that human testers may overlook. For example, students could use an AI-powered coding or generative testing assistant to review their initial test suite, then determine which AI-recommended test cases would improve coverage. Students are not expected to design, train, or modify the underlying machine learning models used to generate AI test case recommendations.
Pillar(s) and Practice(s)	Computational Thinking: 8. Create computational artifacts. Computational Thinking: 9. Test and refine computational artifacts.
Disposition(s)	Critical Thinking, Persistence

S1-SWD-07: Use AI-assisted IDE features to understand unfamiliar code and identify errors during debugging.

Boundary Statement(s)	Students should be able to use AI-assisted IDE features for code comprehension and debugging tasks. This includes generating code summaries and documentation to understand unfamiliar or legacy code, and using intelligent code completion, error detection, and refactoring suggestions during debugging. For example, students could use an AI assistant to generate a summary explaining an unfamiliar function's purpose, inputs, and outputs before fixing a bug. Students should apply existing, commercially available or open-source AI IDE features. Students are not expected to manually configure or fine-tune the underlying AI/Machine Learning models within the IDE (e.g., modifying the weights or architecture of the code-generation transformer model), integrate external non-IDE based AI tools, or develop custom AI extensions for their IDE.
Pillar(s) and Practice(s)	Computational Thinking: 8. Create computational artifacts. Computational Thinking: 9. Test and refine computational artifacts.
Disposition(s)	Resourcefulness, Critical Thinking

Focus Area

- **Software Development**
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- Data Science
- Game Development
- X + CS

S1-SWD-08: Design thorough and systematic test cases that exercise the functionality of a program, considering potential edge cases, error conditions, and user inputs.

Boundary Statement(s)	Students should create a comprehensive test plan that includes a variety of test case types, such as unit tests and integration tests, and apply them systematically to a software project involving multiple functions and data structures. Students should design tests for valid inputs, edge cases (e.g., empty strings, extreme values, zero), boundary conditions, and error conditions (e.g., invalid data types, out-of-range values, missing required inputs, format violations). For example, when testing an input field that expects a number between 1 and 100, students should test valid values (e.g., 50), boundary values (e.g., 1, 100), values outside boundaries (e.g., 0, 101), and invalid types (e.g., text instead of numbers). When testing an input field that expects a string, students should test strings that contain special characters that might break code or formatting (e.g., single quotes, double quotes, semicolons, angle brackets, slashes) and ensure the program behaves robustly. Students are not expected to implement automated testing frameworks or manage quality assurance processes for large systems. Students are not expected to perform adversarial security testing (e.g., penetration testing).
Pillar(s) and Practice(s)	Computational Thinking: 8. Create computational artifacts. Computational Thinking: 9. Test and refine computational artifacts.
Disposition(s)	Critical Thinking, Reflectiveness

Focus Area

- **Software Development**
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- Data Science
- Game Development
- X + CS

S1-SWD-09: Develop software collaboratively, setting team norms to integrate diverse viewpoints, using software development best practices.

Boundary Statement(s)	Students should work in teams where different members contribute to different parts of a software project. Students should use version control tools to manage software development workflows, including creating and managing branches, performing code reviews, resolving merge conflicts, and using pull requests to integrate contributions. Students should demonstrate a capacity for inclusive collaboration by establishing team norms that solicit and integrate diverse perspectives on code design, user experience, and ethical considerations during development. For example, students could use version control tools to develop a multi-module program where team members submit and review pull requests for new features. Review discussions should address both technical implementation and broader considerations such as usability and ethical implications, ensuring that input from members with different areas of expertise (e.g., lived experience with a particular user community, user interface design, accessibility, data privacy) shapes the final decisions. Students are not expected to administer version control servers, manage repository security policies, or integrate version control with continuous integration/continuous deployment (CI/CD) pipelines.
Pillar(s) and Practice(s)	Inclusive Collaboration: 4. Manage computing projects. Inclusive Collaboration: 5. Act responsibly in computing collaborations.
Disposition(s)	Sense of Belonging in CS, Reflectiveness, Persistence

Focus Area

- **Software Development**
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- Data Science
- Game Development
- X + CS

Specialty II

S2-SWD-01: Apply associative and hierarchical data structures to solve computational problems.

Boundary Statement(s)	Students should be able to analyze a computational problem and select appropriate associative and hierarchical data structures including dictionaries (hash tables), trees, graphs, and heaps. Students should use built-in or library implementations and their associated methods. Students should use dictionaries for efficient key-value lookups, trees for hierarchical data or ordered collections, graphs for modeling networks and relationships, and heaps for priority-based access. Students should be able to explain relative efficiency trade-offs in practical terms (e.g., which operations are faster or use less memory). For example, students might use a graph structure to find the shortest path in a navigation app, a dictionary to quickly look up student records by ID, or a tree to organize a file system hierarchy. Students are not expected to implement these data structures or their operations from scratch. Students are not expected to use Big O notation to express time and space complexity.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Computational Thinking: 7. Develop and use abstractions.
Disposition(s)	Persistence, Critical Thinking

Focus Area

- **Software Development**
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- Data Science
- Game Development
- X + CS

S2-SWD-02: Develop a prototype using diverse user personas and user journey maps to guide design decisions.

Boundary Statement(s)	Students should be able to create and use user personas and user journey maps to guide ideation and prototyping. User personas represent different types of users and their characteristics, needs, and goals. User journey maps visualize how users interact with a product from initial contact through achieving their goals, highlighting pain points and opportunities for improvement. Students should use personas and journey maps in the early stages of design to guide and justify design decisions. For example, when designing an educational app, students might create personas representing different user types (e.g., a student with a learning disability, a highly motivated student, a parent) and develop user journey maps for each persona to ensure the design addresses diverse needs. Students are not expected to conduct market research to validate personas or produce fully functional applications.
Pillar(s) and Practice(s)	Human-Centered Design: 10. Understand and involve diverse users in design decisions. Human-Centered Design: 12. Design computational technologies that empower and inform users.
Disposition(s)	Creativity, Reflectiveness, Critical Thinking

S2-SWD-03: Develop a project in interactive cycles, documenting changes and the rationale for each cycle.

Boundary Statement(s)	Students should be able to manage a software development project using an iterative process. This means breaking the project into multiple distinct cycles of planning, implementation, testing, review, and documentation that continuously refine and build project features. Students should document each cycle with a clear record of changes made and the reasoning behind them (e.g., bug fix, new feature, response to user feedback). For example, in one cycle a student might build a basic user login system, in the next cycle add a user profile feature, and in a third cycle implement a search function. Students are not expected to use formal, complex project management methodologies like Agile or Scrum and their specific ceremonies (e.g., daily stand-ups, sprint retrospectives).
Pillar(s) and Practice(s)	Inclusive Collaboration: 4. Manage computing projects. Human-Centered Design: 11. Use iterative design processes.
Disposition(s)	Persistence, Reflectiveness, Resourcefulness

S2-SWD-04: Modify an existing algorithm to improve efficiency, considering factors such as data structures and algorithmic paradigms.

Boundary Statement(s)	Students should be able to analyze an existing algorithm and identify opportunities to improve efficiency in terms of time or space complexity (e.g., big-O notation). This involves understanding how different data structures (e.g., arrays, dictionaries, trees) impact performance and selecting a more suitable one. Students should be able to apply algorithmic paradigms such as divide-and-conquer to break a problem into smaller subproblems. Students should apply established principles to modify and optimize existing solutions, analyzing performance and making deliberate improvements. For example, a student might improve a linear search algorithm for a large, sorted dataset by using a more efficient data structure and applying binary search. Students are not expected to invent new algorithmic paradigms.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Computational Thinking: 7. Develop and use abstractions.
Disposition(s)	Critical Thinking, Persistence

S2-SWD-05: Implement the user experience design process by collecting data from intended users.

Boundary Statement(s)	Students should be able to conduct usability evaluations on digital interfaces by observing user interactions, collecting qualitative and quantitative feedback, and documenting findings in a structured format. For example, students could create a user survey or think-aloud protocol to test a peer's program or a real-world app and compile a report outlining specific usability issues (e.g., confusing navigation, inaccessible features). Students are not expected to obtain research ethics approval, recruit participants from outside of their peer group, or use specialized usability testing tools (e.g., eye-tracking software). Students are not expected to perform statistical analysis beyond calculating basic descriptive statistics for survey questions (e.g., average ratings, distribution of responses to Likert-type items).
Pillar(s) and Practice(s)	Computational Thinking: 8. Create computational artifacts. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Reflectiveness

Focus Area

- **Software Development**
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- Data Science
- Game Development
- X + CS

S2-SWD-06: Apply UI design principles and tools to create user-friendly, accessible, and responsively designed interfaces.

Boundary Statement(s)	Students should be able to create user interfaces for digital products that are usable, accessible, and responsive across different devices and screen sizes. For example, students could develop a web application that adapts its layout and functionality to be effective on smartphones, tablets, and desktop computers, ensuring that elements are navigable and legible for users with different needs. Students are not expected to create applications that function on every possible device and operating system or master responsive design frameworks.
Pillar(s) and Practice(s)	Computational Thinking: 8. Create computational artifacts. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Creativity

S2-SWD-07: Apply AI-assisted IDE features to test and refine complex software projects.

Boundary Statement(s)	Students should be able to apply AI-assisted features within an IDE to enhance the testing and refinement phases of a complex software project. This includes using features for automated test generation (e.g., unit test boilerplate or test cases), identifying redundant or inefficient tests, and receiving real-time suggestions for code optimization and performance improvements during the refinement phase. Students could use an AI tool to generate a basic suite of unit tests for a new class they have written, review and critique the generated tests for test coverage, and then accept the IDE's AI-driven recommendation to refactor a slow loop into a more performant structure. Students are not expected to perform deep-dive performance profiling, interpret low-level hardware performance counters, or manually implement complex statistical analysis of test results derived from large-scale, enterprise-level performance testing tools. The "complex software project" should remain within the scope of advanced high school application development (e.g., multi-module, multi-file application with external libraries).
Pillar(s) and Practice(s)	Computational Thinking: 8. Create computational artifacts. Computational Thinking: 9. Test and refine computational artifacts.
Disposition(s)	Critical Thinking, Persistence

Focus Area

- **Software Development**
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- Data Science
- Game Development
- X + CS

S2-SWD-08: Employ systematic debugging techniques on complex software projects to identify, isolate, and fix program errors, utilizing debugging tools and effective problem-solving strategies.

Boundary Statement(s)	Students should be able to apply formal debugging methodologies (e.g., divide and conquer, binary search, backtracking) to software projects involving multiple files, libraries, and external dependencies. This includes using advanced IDE debugger features such as setting conditional breakpoints, inspecting the call stack and watch window for complex data structures, and remote debugging. For example, students could debug a multi-module program with a subtle bug by first reproducing the error, isolating the fault by tracing state changes across functions, and then implementing a fix. Students should focus on logical program errors rather than systems-level failures. Students are not expected to debug at the operating system level, such as analyzing assembly code, directly manipulating memory registers, or performing post-mortem memory dump analysis for errors in production operating systems or hardware drivers.
Pillar(s) and Practice(s)	Computational Thinking: 8. Create computational artifacts. Computational Thinking: 9. Test and refine computational artifacts.
Disposition(s)	Persistence, Critical Thinking, Resourcefulness

Focus Area

- **Software Development**
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- Data Science
- Game Development
- X + CS

S2-SWD-09: Apply an industry-standard software development process to plan and deliver software projects while prioritizing equity and justice.

Boundary Statement(s)	Students should be able to select and apply an industry-standard software development process (e.g., Agile Scrum, Kanban, Waterfall) to a medium-to-large scale project. Students should effectively use tools (e.g., Trello, Jira, GitHub Projects) to manage tasks, track progress, and communicate with team members. Students should demonstrate that equity and justice are non-negotiable priorities that override technical or budgetary pressures, requiring them to articulate and justify project modifications based on ethical or societal impacts. For example, a student team could adjust a project timeline and scope to prioritize features that improve accessibility for users with disabilities or mitigate algorithmic bias, even if it delays the release or requires refactoring complex code. Students should work on projects of high-school appropriate complexity, emphasizing process application and ethical prioritization. Students are not expected to manage the budget or financial resources of the project, use advanced proprietary or enterprise-level project management software, or manage external stakeholders beyond their classroom or school environment.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Inclusive Collaboration: 4. Manage computing projects.
Disposition(s)	Sense of Belonging in CS, Critical Thinking, Reflectiveness

Focus Area

Software Development

Cybersecurity

Artificial Intelligence

Physical Computing

Data Science

Game Development

X + CS

Cybersecurity

Specialty I

S1-CYB-01: Analyze network services and protocols to explain their role in secure communication and potential vulnerabilities.

Boundary Statement(s)	Students should analyze the functionality and security implications of common network services and protocols (e.g., DNS, DHCP, ARP, BGP, SNMP), going beyond simply identifying protocols to critically evaluating their design and implementation flaws. For example, students could perform a Man-in-the-Middle (MitM) simulation in a safe, isolated lab environment to demonstrate the vulnerabilities of unencrypted protocols like HTTP and Telnet, and then implement and configure their secure counterparts (HTTPS, SSH) to mitigate those risks. Students are not expected to design or implement novel network security protocols or perform network penetration testing beyond specific controlled exercises. Students are not required to have kernel or hardware knowledge.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Resourcefulness

Focus Area

- Software Development
- **Cybersecurity**
- Artificial Intelligence
- Physical Computing
- Data Science
- Game Development
- X + CS

S1-CYB-02: Explain the concepts of the OSI model and its role in network communication.

Boundary Statement(s)	Students should be able to explain the specific function of each of the seven layers of the OSI model and articulate the concepts of encapsulation (data moving down the stack with headers/trailers added) and decapsulation (data moving up the stack with headers/trailers removed) as data travels between hosts. For example, students could make a web request from the Application Layer down through the stack on the sender's device and then back up on the receiver's server, identifying how the data unit changes (e.g., from segment to packet to frame). Students are not expected to memorize all the specific protocols associated with every layer of the 7-layer OSI model or the precise structure of all headers and trailers. They are also not expected to manually calculate checksums or implement custom network software that directly manipulates data at lower layers of the stack.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Computational Thinking: 7. Develop and use abstractions.
Disposition(s)	Critical Thinking

S1-CYB-03: Classify a network by its protocols, topologies, and addressing schemes.

Boundary Statement(s)	Students should be able to use three characteristics of a network (protocols, topologies, and addressing schemes) to precisely describe and differentiate between various real-world network architectures (e.g., classifying a network as a private LAN using a star topology, the TCP/IP protocol suite, and IPv4 addressing). Students could analyze the output from network configuration tools (e.g., ipconfig/ifconfig or a basic network scanner) on an unknown network to correctly identify the network type, its structure, and its addressing scheme, and explain how these elements interact. Students are not expected to perform detailed packet-level analysis of network traffic (e.g., using Wireshark to decipher raw hex data for obscure protocols) or implement network routing protocols from scratch.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Computational Thinking: 7. Develop and use abstractions.
Disposition(s)	Critical Thinking, Reflectiveness

Focus Area

- Software Development
- **Cybersecurity**
- Artificial Intelligence
- Physical Computing
- Data Science
- Game Development
- X + CS

S1-CYB-04: Investigate security risks in digital systems and corresponding mitigation strategies.

Boundary Statement(s)	Students should be able to investigate and model common security risks by focusing on three key failure areas: configuration weaknesses (e.g., default passwords, unnecessary open ports), insecure devices (e.g., unpatched software, weak encryption settings on IoT devices), and risky user behavior (e.g., poor password habits, clicking phishing links). Students should analyze case studies or use an isolated virtual environment to identify these risks and then propose appropriate mitigation strategies for each. For example, students could analyze a policy document and identify that "always using the default administrative password" is a configuration weakness, and the proposed mitigation strategy is "enforcing a complex password policy and disabling the default administrative account." Students are not expected to perform penetration testing against non-consensual targets or conduct live social engineering attacks on real users.
Pillar(s) and Practice(s)	Inclusive Collaboration: 5. Act responsibly in computing collaborations. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Reflectiveness

Focus Area

- Software Development
- **Cybersecurity**
- Artificial Intelligence
- Physical Computing
- Data Science
- Game Development
- X + CS

S1-CYB-05: Apply diagnostic tools and techniques to resolve network connectivity issues.

Boundary Statement(s)	Students should be able to systematically apply a sequence of diagnostic tools and techniques (e.g., ping, tracert/traceroute, ipconfig/ifconfig, netstat, basic packet capture analysis) within a simulated or isolated network environment to effectively troubleshoot and resolve connectivity issues. Students must analyze how a network's topology (e.g., faulty cable in a star network), protocols (e.g., incorrect IP address or subnet mask), and security configurations (e.g., an unintended firewall rule blocking a port) contribute to the failure. For example, students could use the OSI model as a framework to logically isolate a problem, use ping to test Layer 3 connectivity, and then check firewall logs to rule out a Layer 4 security block. Students are not expected to troubleshoot inter-domain routing failures, analyze encrypted commercial network traffic beyond basic header inspection, or physically replace or repair failed hardware components.
Pillar(s) and Practice(s)	Inclusive Collaboration: 5. Act responsibly in computing collaborations. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Persistence

Focus Area

- Software Development
- **Cybersecurity**
- Artificial Intelligence
- Physical Computing
- Data Science
- Game Development
- X + CS

S1-CYB-06: Use command-line programming to audit system processes, monitor network traffic, and scan for vulnerabilities.

Boundary Statement(s)	Students should be able to systematically use command-line tools within a virtual or isolated environment to perform basic security auditing tasks. Students should use the command line for defensive posturing, analysis, and auditing within a controlled environment. This includes using native operating system commands (e.g., PowerShell, Bash) and specialized tools (e.g., netstat, ps, nmap, or similar non-harmful network scanner) to audit system processes for unusual activity, monitor network traffic for open ports or unexpected connections, and scan for vulnerabilities on their own virtual machines or explicitly authorized lab targets. For example, students could write a short Bash script that automates the collection and logging of currently running processes and active network connections, then analyze this output to identify unauthorized services. Students are not expected to develop custom exploitation code, perform any scanning or auditing of real-world, external, or unauthorized systems, or interact with tools requiring industry certifications.
Pillar(s) and Practice(s)	Computational Thinking: 7. Develop and use abstractions.
Disposition(s)	Resourcefulness, Persistence

Focus Area

- Software Development
- **Cybersecurity**
- Artificial Intelligence
- Physical Computing
- Data Science
- Game Development
- X + CS

S1-CYB-07: Analyze how common cyber threats related to network activity exploit system vulnerabilities.

Boundary Statement(s)	Students should be able to identify and analyze common network-specific cyber threats by explaining how they work and clearly linking the attacks to the specific system or protocol vulnerabilities they exploit. This includes analyzing common attack types (e.g., IP spoofing exploits the trust model of IP, Distributed Denial of Service attacks exploit bandwidth or resource limitations, and passive packet sniffing exploits unencrypted network protocols). For example, students could analyze a scenario where a client receives a false DNS response and explain that this is likely a DNS spoofing attack exploiting the lack of authentication in the standard DNS protocol. Students are not expected to perform live attacks, develop mitigation code, or conduct security research to discover previously unknown vulnerabilities.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact.
Disposition(s)	Critical Thinking, Reflectiveness

S1-CYB-08: Compare encryption methods used in network communication and how they protect privacy and security.

Boundary Statement(s)	Students should be able to conceptually compare the fundamental mechanisms of common network encryption methods, focusing primarily on the distinction between symmetric key cryptography (e.g., AES) and asymmetric key cryptography (e.g., RSA), and then evaluate why the resulting confidentiality and integrity are essential for both individual privacy and organizational security. Students should go beyond basic data protection to include the societal implications of secure communications protocols like HTTPS and TLS/SSL. For example, students could compare the speed and key distribution methods of AES versus RSA and then evaluate why an unencrypted connection (HTTP) poses a critical privacy risk for online financial transactions. Students are not expected to mathematically derive or break encryption algorithms or investigate the policy requirements of certificate authorities.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Reflectiveness

Focus Area

Software Development

Cybersecurity

Artificial Intelligence

Physical Computing

Data Science

Game Development

X + CS

S1-CYB-09: Classify a security threat using the CIA triad, states of data, and types of control.

Boundary Statement(s)	Students should be able to classify any given cyber threat or security incident by systematically mapping its impact across three distinct security models: 1) the CIA triad (i.e., identifying which principle—confidentiality, integrity, or availability—is affected); 2) the state of data being targeted (i.e., at rest, in transit, or in use); and 3) the appropriate type of control needed for mitigation (i.e., preventative, detective, or corrective). For example, students could classify a ransomware attack as primarily impacting availability while the data is at rest, and determine the best response involves a corrective control—restoring from backups—and a preventative control—endpoint protection) Students are not expected to design and implement the security controls.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Computational Thinking: 7. Develop and use abstractions.
Disposition(s)	Critical Thinking, Reflectiveness

S1-CYB-10: Discuss security measures to protect sensitive information.

Boundary Statement(s)	Students should be able to discuss and categorize security measures beyond basic passwords and personal encryption, including technical, physical, and administrative controls. The discussion must cover common preventative measures like firewalls and network segmentation, detective measures like intrusion detection systems (IDS) and security auditing, and corrective measures like data backup and disaster recovery plans. For example, students could discuss the difference between a preventative technical control (e.g., a strong password policy) and a detective administrative control (e.g., security audit logs or incident reports). Students are not expected to implement or configure organizational systems or design organizational security policies. Students do not need to discuss the legal requirements of different compliance frameworks (e.g., HIPAA, GDPR) in detail.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Reflectiveness

Focus Area

- Software Development
- **Cybersecurity**
- Artificial Intelligence
- Physical Computing
- Data Science
- Game Development
- X + CS

S1-CYB-11: Explain the importance of cybersecurity policies in protecting organizational assets and mitigating risks.

Boundary Statement(s)	Students should be able to explain the crucial role of cybersecurity policies (e.g., Acceptable Use Policy, Remote Access Policy, Data Classification Policy) as the administrative and governance foundation for protecting organizational assets and mitigating risks. This explanation must detail how policies translate business and legal requirements into actionable rules that define expected user behavior, mandate technical controls, and establish accountability. For example, students could explain that a Data Classification Policy ensures that sensitive information is properly labeled and handled, which directly mitigates the risk of a data breach by mandating specific technical controls like encryption. Students are not expected to draft or revise organizational security policies for a real company or design a regulatory compliance program.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Reflectiveness

Focus Area
Software Development
Cybersecurity
Artificial Intelligence
Physical Computing
Data Science
Game Development
X + CS

S1-CYB-12: Identify key components of effective security policies.

Boundary Statement(s)	Students should be able to identify the key components of effective security policies by recognizing the elements that are necessary for a policy to be actionable, enforceable, and aligned with organizational goals. Key components include, but are not limited to: a clear purpose and scope (i.e., what the policy covers), the mandatory requirements/rules (i.e., what users must do), defined roles and responsibilities (i.e., who is accountable), and an enforcement/consequences section (i.e., what happens when the policy is violated). For example, students could analyze an Acceptable Use Policy and clearly identify the section that defines the acceptable resources and user activities (i.e., scope) separate from the section that details disciplinary actions for misuse (i.e., enforcement). Students are not expected to draft organizational policies or analyze the legal sufficiency of policy components.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Reflectiveness

S1-CYB-13: Explain social engineering techniques and how they exploit human cognitive biases and organizational weaknesses.

Boundary Statement(s)	Students should be able to deconstruct common social engineering attacks, identifying the specific human cognitive biases (e.g., authority, scarcity, urgency, or familiarity) and organizational weaknesses (e.g., poor physical security, lack of training) that the techniques exploit. For example, students could analyze a real-world phishing campaign and map the attacker's strategy to the psychological principles being manipulated (e.g., using a CEO's name to trigger an authority bias). Students are not expected to develop or perform real-world social engineering campaigns or study advanced psychological manipulation techniques beyond basic cognitive biases (e.g., neuro-linguistic programming or intelligence elicitation methods).
Pillar(s) and Practice(s)	Inclusive Collaboration: 5. Act responsibly in computing collaborations. Human-Centered Design: 12. Design computational technologies that empower and inform users.
Disposition(s)	Critical Thinking, Reflectiveness

Focus Area

- Software Development
- **Cybersecurity**
- Artificial Intelligence
- Physical Computing
- Data Science
- Game Development
- X + CS

S1-CYB-14: Analyze the potential consequences of cybersecurity decisions on individuals, organizations, and society.

Boundary Statement(s)	Students should be able to evaluate the systemic, ethical, social, and economic consequences of specific cybersecurity decisions on individuals, organizations, and society and the trade-offs inherent in those choices. For example, students could analyze a corporate decision to shift all data storage to a third-party cloud provider, evaluating the trade-offs between privacy policy changes (individual consequences), increased security and potential loss of control (organizational consequences), and systemic reliance on a single vendor (societal consequences). Students are not expected to perform financial risk assessments or quantitative cost-benefit analysis.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 2. Respect others' rights when creating computational technologies. Inclusive Collaboration: 5. Act responsibly in computing collaborations.
Disposition(s)	Critical Thinking, Reflectiveness

S1-CYB-15: Communicate cybersecurity concepts, risks, and solutions clearly to both technical and non-technical audiences.

Boundary Statement(s)	Students should be able to create and deliver tailored communications about cybersecurity topics, demonstrating the ability to translate technical details into clear, actionable language for a general audience while maintaining necessary specificity for technical audiences. For example, students could draft a technical report detailing a vulnerability for an IT team and a separate executive summary explaining the risk and mitigation steps for management and general users. Students are not expected to create publishable marketing materials or communications about active security incidents or other public relations crises.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Inclusive Collaboration: 3. Communicate effectively about computing.
Disposition(s)	Critical Thinking, Resourcefulness

Focus Area

- Software Development
- **Cybersecurity**
- Artificial Intelligence
- Physical Computing
- Data Science
- Game Development
- X + CS

Specialty II

S2-CYB-01: Integrate security features in networking hardware and software.

Boundary Statement(s)	Students should be able to configure and integrate security features found in networking hardware (e.g., routers, firewalls, access points) and software (e.g., host-based firewalls, intrusion detection systems, server configurations) to establish defense-in-depth. This includes setting up a VPN tunnel on a router, configuring Access Control Lists on a simulated switch or router, and integrating a host-based firewall on a server. For example, students could use virtual network labs or simulation tools to configure firewall security policies ensuring that only traffic from a specific subnet with encrypted credentials can access a particular server. Students are not expected to physically manipulate hardware, write custom firmware for network devices, or deploy security solutions on networks with real-time production traffic.
Pillar(s) and Practice(s)	Computational Thinking: 9. Test and refine computational artifacts.
Disposition(s)	Critical Thinking, Persistence

S2-CYB-02: Design a secure network, including servers, switches, routers, endpoints, and firewalls.

Boundary Statement(s)	Students should be able to articulate security requirements of a system (e.g., servers, switches, routers, endpoints, firewalls), select appropriate network components, and create a diagrammatic or virtual model of a secure network architecture that incorporates best practices including network segmentation (e.g., DMZ, VLANs) and defense-in-depth. For example, students could use network simulation tools (e.g., Cisco Packet Tracer) or diagramming tools (e.g., draw.io) to design a secure network for a small business, specifying the IP addressing scheme and firewall rules necessary to enforce the security policy. Students should focus on the design and justification of the architectural decisions. Students are not expected to physically install or configure hardware or deploy networks with more than 50 hosts.
Pillar(s) and Practice(s)	Human-Centered Design: 11. Use iterative design processes.
Disposition(s)	Critical Thinking, Persistence

Focus Area

- Software Development
- **Cybersecurity**
- Artificial Intelligence
- Physical Computing
- Data Science
- Game Development
- X + CS

S2-CYB-03: Analyze the security implications of different network topologies to identify potential vulnerabilities and mitigation strategies.

Boundary Statement(s)	Students should be able to analyze the inherent security implications of fundamental network topologies, identifying where a single point of failure or eavesdropping risk exists and proposing structured mitigation strategies for each risk. This includes articulating why a mesh topology is more resilient than a bus topology but more costly, and how a star topology's reliance on a central device creates a critical point requiring physical and logical hardening. For example, students could analyze a star network diagram, identify the central switch as the main vulnerability, and recommend specific mitigation strategies including physical access control, port security, and network monitoring. Students are not expected to perform quantitative risk assessments, mathematically model fault tolerance, or physically build and test networks.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Reflectiveness

Focus Area

- Software Development
- **Cybersecurity**
- Artificial Intelligence
- Physical Computing
- Data Science
- Game Development
- X + CS

S2-CYB-04: Analyze network traffic patterns to distinguish between normal and potentially malicious behavior.

Boundary Statement(s)	Students should be able to monitor and analyze network traffic patterns within a controlled, simulated environment using packet analysis tools (e.g., Wireshark) and system logs (e.g., firewall, web server, application logs) to distinguish between normal and potentially malicious behavior. This analysis must include identifying key indicators of compromise including unusual connection volume, unexpected port usage, or failed authentication attempts that deviate from established baselines. For example, students could analyze a packet capture to detect a port scan by recognizing a pattern of multiple connection attempts to different ports from a single source and comparing this against typical connection patterns. Students are not expected to configure Security Information and Event Management systems, perform real-time threat hunting on live networks, or write custom intrusion detection signatures.
Pillar(s) and Practice(s)	Computational Thinking: 9. Test and refine computational artifacts.
Disposition(s)	Critical Thinking, Persistence

S2-CYB-05: Use a scripting language securely to automate security operations.

Boundary Statement(s)	Students should be able to design and implement security automation scripts (e.g., using Python, Bash, or PowerShell) within a virtual, sandboxed environment to perform routine tasks including log file parsing, automated vulnerability scanning setup, or user account auditing. Students should focus on writing secure code. Students must use best practices to avoid common vulnerabilities including hardcoding credentials, improper input validation, or command injection. For example, students could write a Python script that uses secure libraries to fetch system logs via SSH while implementing appropriate error handling and input sanitization. Students are not expected to develop mission-critical security software, integrate scripts into large-scale Security Information and Event Management systems, or create custom exploits.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Critical Thinking, Reflectiveness

Focus Area

- Software Development
- **Cybersecurity**
- Artificial Intelligence
- Physical Computing
- Data Science
- Game Development
- X + CS

S2-CYB-06: Implement access controls to protect sensitive information from unauthorized access and data breaches.

Boundary Statement(s)	Students should be able to implement foundational access control mechanisms within a controlled virtual environment or software development project. This implementation must demonstrate user authentication (e.g., configuring strong password requirements or implementing multi-factor authentication), authorization (e.g., implementing Role-Based Access Control to restrict resource access based on user roles), and data encryption (e.g., using secure libraries to encrypt sensitive data files at rest or in transit). For example, students could develop a simple application that encrypts a user's sensitive data file using a standard encryption library and only decrypts it after the user successfully passes two-factor authentication. Students are not expected to design or implement cryptographic primitives.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Persistence

S2-CYB-07: Evaluate security risks to determine appropriate risk mitigation strategies.

Boundary Statement(s)	Students should be able to evaluate security risks by analyzing the likelihood of a threat exploiting a specific vulnerability and the magnitude of its potential impact on the CIA triad. Following this assessment, students should implement appropriate risk mitigation strategies (e.g., configuring a security control, updating software, changing a policy) to reduce the risk to an acceptable level within a simulated environment. For example, students could evaluate the risk of unpatched software, assess the high impact of a potential data breach, and then implement a mitigation strategy by configuring an automated patching schedule or applying security updates to a virtual machine. Students are not expected to perform quantitative financial risk analysis or develop novel mitigation technologies.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Inclusive Collaboration: 4. Manage computing projects.
Disposition(s)	Critical Thinking, Resourcefulness

Focus Area

- Software Development
- **Cybersecurity**
- Artificial Intelligence
- Physical Computing
- Data Science
- Game Development
- X + CS

S2-CYB-08: Discuss incident response plans and processes for real-world scenarios.

Boundary Statement(s)	Students should be able to discuss the structure and purpose of a formal incident response plan and identify the key steps in the incident response process (e.g., NIST's six phases: Preparation, Identification, Containment, Eradication, Recovery, and Lessons Learned). Students must then simulate incident response for real-world scenarios in a safe, virtual environment, focusing on the Identification, Containment, and Eradication phases and using critical thinking to make timely decisions about evidence preservation and system isolation. For example, students could respond to a simulated alert of suspicious activity, identify the threat type, contain the compromised system by isolating it from the network, and document the steps taken. Students are not expected to execute incident response plans during real attacks.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Persistence

S2-CYB-09: Debate how various regulations impact organizational security policies, procedures, and compliance.

Boundary Statement(s)	Students should be able to conduct a comparative analysis of major data protection and privacy regulations (e.g., HIPAA for healthcare data, GDPR for personal data, PCI DSS for payment data) and debate the specific, often costly, changes to organizational policies and procedures. The debate should center on the tension between enhanced individual rights and data protection versus increased administrative complexity and financial burden for organizations. Students must accurately identify core compliance requirements of at least two regulations. For example, students could debate GDPR's "Right to be Forgotten" provision, examining both the privacy benefit and the technical difficulty and cost of ensuring all data backups comply with deletion requests. Students are not expected to analyze the technical details of regulatory audits or create compliance frameworks.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Inclusive Collaboration: 5. Act responsibly in computing collaborations.
Disposition(s)	Critical Thinking, Reflectiveness

Focus Area

- Software Development
- **Cybersecurity**
- Artificial Intelligence
- Physical Computing
- Data Science
- Game Development
- X + CS

S2-CYB-10: Practice responsible disclosure of vulnerabilities and incidents in accordance with professional protocols.

Boundary Statement(s)	Students should be able to simulate the process of responsible disclosure, demonstrating an understanding of the ethical timeline, communication flow, and coordination required among researchers, vendors, and the public following the discovery of a vulnerability or security incident. For example, students could draft internal and external communications, including a CISO statement and a vendor notification email, adhering to a defined 90-day disclosure timeline. Students are not expected to discover or report vulnerabilities in real-world production systems.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 2. Respect others' rights when creating computational technologies. Inclusive Collaboration: 5. Act responsibly in computing collaborations.
Disposition(s)	Critical Thinking, Reflectiveness

S2-CYB-11: Document cybersecurity processes and decisions in a manner that supports team coordination and accountability.

Boundary Statement(s)	Students should be able to create formal documentation artifacts for cybersecurity processes including change management logs or system security plans, ensuring that documents clearly articulate the rationale, implementation details, and responsible parties for key decisions to support team coordination and accountability. For example, students could draft an Incident Response Playbook detailing roles, communication steps, and decision points, or a Security Architecture Diagram with accompanying change logs. Students are not expected to use ticketing or documentation software. Students are not expected to produce documents that fully comply with regulations.
Pillar(s) and Practice(s)	Inclusive Collaboration: 4. Manage computing projects. Inclusive Collaboration: 5. Act responsibly in computing collaborations.
Disposition(s)	Critical Thinking, Persistence

Focus Area

- Software Development
- **Cybersecurity**
- Artificial Intelligence
- Physical Computing
- Data Science
- Game Development
- X + CS

S2-CYB-12: Analyze the potential benefits, risks, and ethical implications of AI in cybersecurity, including its use in threat detection, incident response, and offensive cyber operations.

Boundary Statement(s)	Students should be able to analyze the dual-use nature of AI within cybersecurity by systematically outlining its potential benefits (e.g., faster threat detection through machine learning), technical risks (e.g., adversarial attacks against AI models), and ethical implications (e.g., bias in threat modeling, autonomous decision-making). The analysis must specifically include how AI is used in threat detection (e.g., pattern recognition in network traffic), incident response (e.g., automated containment), and the ethical considerations of offensive cyber operations (e.g., potential for autonomous weapons systems). For example, students could analyze how an AI system's benefit of improving threat detection speed introduces the risk of false positives and the risk of unjustly isolating users based on biased training data. Students are not expected to develop or train machine learning models for security or engage in abstract philosophical debates about AI consciousness or moral agency unrelated to cybersecurity applications.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Reflectiveness

Focus Area

Software Development

Cybersecurity

Artificial Intelligence

Physical Computing

Data Science

Game Development

X + CS

Artificial Intelligence

Specialty I

S1-AIN-01: Describe the differences between deterministic and probabilistic algorithms.

Boundary Statement(s)	Students should be able to articulate the fundamental distinction between algorithms that always produce the same output for a given input (deterministic) and algorithms that incorporate randomness, leading to a distribution of possible outputs (stochastic/probabilistic), and be able to identify specific scenarios where one approach is computationally advantageous over the other. For example, students could analyze a deterministic pathfinding algorithm (e.g., Dijkstra's algorithm) and compare its efficiency and outcome to a probabilistic approach for the same problem (e.g., a Monte Carlo simulation or a randomized search heuristic) when applied to large, complex graph data structures. Students are not expected to formally prove the convergence properties or asymptotic runtime of complex randomized algorithms (e.g., Monte Carlo algorithms) or perform advanced calculus-based statistical analysis on the distribution of stochastic algorithm outcomes.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Reflectiveness

Focus Area

- Software Development
- Cybersecurity
- **Artificial Intelligence**
- Physical Computing
- Data Science
- Game Development
- X + CS

S1-AIN-02: Compare data representations and how representation choice constrains applicable algorithms.

Boundary Statement(s)	Students should be able to analyze and articulate the trade-offs between different data representations (e.g., tabular/relational data, graph/network data, image/pixel data, text/sequence data) and explain how the chosen representation dictates the suitable class of algorithms for analysis. Students are not expected to develop custom data representations, convert data into multiple representations, or mathematically prove why certain algorithms cannot work with specific data representations. Students are not required to work with specialized data formats beyond common types (e.g., tabular, graph, image, text).
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Reflectiveness, Resourcefulness

S1-AIN-03: Investigate AI systems to differentiate the types of problems they address.

Boundary Statement(s)	Students should explore and compare different types of AI systems to understand how they work and what problems they solve. They will investigate generative AI (e.g., chatbots or image creators) and discriminative AI (e.g., Teachable Machine or image classifiers) to see how each supports real-world applications (e.g., language translation, image recognition, Sentiment Analysis, recommendations, and game playing. Students are not expected to create AI systems.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems.
Disposition(s)	Resourcefulness, Creativity

Focus Area

- Software Development
- Cybersecurity
- **Artificial Intelligence**
- Physical Computing
- Data Science
- Game Development
- X + CS

S1-AIN-04: Modify AI system inputs to optimize accuracy and reduce bias in outputs.

Boundary Statement(s)	Students should move beyond merely identifying bias in an existing AI model's output and actively mitigate that bias by analyzing and modifying the data used to train or as input to the model. Students could analyze a facial-recognition model's poor performance on dark-skinned faces, identify the lack of diversity in the training dataset as a source of bias, and add images with the underrepresented skin tones to improve accuracy for darker-skinned faces. Students are not expected to understand the mathematical formulas behind fairness metrics or prove why different fairness measures cannot be satisfied at the same time.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Critical Thinking, Reflectiveness

S1-AIN-05: Create an application using supervised learning models.

Boundary Statement(s)	Students should be able to select an appropriate, pre-existing supervised learning model and integrate it into a functional application that uses input data to make a prediction or classification. For example, students could use a pre-trained image classification model (e.g., from Teachable Machine or a Python library) and integrate it into a simple web or mobile application that allows users to upload a photo and receive a prediction about what object is in the image. Students are not expected to train models from scratch using raw data, develop production-ready applications, or work with advanced architectures requiring specialized hardware (e.g., large language models, deep convolutional networks with dozens of layers).
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Creativity, Critical Thinking

Focus Area

- Software Development
- Cybersecurity
- **Artificial Intelligence**
- Physical Computing
- Data Science
- Game Development
- X + CS

S1-AIN-06: Analyze metadata and data pipelines to support transparency in AI model selection.

Boundary Statement(s)	Students should be able to critically analyze the entire data pipeline, from data acquisition through data cleaning, for potential sources of bias, inaccuracy, and lack of transparency. Students should use this analysis to support or advocate for the selection and ethical usage of a specific AI model and its deployment context. For example, students could analyze the metadata of an image dataset (e.g., date of collection, sensor type, image subject) used to train a face recognition model, identifying a lack of demographic diversity, and then clearly articulate how making this information transparent is essential for determining the model's appropriate scope of use (e.g., deciding the model should only be used in non-critical applications or only on populations represented in the data). Students are not expected to design and implement algorithmic fairness metrics (e.g., equal opportunity difference, disparate impact).
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Inclusive Collaboration: 5. Act responsibly in computing collaborations.
Disposition(s)	Critical Thinking, Reflectiveness

S1-AIN-07: Explain neural network structure.

Boundary Statement(s)	Students should be able to articulate the fundamental components of a basic feed-forward neural network (e.g., layers, neurons, weights, biases, and activation functions) and describe the function of each component. Students should be able to conceptually trace how a numerical input value flows through a single neuron, is modified by the weight and bias, and then transformed by the activation function before moving to the next layer. Students are not expected to implement neural networks from scratch, understand the mathematical formulas for backpropagation, or work with advanced architectures (e.g., recurrent neural networks, transformers, convolutional neural networks with many layers).
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Computational Thinking: 7. Develop and use abstractions.
Disposition(s)	Critical Thinking, Resourcefulness

Focus Area

- Software Development
- Cybersecurity
- **Artificial Intelligence**
- Physical Computing
- Data Science
- Game Development
- X + CS

S1-AIN-08: Apply data acquisition, cleaning, and transformation techniques to prepare data for AI analysis.

Boundary Statement(s)	Students should be able to perform hands-on application of data preparation steps, including data acquisition from sources (e.g., CSV files, public APIs, web scraping of small-scale data), identifying and handling missing values (e.g., imputation or removal), and performing basic transformations (e.g., normalization) necessary for training an AI model. For example, students could use a programming library (e.g., Python's Pandas) to load a raw dataset, clean inconsistent entries, convert non-numeric features into a machine-readable format, and split the data into training and testing sets before model application. Students are not expected to work with massive datasets requiring distributed computing infrastructure or implement advanced data preprocessing algorithms from scratch.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Human-Centered Design: 11. Use iterative design processes.
Disposition(s)	Persistence, Critical Thinking, Resourcefulness

S1-AIN-09: Analyze the environmental impacts of widespread AI adoption.

Boundary Statement(s)	Students should be able to quantitatively and qualitatively analyze the environmental impacts of AI, specifically focusing on the energy consumption required for model training and inference and the resulting carbon footprint and electronic waste. For example, students could calculate and compare the estimated carbon dioxide emissions generated by training a large language model versus the energy used by a local, optimized model running on an edge device, and propose a sustainable AI strategy to reduce the overall impact. Students are not expected to conduct complex, full life cycle assessments for computing hardware.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Reflectiveness

Focus Area

- Software Development
- Cybersecurity
- **Artificial Intelligence**
- Physical Computing
- Data Science
- Game Development
- X + CS

S1-AIN-10: Promote safeguards in AI systems that protect human well-being, privacy, and ensure meaningful human involvement in decision-making.

Boundary Statement(s)	Students should be able to analyze and articulate the need for safeguards in AI systems, including multi-agent systems, focusing on practical measures to protect privacy, ensure human well-being, and maintain meaningful human involvement in critical decision-making processes. For example, students could analyze a scenario where an AI is used for medical diagnosis and propose a human-in-the-loop workflow where the AI provides a recommendation, but a qualified human doctor is required to make the final, auditable diagnosis, thereby ensuring meaningful human involvement and protecting patient well-being. Students are not expected to develop formal methods for verifying AI safety.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Human-Centered Design: 12. Design computational technologies that empower and inform users.
Disposition(s)	Critical Thinking, Reflectiveness, Creativity

Focus Area

- Software Development
- Cybersecurity
- **Artificial Intelligence**
- Physical Computing
- Data Science
- Game Development
- X + CS

S1-AIN-11: Analyze the potential biases and limitations of AI systems.

Boundary Statement(s)	Students should be able to employ both qualitative and quantitative methods to identify and analyze systemic issues in AI systems, moving beyond simple error identification to investigating how social, cultural, and historical biases are encoded in the training data, model architecture, and application design. Students should analyze AI systems they created themselves as well as AI systems created by others. For example, students could perform an audited comparison of a facial recognition model created by others versus a text classifier created by the student, using disaggregated metrics to show how performance varies across different demographic groups and linking those performance gaps to specific data collection limitations. Students are not expected to develop novel bias detection algorithms, perform complex causal inference to determine the root cause of systemic inequality, or access proprietary, non-public model weights and data from systems created by industry.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Human-Centered Design: 12. Design computational technologies that empower and inform users.
Disposition(s)	Critical Thinking, Reflectiveness

Focus Area

- Software Development
- Cybersecurity
- **Artificial Intelligence**
- Physical Computing
- Data Science
- Game Development
- X + CS

S1-AIN-12: Explore how AI tools shape user experiences for people with diverse backgrounds and characteristics.

Boundary Statement(s)	Students should be able to explore and articulate how AI tools shape user experiences differently across diverse user characteristics and backgrounds (e.g., socioeconomic status, cultural background, geographic region, language). For example, students could analyze how a voice assistant performs differently for users with different accents or how a recommendation algorithm exposes users to different content based on their demographic characteristics. Students are not expected to design user experience studies for AI tools or implement adaptive personalization systems.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Reflectiveness

S1-AIN-13: Use AI as a tool for software development of an AI agent.

Boundary Statement(s)	Students should be able to strategically integrate AI-powered development tools to manage the lifecycle of AI agent development, including using AI for code generation, debugging, testing, and documentation. For example, students could use an AI coding assistant to help write a game-playing agent, debug errors in the agent's decision-making logic, and generate test cases to verify the agent's behavior. Students are not expected to design or train their own AI-assisted development tools.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Resourcefulness, Persistence

S1-AIN-14: Debate trade-offs between respecting creators' intellectual property rights and using their works to train AI models.

Boundary Statement(s)	Students should be able to analyze and debate the tension between respecting creators' intellectual property rights and the practice of using copyrighted works to train AI models. Students could examine whether AI companies should obtain permission and provide compensation when training models on copyrighted works (e.g., books, articles, artwork, code), considering arguments about fair use, the transformative nature of AI-generated outputs, and whether scraping publicly accessible content from the internet constitutes legitimate use or infringement. Students should consider the economic impact on creators, the benefits of AI advancement to society, and the perspectives of different perspectives from creators, AI developers, and the public. Students are not expected to cite specific legislation or case law.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Reflectiveness

S1-AIN-15: Evaluate AI versus non-AI computational solutions for real-world problems.

Boundary Statement(s)	Students should be able to analyze a real-world problem and systematically compare the trade-offs, feasibility, and appropriateness of AI-based solutions versus non-AI computational solutions. For example, students could evaluate whether to use a pre-trained image classification model (AI) to identify defective parts using assembly line photos or use a traditional image processing algorithm that checks for specific pixel color ranges or geometric shapes (non-AI). Students should consider factors like data availability and interpretability, computational resources, and accuracy needed. Students are not expected to implement AI or non-AI solutions for real-world problems. Students do not need to consider financial factors.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Reflectiveness

Focus Area

- Software Development
- Cybersecurity
- **Artificial Intelligence**
- Physical Computing
- Data Science
- Game Development
- X + CS

S1-AIN-16: Evaluate the ethical implications of AI throughout history and into the future.

Boundary Statement(s)	Students should be able to conduct a critical evaluation of AI ethics, moving beyond surface-level concerns to analyzing how ethical considerations have evolved over time. For example, students could compare the ethical concerns surrounding early conceptions of AI (e.g., Asimov's Three Laws of Robotics) and rule-based AI systems (e.g., concerns about programming errors and logic), with the ethical implications of contemporary AI systems that focus on bias, fairness, and transparency. Students should also evaluate ethical challenges that may emerge as AI capabilities advance, drawing from futuristic depictions of AI in media or their own imaginations. Students are not expected to conduct research on AI development trajectories or apply specific ethical frameworks. Students are not expected to rigorously define consciousness or sentience in discussions of future AI capabilities.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Inclusive Collaboration: 3. Communicate effectively about computing.
Disposition(s)	Critical Thinking, Reflectiveness

Focus Area

- Software Development
- Cybersecurity
- **Artificial Intelligence**
- Physical Computing
- Data Science
- Game Development
- X + CS

Specialty II

S2-AIN-01: Investigate problems that can be addressed using unsupervised learning.

Boundary Statement(s)	Students should be able to investigate and categorize the types of real-world problems that can be effectively addressed using unsupervised learning approaches. Students should demonstrate conceptual understanding of when and why to apply different types of models. Students should be able to identify problem characteristics that make unsupervised learning appropriate (e.g., lack of labeled data, need to discover hidden patterns, exploratory data analysis) and provide examples of problem types (e.g., customer segmentation, anomaly detection, dimensionality reduction, pattern discovery in large datasets). Students are not expected to implement unsupervised learning algorithms or prove their mathematical convergence properties.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Resourcefulness

S2-AIN-02: Create an application using complex supervised learning models.

Boundary Statement(s)	Students should be able to select, implement, and evaluate supervised learning models and integrate them into real-world applications. Students should demonstrate understanding of model evaluation and feature selection for the application. For example, students could create a home price prediction application, select relevant features from housing data (e.g., square footage, location, number of bedrooms), train a supervised learning model, and evaluate its performance by testing predictions against actual prices using appropriate metrics. Students are not expected to design or implement deep learning models.
Pillar(s) and Practice(s)	Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Persistence, Critical Thinking

Focus Area

- Software Development
- Cybersecurity
- **Artificial Intelligence**
- Physical Computing
- Data Science
- Game Development
- X + CS

S2-AIN-03: Optimize AI algorithms to improve performance.

Boundary Statement(s)	Students should be able to optimize AI algorithms to improve their performance on a given task by systematically adjusting parameters and evaluating the impact of changes. For example, students could tune hyperparameters (e.g., learning rate, number of training epochs) or modify model architecture (e.g., number of layers, activation functions) and measure how these adjustments affect accuracy or other performance metrics. Students are not expected to invent novel algorithms or develop new optimization techniques.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact.
Disposition(s)	Critical Thinking, Reflectiveness, Persistence

S2-AIN-04: Modify an AI algorithm to optimize its performance for a given problem.

Boundary Statement(s)	Students should modify and tune the parameters and structure of an accessible AI algorithm and model to optimize performance for a given problem, demonstrating an understanding of how internal algorithm mechanics influence output. This goes beyond simply using pre-set tools to actively altering the algorithm's control flow or hyperparameters. For example, students might modify an algorithm used in discriminative systems (e.g., changing the depth of a decision tree used for an image classifier) or adjust the sampling parameters of a generative system. Students could work across different problem domains, such as tuning weights for sentiment analysis, modifying neural network layers for image recognition, altering minimax depth for game agents, adjusting k-values in recommendation systems, or modifying PID controller parameters for simulated autonomous vehicles. Students are not expected to implement algorithms from scratch, work with production-scale systems, or optimize across all possible hyperparameters using advanced techniques (e.g., Bayesian optimization, neural architecture search).
Pillar(s) and Practice(s)	Computational Thinking: 7. Develop and use abstractions. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Persistence, Resourcefulness

Focus Area

- Software Development
- Cybersecurity
- **Artificial Intelligence**
- Physical Computing
- Data Science
- Game Development
- X + CS

S2-AIN-05: Transform and restructure data for AI analysis.

Boundary Statement(s)	Students should be able to apply data transformation and restructuring techniques that prepare data for AI algorithms, including feature scaling, handling categorical data, aggregating data, and reshaping data. For example, students could take a raw dataset with mixed feature types and apply normalization to numerical columns and one-hot encoding to categorical columns, explaining how these transformations prepare the data for a specific algorithm (e.g., K-Nearest Neighbors classifier). Students are not expected to implement dimensionality reduction techniques.
Pillar(s) and Practice(s)	Inclusive Collaboration: 4. Manage computing projects. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Reflectiveness, Persistence

S2-AIN-06: Evaluate metadata and data pipelines to support transparency in AI model deployment.

Boundary Statement(s)	Students should be able to critically evaluate the entire data pipeline from acquisition through cleaning for potential sources of bias, inaccuracy, and lack of transparency. Students should use this analysis to justify or critique the selection of a specific AI model and its deployment context. For example, students could analyze the metadata of an image dataset used to train a face recognition model (e.g., date of collection, sensor type, demographic coverage), identify a lack of demographic diversity, and articulate how this flaw necessitates either selecting a different model less sensitive to demographic bias or changing the deployment plan (e.g., limiting its use to controlled environments). Students are not expected to design and implement algorithmic fairness metrics.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Inclusive Collaboration: 5. Act responsibly in computing collaborations.
Disposition(s)	Critical Thinking, Reflectiveness, Persistence

Focus Area

- Software Development
- Cybersecurity
- **Artificial Intelligence**
- Physical Computing
- Data Science
- Game Development
- X + CS

S2-AIN-07: Propose a policy change that would promote transparency in AI applications.

Boundary Statement(s)	Students should be able to articulate a well-reasoned, actionable, and specific policy recommendation aimed at increasing transparency across the entire AI lifecycle, from raw data to deployed impact, and justify how that policy balances competing values like innovation and privacy. An example of this is drafting a "Model Card" or "Data Sheet" mandate for high-risk AI applications (e.g., in hiring or criminal justice), requiring public disclosure of the training data sources, the specific model chosen (e.g., random forest), and the results of a disparate impact analysis before deployment. Students are not expected to draft policy documents, cite existing state or federal statutes, or implement the technical infrastructure (e.g., secure data vaults) required to actually make sensitive data accessible.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact.
Disposition(s)	Critical Thinking, Creativity

S2-AIN-08: Analyze how model optimization choices impact a model's performance.

Boundary Statement(s)	Students should be able to explain how different optimization choices (e.g., using gradient descent or adding regularization) and accuracy measures (e.g., F1-score or mean absolute error) affect how an AI model performs and how those choices can impact fairness and transparency. For example, students could discuss how a loan approval model focused only on high recall might lead to more defaults in certain groups, while one focused only on high precision might unfairly reject qualified applicants. Students are not expected to write code or calculate specific metrics.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Computational Thinking: 6. Define computational problems.
Disposition(s)	Reflectiveness, Critical Thinking

Focus Area

- Software Development
- Cybersecurity
- **Artificial Intelligence**
- Physical Computing
- Data Science
- Game Development
- X + CS

S2-AIN-09: Debate aspects of AI regulatory frameworks and legislation across countries.

Boundary Statement(s)	Students should be able to analyze and contrast major AI regulatory frameworks from at least two regions of the world (e.g., the European Union's AI Act versus a US state or federal approach). Students should understand the core philosophical approaches and practical impacts on key areas like data sovereignty and governance. For example, students could compare how the EU's focus on regulating "high-risk" AI systems differs from a less restrictive regulatory environment, and debate the resulting impact on innovation speed versus human protections in areas like facial recognition. Students are not expected to read the text of legislation.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Inclusive Collaboration: 5. Act responsibly in computing collaborations.
Disposition(s)	Critical Thinking, Reflectiveness

S2-AIN-10: Demonstrate professional communication by adapting technical AI results for diverse audiences.

Boundary Statement(s)	Students should be able to move beyond simply stating technical findings to translating model performance metrics (e.g., F1-score, precision, recall) and ethical risks (e.g., disparate impact, bias) into language that is relevant and actionable for specific non-technical audiences. For example, students could prepare a presentation on an AI-driven school admissions model where the executive summary for the school board focuses on cost-savings and scalability, while the community group briefing focuses on fairness metrics and due process for appeals. Students are not expected to create publishable graphics or animated media or participate in real-world communications campaigns.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Creativity, Resourcefulness

Focus Area

- Software Development
- Cybersecurity
- **Artificial Intelligence**
- Physical Computing
- Data Science
- Game Development
- X + CS

S2-AIN-11: Use an AI-assisted development workflow to design, implement, and optimize an AI agent.

Boundary Statement(s)	Students should be able to strategically establish and execute a complete AI agent development workflow that incorporates AI tools at multiple stages, from initial prompt-based design to final performance optimization. Students should view the AI assistant as a continuous collaborator. For example, students could use a large language model to draft a project plan and pseudocode (design), generate initial boilerplate code (implement), use an AI debugger to fix complex errors (implement/optimize), and use an AI code analyzer to suggest refactoring for improved efficiency and readability (optimize). This combined workflow is used to create the AI agent. Students are not expected to build custom AI models for their workflow.
Pillar(s) and Practice(s)	Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Creativity, Resourcefulness

Focus Area

Software Development

Cybersecurity

Artificial Intelligence

Physical Computing

Data Science

Game Development

X + CS

Physical Computing

Specialty I

S1-PHY-01: Construct an electrical circuit to power and control physical computing devices, including creating and interpreting schematic diagrams.

Boundary Statement(s)	Students should be able to create and troubleshoot a simple electrical circuit that powers and controls physical computing devices. They should create and interpret schematic diagrams. For example, students could build a circuit with a microcontroller, resistor, and LED, use a multimeter to check voltage and current, draw a corresponding schematic diagram, and troubleshoot if the LED doesn't light as expected by checking for loose connections or incorrect component orientation. Students are not expected to design circuits with multiple integrated circuits or design and etch custom printed circuit boards.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Persistence, Resourcefulness, Critical Thinking

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- **Physical Computing**
- Data Science
- Game Development
- X + CS

S1-PHY-02: Integrate sensors, actuators, and peripherals with physical computing devices to extend their functionality and gather real-world data for analysis and control.

Boundary Statement(s)	Students should be able to select, connect, and program multiple sensors and actuators with a microcontroller to build interactive systems. For example, students might integrate a light sensor and a motor to create a system that automatically opens and closes a blind based on ambient light level. Students would program the system to read sensor data, process it, and control the actuator accordingly. Students are not expected to integrate a large number of components, work with industrial-grade sensors or actuators that require complex wiring or power considerations, or build systems for mission-critical applications.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Resourcefulness, Persistence, Creativity

S1-PHY-03: Implement software to control physical devices.

Boundary Statement(s)	Students should write and deploy code that directly interacts with physical hardware components (e.g., sensors, actuators, microcontrollers) to create a functional physical system. For example, students might write a program in an environment like the Arduino IDE to read data from a motion sensor and trigger a servo motor to open a small door. Students are not expected to design or fabricate hardware components or work at a low-level machine code level.
Pillar(s) and Practice(s)	Computational Thinking: 8. Create computational artifacts. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Creativity, Resourcefulness

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- **Physical Computing**
- Data Science
- Game Development
- X + CS

S1-PHY-04: Employ IoT devices to collect sensor data and transmit it locally using device-to-device or device-to-gateway communication.

Boundary Statement(s)	Students should be able to configure and program a physical computing device to act as an IoT endpoint. They should collect sensor data and transmit it to another device or a local hub (gateway). Students should focus on data collection and local transmission rather than cloud-based systems. For example, students might use a microcontroller with a Wi-Fi module to collect temperature data from a sensor and send it to a local computer running a server, or directly to another microcontroller. Students are not expected to work with cloud-based IoT platforms or manage complex networking protocols. Students do not need to implement security measures beyond basic passwords or keys.
Pillar(s) and Practice(s)	Computational Thinking: 8. Create computational artifacts. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Resourcefulness, Persistence

S1-PHY-05: Implement IoT communication by connecting physical devices with protocols, applying security practices.

Boundary Statement(s)	Students should be able to configure and program physical devices to communicate with other devices using established protocols (e.g., serial, Bluetooth, Wi-Fi). They should understand fundamental concepts of how data is sent and received over networks. For example, students might build a system where a microcontroller uses a Bluetooth module to send sensor data to a smartphone, or uses Wi-Fi to send data to a local server. They should implement security practices (e.g., using passwords or keys to protect against unauthorized access). Students are not expected to design custom communication protocols or work with advanced networking concepts.
Pillar(s) and Practice(s)	Inclusive Collaboration: 4. Manage computing projects. Computational Thinking: 9. Test and refine computational artifacts.
Disposition(s)	Critical Thinking, Resourcefulness

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- **Physical Computing**
- Data Science
- Game Development
- X + CS

S1-PHY-06: Evaluate social, technical, and sociotechnical impacts of physical computing projects to assess viability.

Boundary Statement(s)	Students should analyze physical computing projects from multiple perspectives to assess feasibility and impact before significant development begins. They should evaluate technical dimensions (e.g., power consumption, sensor accuracy, hardware capabilities), social dimensions (e.g., user accessibility, community benefits, equity considerations), and sociotechnical dimensions (e.g., how the technology changes human behavior, impacts social structures, or affects policies). For example, when evaluating a smart irrigation system design, students might assess whether sensors can accurately measure soil moisture in different conditions, whether the system is accessible to users with varying technical expertise, and how automated watering might change community water-use practices. Students are not expected to develop project budgets, calculate return on investment, or produce compliance or other documentation that would be required in a professional business or engineering setting.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Reflectiveness, Critical Thinking

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- **Physical Computing**
- Data Science
- Game Development
- X + CS

S1-PHY-07: Collaborate using the engineering design process to develop and refine physical computing solutions for diverse users.

Boundary Statement(s)	Students should be able to work effectively in teams to design, build, and test physical computing solutions using a structured engineering design process. Students should demonstrate an awareness of the needs of diverse users and address them in the design process. Students should provide and receive constructive feedback on designs. For example, a group of students might design a smart doorbell for a community center, considering different user needs (e.g., individuals with visual or hearing impairments, those who speak different languages), and iterate on their design based on user feedback. Students are not expected to conduct formal user studies or implement solutions that meet professional accessibility standards.
Pillar(s) and Practice(s)	Inclusive Collaboration: 5. Act responsibly in computing collaborations. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Reflectiveness, Sense of Belonging in CS

S1-PHY-08: Evaluate the security implications of physical computing projects, including data privacy, unauthorized access, and potential vulnerabilities, and implement measures to mitigate risks.

Boundary Statement(s)	Students should be able to identify, analyze, and address security threats and vulnerabilities specific to physical computing devices. They should demonstrate a foundational understanding of security principles and apply them in the context of their projects. Students should understand how physical access to a device can lead to security breaches and how to protect against them. For example, students might create a smart lock and implement measures like strong authentication (e.g., using a keypad and unique PIN), data encryption for communication between the device and a server, and physical safeguards to prevent tampering. Students are not expected to perform penetration testing or conduct systematic security audits.
Pillar(s) and Practice(s)	Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Reflectiveness

Focus Area
Software Development
Cybersecurity
Artificial Intelligence
Physical Computing
Data Science
Game Development
X + CS

S1-PHY-09: Collaborate using project management methodologies to design, develop, and test physical computing projects.

Boundary Statement(s)	Students should be able to apply project management methodologies (e.g., agile frameworks) to collaboratively manage a physical computing project from conception through delivery. They should share responsibilities, resolve conflicts equitably, and communicate progress and outcomes. For example, students might use a kanban board to track tasks (e.g., prototyping, circuit design, programming the microcontroller, user testing) and hold regular stand-up meetings to discuss progress and resolve issues. Students are not expected to use professional project management software or lead projects with multiple teams and external stakeholders.
Pillar(s) and Practice(s)	Inclusive Collaboration: 5. Act responsibly in computing collaborations. Computational Thinking: 9. Test and refine computational artifacts.
Disposition(s)	Persistence, Reflectiveness, Resourcefulness

Specialty II

S2-PHY-01: Develop an electromechanical system using CAD tools for design and testing, considering power requirements, motor types, and control algorithms.

Boundary Statement(s)	Students should be able to design and build a system that integrates mechanical components (e.g., motors, actuators, structural elements) with electrical components. They should use CAD software to model and refine their designs. For example, students could design a robotic arm that picks up and moves objects, using CAD software to design and simulate the arm's mechanical structure before building it. Students are not expected to perform stress analysis, thermal simulations, or other advanced engineering analyses within CAD software. Students do not need to design or machine custom parts.
Pillar(s) and Practice(s)	Computational Thinking: 8. Create computational artifacts. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Creativity, Resourcefulness, Critical Thinking

S2-PHY-02: Evaluate sensor types and implement closed-loop feedback control to maintain a desired outcome.

Boundary Statement(s)	Students should apply quantitative analysis to evaluate sensor characteristics (e.g., accuracy, precision, sensitivity, response time, range) and select appropriate sensors for specific applications. They should design the logic for closed-loop control systems that automatically adjust outputs based on sensor inputs. For example, students might design a heating, ventilation, and air conditioning (HVAC) control system prototype for temperature and humidity regulation, evaluating sensors for their specifications and programming a microcontroller to control fan and heater/cooler actuators to maintain target ranges. Students are not expected to design or fabricate custom circuit boards for sensor signal conditioning (e.g., designing an analog filter circuit from scratch) or perform the calculus necessary to model the full dynamics of a complex, higher-order control system (e.g., using Laplace transforms to analyze system stability).
Pillar(s) and Practice(s)	Inclusive Collaboration: 4. Manage computing projects. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Resourcefulness

S2-PHY-03: Develop an application that extends functionality and user engagement with physical devices.

Boundary Statement(s)	Students should be able to design and program a full-stack application that includes both a front-end user interface and a back-end program that interacts with and controls a physical computing device. For example, students could develop a mobile app that allows a user to remotely control a physical robot, adjust its speed, and receive real-time sensor data from the device. Students are not expected to use low-level programming languages to manage hardware registers or design and fabricate custom circuit boards.
Pillar(s) and Practice(s)	Inclusive Collaboration: 4. Manage computing projects. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Persistence, Resourcefulness

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- **Physical Computing**
- Data Science
- Game Development
- X + CS

S2-PHY-04: Employ IoT devices to collect and transmit data, enabling remote monitoring and control of physical systems.

Boundary Statement(s)	Students should be able to configure a physical computing device to collect sensor data and transmit it to a remote location (e.g., a cloud service, a web server) for visualization and control. Students should use existing IoT services and libraries to connect their devices to the internet. Students should be able to send commands from the remote interface back to the device to control an actuator. For example, they might build a weather station that uses temperature and humidity sensors to collect data and transmit it to a web page viewable from a remote location, with the ability to send commands back to control an LED or fan. Students are not expected to manage cloud infrastructure or build complete data pipelines.
Pillar(s) and Practice(s)	Computational Thinking: 9. Test and refine computational artifacts. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Resourcefulness

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- **Physical Computing**
- Data Science
- Game Development
- X + CS

S2-PHY-05: Develop an embedded network, evaluating trade-offs among network protocols, security measures, and scalability.

Boundary Statement(s)	Students should be able to design, build, and evaluate a multi-device physical computing network that enables communication, data sharing, and remote management. They should focus on fundamental networking principles within a physical computing context. Students should select an appropriate network topology and protocol (e.g., Bluetooth, Wi-Fi, LoRa) based on project requirements (e.g., range, power consumption, data rate). They should implement security practices (e.g., encryption, password protection) and consider how to scale their network to include more devices. For example, students could design a system with multiple environmental sensors (e.g., temperature, humidity) distributed across a room, all communicating with a central hub that logs data and can be accessed remotely via a web server. Students are not expected to configure routing protocols, work with professional networking hardware, or implement custom protocols.
Pillar(s) and Practice(s)	Inclusive Collaboration: 4. Manage computing projects. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Critical Thinking, Reflectiveness

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- **Physical Computing**
- Data Science
- Game Development
- X + CS

S2-PHY-06: Analyze how physical computing technologies could shape social processes, communities, power, and equity in global society.

Boundary Statement(s)	Students should be able to identify and critically evaluate the multifaceted impacts of physical computing technologies on a global scale. Students should go beyond listing pros and cons to examine the nuanced ways these technologies can reinforce or challenge existing social structures, power dynamics, and inequities. For example, students could analyze how the deployment of agricultural drones and sensors in developing nations affects local economies and community labor practices, considering both the potential for increased crop yield and the risk of job displacement for farm workers. Students are not expected to conduct primary sociological research or develop sophisticated economic models to quantify these impacts.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Ethics and Social Responsibility: 2. Respect others' rights when creating computational technologies.
Disposition(s)	Critical Thinking, Reflectiveness

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- **Physical Computing**
- Data Science
- Game Development
- X + CS

S2-PHY-07: Use a source control system to coordinate development in physical computing projects.

Boundary Statement(s)	Students should be able to use a source control system (e.g., Git) to manage and coordinate code development when collaborating on physical computing projects. Students should create branches for different features or components, commit changes with descriptive messages, merge contributions from team members, and resolve basic merge conflicts. Students should recognize and value the contributions of diverse team members, understanding that different perspectives and approaches strengthen the project. Students should understand how source control enables collaborative work by tracking who made what changes and allowing team members to work on different parts of the project simultaneously. For example, a team building a weather station could use Git to manage separate branches for sensor data collection code and display control code, with each team member committing their work and the team merging contributions into a main branch to create the integrated system. Students are not expected to set up and administer repository servers or hosting platforms. Students are not expected to resolve complex merge conflicts involving extensive code changes across many files. Students are not expected to use advanced features like rebasing, cherry-picking, or managing complex branching strategies.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Inclusive Collaboration: 5. Act responsibly in computing collaborations.
Disposition(s)	Reflectiveness, Resourcefulness

Focus Area

Software Development

Cybersecurity

Artificial Intelligence

Physical Computing

Data Science

Game Development

X + CS

Data Science

Specialty I

S1-DSC-01: Apply exploratory data analysis techniques to non-hierarchical data.

Boundary Statement(s)	Students should be able to apply exploratory data analysis (EDA) techniques to non-hierarchical quantitative data (e.g., tabular data stored in CSV files or tab-limited files). Students should use computational tools to calculate summary statistics (e.g., measures of center, spread, and shape), create visualizations, and identify key characteristics of the data (e.g., shape of distributions, outliers, and associations between variables). For example, students could use a programming environment with a data analysis library (e.g., Python's Pandas) to load a CSV file, then use functions to calculate the mean, median, and standard deviation of values for different variables. Students could then generate a histogram to visualize the data distribution for each variable and a scatter plot to examine the relationship between two variables. Students are not expected to analyze or manipulate hierarchical data formats like JSON or XML. Students are also not expected to perform statistical analysis beyond generating summary statistics. Students are not expected to account for nesting or clustering of data when generating summary statistics.
Pillar(s) and Practice(s)	Computational Thinking: 7. Develop and use abstractions. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Creativity

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- **Data Science**
- Game Development
- X + CS

S1-DSC-02: Discuss the metadata when using data collected by others.

Boundary Statement(s)	Students should be able to identify, interpret, and discuss the metadata associated with a dataset they did not collect themselves. This involves examining and explaining what the metadata tells them about the data's origin, collection methods, variables, and potential limitations before beginning an analysis. For example, when given a dataset, students should analyze a readme file or data dictionary to understand where the data came from, when it was last updated, what each column or variable represents, and what units of measurement were used. They should then use this information to inform their analysis, discuss potential sources of bias in the data and other factors that will influence data interpretation. Students are not expected to work with datasets that lack any form of metadata. Students are also not expected to perform quality assurance on the metadata and data (e.g., to compare the metadata and data to ensure they align).
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 2. Respect others' rights when creating computational technologies. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Resourcefulness

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- **Data Science**
- Game Development
- X + CS

S1-DSC-03: Write code to manipulate and transform data to prepare for analysis.

Boundary Statement(s)	Students should be able to write code to perform common data manipulation and transformation tasks (e.g., filtering, grouping, summarizing, calculating new variables, restructuring, merging datasets) to prepare a dataset for a specific analysis. For example, students could filter a dataset to include only rows that meet specific criteria (e.g., sales from a particular region), group data by a certain category (e.g., sales by product type), and then summarize those groups (e.g., calculate the total sales for each product type). They should also be able to calculate new variables (e.g., product prices converted to a different currency), restructure a dataset (e.g., from a long format to a wide format), and merge two or more datasets based on a common key. Students should work with datasets large enough to require processing with computational tools. They should use existing libraries (e.g., Python's pandas, R's dplyr) designed for these purposes. Students are not expected to build their own tools for data manipulation. Students should not work with datasets so large that standard computational tools run slowly or run out of memory. Students are not expected to optimize code for performance with very large datasets.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Persistence, Resourcefulness

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- **Data Science**
- Game Development
- X + CS

S1-DSC-04: Apply an appropriate analytic or visualization technique for categorical and quantitative data.

Boundary Statement(s)	Students should be able to select and apply appropriate computational methods for analyzing and visualizing categorical data (e.g., bar charts, mosaic plots, frequency tables) and quantitative data (e.g., histograms, scatter plots, box plots, summary statistics). Students should understand how the data type influences the choice of analytic and visualization techniques. For example, students could use computational tools to create a bar chart showing the distribution of car types in a dataset and a scatter plot examining the relationship between engine size and fuel efficiency. Students are not expected to perform inferential statistical tests (e.g., chi-squared tests, t-tests, ANOVA) or statistical modeling techniques that require understanding of probability distributions and statistical significance (e.g., linear regression, logistic regression). Students do not need to work with multidimensional data requiring techniques like principal component analysis or advanced machine learning models.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Reflectiveness, Resourcefulness

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- **Data Science**
- Game Development
- X + CS

S1-DSC-05: Examine missing data and its impact on data analysis.

Boundary Statement(s)	Students should be able to examine datasets with missing values to understand the nature and extent of the missing data (e.g., identifying which variables have missing values, how much data is missing, whether there are patterns in what is missing). Students should evaluate how missing data affects different analytical approaches, recognizing that some methods require complete data while others can handle missing values. For example, students might explore a housing dataset where income data is partially missing and observe that some visualization techniques (e.g., scatter plots) automatically exclude incomplete cases while summary statistics can be calculated on available data. Students should understand that decisions about handling missing data (e.g., removing incomplete cases, using mean imputation) can affect results and potentially introduce bias. Students should develop awareness of missing data as an important data quality issue that requires thoughtful consideration. Students are not expected to formally classify missing data mechanisms (missing completely at random, missing at random, missing not at random) using statistical tests, though discussing these concepts is appropriate. Students are not expected to implement advanced imputation techniques (e.g., k-nearest neighbors imputation, multiple imputation by chained equations) or to perform formal statistical comparisons of model performance with different missing data treatments.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- **Data Science**
- Game Development
- X + CS

S1-DSC-06: Analyze structured categorical and/or quantitative datasets, using computational tools and libraries.

Boundary Statement(s)	Students should be able to use computational tools and libraries to perform exploratory data analysis on structured datasets. They should write and execute code to clean, transform, and aggregate data, and use visualization libraries to create charts and graphs that reveal trends, patterns, and relationships within the data. For example, students could use a library (e.g., Pandas in Python) to analyze a dataset of global climate information, computing and visualizing average temperature changes over a decade for different continents, or grouping temperature data by month to calculate average monthly temperatures. Students are not expected to work with datasets so large that standard tools run slowly or run out of memory, use distributed computing frameworks (e.g., Spark or Hadoop), or build data analysis tools from scratch.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Critical Thinking, Persistence

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- **Data Science**
- Game Development
- X + CS

S1-DSC-07: Interpret the results of data analyses to explain patterns, anomalies, and trends, and connect them back to the original problem or research question.

Boundary Statement(s)	Students should be able to examine existing data analyses (e.g., visualizations, summary statistics, published findings) and interpret what the results mean in relation to the original research question or problem. Students should identify key patterns, trends, and anomalies in the data and explain how these findings address the research question. Students should discuss limitations of the analysis and consider alternative explanations for the observed patterns. For example, students might examine a visualization showing declining bee populations over time, identify the downward trend, connect this pattern to a research question about pollinator health, and discuss possible explanations (e.g., pesticide use, habitat loss) while noting limitations such as the time period covered or other factors not measured in the dataset. Students are not expected to perform the statistical calculations or create the visualizations themselves for this standard. Students are not expected to conduct formal hypothesis testing, calculate inferential statistics, or replicate the original analysis to validate their interpretations.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Computational Thinking: 7. Develop and use abstractions.
Disposition(s)	Critical Thinking, Reflectiveness

Focus Area

Software Development

Cybersecurity

Artificial Intelligence

Physical Computing

Data Science

Game Development

X + CS

S1-DSC-08: Explain how dataset size affects model stability and performance.

Boundary Statement(s)	Students should be able to explore how adding or removing data points (rows) affects model behavior when the model is re-run. Students should recognize that small datasets are more sensitive to changes than large datasets - adding or removing even a small number of rows from a small dataset can cause substantial changes in model outputs, while adding or removing the same number of rows from a large dataset has minimal impact. Students should understand that larger datasets generally produce more stable and reliable models. Students should observe and describe the practical effects of dataset size on model behavior. For example, students might start with a model trained on 20 house sales, observe how predictions can change dramatically when adding or removing 10 sales, then compare this to adding or removing 10 sales from a dataset of 1,000 sales where predictions barely change. Students are not expected to calculate formal statistical measures of model stability. Students are not expected to understand the mathematical relationship between sample size and variance.
Pillar(s) and Practice(s)	Computational Thinking: 7. Develop and use abstractions. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Reflectiveness

S1-DSC-09: Assess the appropriateness of predictive models for the specific problem being addressed.

Boundary Statement(s)	Students should be able to evaluate the suitability of a predictive model for a given problem at a conceptual level. Students should consider the type of data, the nature of the outcome (e.g., categorical vs. continuous), and the potential real-world impact of the model's performance. For example, students could be given a problem to predict whether a customer will click on an ad. They should explain why a logistic regression model is more appropriate than a linear regression model for this specific problem, which requires predicting a categorical outcome (yes/no) rather than a continuous numerical value. Students are not expected to build or assess deep learning models. Students are not required to statistically evaluate different models to determine which model has the best fit.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Reflectiveness

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- **Data Science**
- Game Development
- X + CS

S1-DSC-10: Create a visualization that clearly communicates key findings to diverse audiences, selecting appropriate chart types and formatting.

Boundary Statement(s)	Students should be able to create data visualizations that effectively communicate specific findings to different audiences. Students should demonstrate understanding of how different visual elements impact viewers' comprehension and interpretation of the data and choose design elements intentionally. Students should select appropriate chart types (e.g., line graphs for trends over time, bar charts for comparing categories, or scatter plots for relationships) and apply formatting to enhance clarity and readability (e.g., using clear titles and labels and choosing meaningful colors). For example, students could create a bar chart to visualize the average height of students in each grade level for a general audience. Students could intentionally use color to visually distinguish between elementary school students, middle school students, and high school students. For a more research-oriented or data-savvy audience, students could then add error bars to the bar chart or create a density plot to highlight the distribution of height in each grade level and how much the distributions overlap across grade levels. Students should focus on designing intentionally and developing a basic understanding of how different visual elements impact a viewer's comprehension and interpretation of the data. Students are not expected to create complex data visualizations or interactive dashboards. Students are not expected to user-test and collect feedback on their visualizations.
Pillar(s) and Practice(s)	Computational Thinking: 8. Create computational artifacts. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Reflectiveness

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- **Data Science**
- Game Development
- X + CS

S1-DSC-11: Demonstrate how common graphical conventions are used in data visualizations and how breaking these conventions can lead to misleading interpretations.

Boundary Statement(s)	Students should be able to demonstrate an understanding of established graphical conventions in data visualization and explain how intentionally or unintentionally deviating from these conventions can misrepresent data and mislead an audience. This includes conventions such as a y-axis that starts with smaller values at the bottom and higher values at the top of the chart, how specific colors are associated with specific meanings (e.g., red is associated with warnings in American culture), and the appropriate use of chart types for different data types. For example, students could be given a visualization that inverts the y-axis so smaller values are at the top and larger values at the bottom. They would explain how the convention aligns with our natural sense of larger vs. smaller and how inverting the y-axis makes the data more difficult to interpret correctly. Students could then create a corrected version of the visualization where the y-axis is oriented as expected. Students are not expected to analyze or correct 3D or interactive data visualizations, data visualizations that use logarithmic scales, or other more advanced types of data visualizations (e.g., network graphs, radar graphs, sankey diagrams).
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Critical Thinking, Reflectiveness

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- **Data Science**
- Game Development
- X + CS

S1-DSC-12: Apply ethical principles to data collection, analysis, and communication to promote privacy, transparency, and accountability.

Boundary Statement(s)	Students should be able to apply ethical principles in the context of data science projects, going beyond simple awareness of privacy and bias to promote transparency and accountability. For example, in a project involving a social media dataset, students could develop a plan for data anonymization to protect user privacy and draft a public-facing statement explaining what data was used, how it was analyzed, and what conclusions were drawn, thereby promoting transparency and accountability. Students are not expected to read or interpret data privacy laws or use statistical techniques to quantify bias or representativeness.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 2. Respect others' rights when creating computational technologies. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking

S1-DSC-13: Assess how data collection and use may impact marginalized and underrepresented groups.

Boundary Statement(s)	Students should be able to analyze and articulate the potential impacts of data collection and use on different communities, particularly those who are marginalized or underrepresented. This goes beyond a general discussion of bias to a specific examination of how data choices can exacerbate or mitigate existing social inequalities. For example, students could analyze a facial recognition dataset to assess its representation of different skin tones and genders and propose ways to collect a more inclusive and equitable dataset. Students are not expected to use statistical tests to quantify demographic representation or develop bias mitigation algorithms.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Ethics and Social Responsibility: 2. Respect others' rights when creating computational technologies.
Disposition(s)	Critical Thinking, Reflectiveness

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- **Data Science**
- Game Development
- X + CS

S1-DSC-14: Communicate results of data analyses in formats appropriate for audiences with different backgrounds and perspectives.

Boundary Statement(s)	Students should be able to translate data analysis results into clear narratives for different audiences, using appropriate communication formats. This goes beyond simple charts and graphs. Students should tell the story of their data by articulating the purpose, methods, findings, and ethical considerations of their analysis. For example, students could create a presentation for a public audience explaining findings from a community health data analysis and write a technical report for other data scientists detailing the study design, data collection methods, and statistical models used in the analysis. Students are not expected to create data visualizations using professional tools (e.g., Tableau, Power BI) without structured guidance. Students are not expected to be polished public speakers, but they should be able to convey their findings clearly and confidently.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Inclusive Collaboration: 3. Communicate effectively about computing.
Disposition(s)	Critical Thinking, Resourcefulness

Focus Area

Software Development

Cybersecurity

Artificial Intelligence

Physical Computing

Data Science

Game Development

X + CS

Specialty II

S2-DSC-01: Select appropriate data to collect for a data science project based on available tools, skills, and project goals.

Boundary Statement(s)	Students should be able to analyze a data science project's objectives and make informed decisions about what kind of data to collect. This involves evaluating project goals to determine what questions need to be answered. Students should consider available tools (e.g., Python libraries for web scraping, APIs, or sensor hardware) and their own skills to choose a feasible data collection strategy. For example, if a project's goal is to analyze local air quality, students should consider whether to use a public API, purchase and use a physical sensor, or use an existing government dataset, weighing the pros and cons of each method against their resources and abilities and the nature of the research question. Students should focus on the decision-making process and justifying their choice of data source and collection method. Students are not expected to build data collection tools from scratch (e.g., custom web scrapers). Students should not focus on technical implementation. Students are not required to work with large data sets that require memory or processing power beyond what is available on their personal devices. Students are not required to work with data from multiple, disparate sources.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Resourcefulness, Reflectiveness

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- **Data Science**
- Game Development
- X + CS

S2-DSC-02: Apply exploratory data analysis techniques to hierarchical structured data sources.

Boundary Statement(s)	Students should be able to apply exploratory data analysis techniques to hierarchical structured data sources, moving beyond processing tabular data. This involves navigating nested data, extracting relevant information from different levels of the hierarchy, and applying summary statistics and visualizations to this extracted data. For example, students could be given a JSON file containing nested information about an API response, such as a list of songs, each with a nested object for the artist, including their name, genre, and a list of other albums. Students could parse this data to calculate the average length of songs for a specific genre or to create a bar chart showing the number of songs per artist. Students are not expected to handle deeply nested or graph-structured data.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Computational Thinking: 7. Develop and use abstractions.
Disposition(s)	Critical Thinking, Resourcefulness

S2-DSC-03: Document the origin, structure, and preparation of datasets to support clarity and reproducibility.

Boundary Statement(s)	Students should be able to create clear and detailed documentation for a dataset they have prepared for analysis. This goes beyond simply using metadata to creating a comprehensive record of the data's entire lifecycle, from its source to its final, prepared state to ensure that their work is reproducible. Students should document the data's origin, data collection methods, and all the steps taken to clean, transform, or organize it for analysis. For example, students could write a report detailing that their dataset came from a specific government website's API, was collected on a certain date, and was then cleaned by removing missing rows and normalizing a date column into a standard format, providing the code snippets used for each step. Students are not expected to use version control systems (e.g., Git) to track every change.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Persistence, Reflectiveness

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- **Data Science**
- Game Development
- X + CS

S2-DSC-04: Write code to collect and integrate data from multiple sources.

Boundary Statement(s)	Students should be able to write code to collect data from diverse sources (e.g., APIs, web scraping, databases) and integrate datasets that were not originally designed to be combined. Students should identify or create common keys to enable merging (e.g., matching records based on ticker symbols, dates, or other shared attributes) and verify that the merge was performed correctly by checking for expected row counts and examining merged records. Students should then transform the combined data using data moves (e.g., filtering, grouping, summarizing, calculating new variables, restructuring) to answer more complex and original questions that cannot be answered with a single, pre-cleaned dataset. For example, students might collect real-time stock market data from an API, integrate it with a historical dataset from a file, create a common key based on stock ticker and date, verify the merge, and then group by industry sector and calculate summary statistics to analyze trends and make predictions. Students are not expected to perform complex data matching algorithms (e.g., fuzzy matching for approximate string matches, probabilistic record linkage). Students are not expected to handle unstructured data (e.g., images, audio files, raw text) that requires parsing or machine learning techniques. Students do not need to write their own API clients or web scrapers from scratch - they may use existing libraries for data collection.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Persistence, Resourcefulness

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- **Data Science**
- Game Development
- X + CS

S2-DSC-05: Apply models to explain relationships, make predictions, and evaluate the influence of different variables.

Boundary Statement(s)	Students should be able to create and apply simple computational models (e.g., linear regression, decision trees, or basic simulations) to make predictions and explain relationships within a dataset. Students should be able to evaluate how different input variables influence the model's output. For example, students could build a simple linear regression model to predict a runners' race times based on factors like training hours and diet and use the model's output to explain which factors have the most significant influence on performance. Students could also use a decision tree to classify an email as spam or not spam and explain the sequence of decisions the model makes to arrive at its conclusion. Students are not expected to implement these model algorithms from scratch or to understand the underlying mathematical theory in depth. Students are not expected to work with advanced machine learning techniques (e.g., neural networks, ensemble methods, deep learning).
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Critical Thinking, Persistence

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- **Data Science**
- Game Development
- X + CS

S2-DSC-06: Apply strategies for handling missing data while considering the effects on analysis results.

Boundary Statement(s)	Students should be able to identify missing data in a dataset (e.g., empty cells, null values, placeholder values like "NA"), select and implement an appropriate strategy for handling it using code. Common strategies include removing rows or columns with missing data, imputing missing values (e.g., using mean, median, or a specified value), or retaining missing values and excluding them from specific calculations. Students should consider how their chosen approach affects analysis results when selecting and implementing a strategy. Students should make informed decisions about handling missing data based on observed effects on their analysis. For example, students might observe that removing all rows with any missing values dramatically reduces dataset size and then try mean imputation instead, comparing how the two approaches affect summary statistics or visualizations to inform their choice of which strategy to use. Students are not expected to implement advanced imputation algorithms (e.g., k-nearest neighbors imputation, multiple imputation). Students are not expected to use statistical tests to assess the quality of different missing data strategies.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Reflectiveness, Persistence

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- **Data Science**
- Game Development
- X + CS

S2-DSC-07: Analyze unstructured, mixed data type, or high-dimensional datasets using computational tools and libraries.

Boundary Statement(s)	Students should be able to analyze and derive insights from unstructured data (e.g., text documents) or high-dimensional data where the number of features is very large (e.g., image data where each pixel is a feature, or sensor data that is sampled many times per second). Students should use appropriate computational libraries and techniques to process, clean, and analyze these complex data formats. For example, students could be given a collection of text documents and use a library to perform a simple analysis (e.g., word frequency counting) to find key themes. For a high-dimensional dataset, students could apply a dimensionality reduction technique to visualize or analyze the data more effectively. Students should demonstrate fundamental skills for handling data sets that are common in data science but not easily managed by traditional spreadsheet software. Students are not expected to build models for complex tasks (e.g., natural language processing, image recognition, or complex signal processing).
Pillar(s) and Practice(s)	Computational Thinking: 7. Develop and use abstractions. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Resourcefulness

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- **Data Science**
- Game Development
- X + CS

S2-DSC-08: Evaluate the performance of models using established metrics.

Boundary Statement(s)	Students should be able to use computational tools and established metrics to evaluate and compare the performance of different models. Students should assess a model's fit by using metrics such as accuracy or precision for classification models, or mean squared error or R-squared for regression models. Students should also assess computational efficiency by measuring how long it takes to train or execute a model. Students should use existing library functions to generate evaluation metrics and use those results to make informed judgments about model quality, considering trade-offs between speed and accuracy. For example, after building a model to predict house prices, students could compare its performance to another model with more predictor variables by calculating the R-squared value for each. Students could then explain which model is a better fit for the data and discuss if the inclusion of more variables is worth the extra resources needed to collect additional data. Students are not expected to implement evaluation metrics from scratch or to understand the mathematical derivations of metrics. Students are not expected to perform theoretical algorithmic analysis (e.g., Big O notation) to assess computational complexity.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Critical Thinking, Persistence

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- **Data Science**
- Game Development
- X + CS

S2-DSC-09: Explain the trade-off between interpretability, accuracy, and generalizability as it relates to model complexity.

Boundary Statement(s)	Students should be able to explain how choices in model complexity create trade-offs between interpretability, accuracy, and generalizability. A more complex model may be more accurate on training data but is often more difficult to understand and interpret. Additionally, complex models risk overfitting—learning the training data too well, including noise and outliers—which reduces their ability to generalize to new, unseen data. Students should be able to articulate why a simpler model that is less accurate but highly interpretable may be more suitable for certain applications. For example, students could build two models to predict whether a student will pass or fail a course: a simple logistic regression model that uses only study hours and attendance as features, and a complex neural network that uses dozens of features. The logistic regression might achieve 80% accuracy and provide a clear, interpretable rule (e.g., "students who study more than 10 hours per week and attend at least 80% of classes are likely to pass"), while the neural network might achieve 85% accuracy but function as a "black box" where teachers cannot understand why specific predictions are made. For advising students, the interpretable model may be more valuable despite lower accuracy. Students should focus on conceptual understanding of these trade-offs and their real-world implications. Students are not expected to build or train machine learning models from scratch. Students are not expected to understand or derive the mathematical theory behind why these trade-offs exist.
Pillar(s) and Practice(s)	Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Critical Thinking

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- **Data Science**
- Game Development
- X + CS

S2-DSC-10: Analyze how adding or removing variables affects model behavior and performance.

Boundary Statement(s)	Students should be able to systematically add, remove, or modify variables (features) in a dataset and analyze how these changes impact model predictions or performance when the model is re-run. Students should understand that adding correlated variables (e.g., number of bedrooms and square footage) can affect how the model uses each variable and may change which variables appear most important. Students should explore how different combinations of variables affect model accuracy or error. For example, students might start with a model predicting house prices based on square footage alone, then add variables like years since remodel or median neighborhood price to see how this affects the model's accuracy or changes the relative importance of square footage. Students are not expected to calculate variance inflation factors or other formal measures of multicollinearity. Students do not need to understand the underlying mathematical reasons for how correlated variables affect model coefficients.
Pillar(s) and Practice(s)	Computational Thinking: 7. Develop and use abstractions. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Persistence

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- **Data Science**
- Game Development
- X + CS

S2-DSC-11: Create a visualization that accurately represents data and avoid misleading design choices.

Boundary Statement(s)	Students should be able to create data visualizations while making deliberate design choices to avoid common pitfalls that can mislead viewers. Students should use appropriate axis ranges, maintain consistent scales, select chart types that match the data and message, and avoid visual distortions. Students should refine their visualizations by checking that the visual representation accurately reflects the underlying data. Students should also consider accessibility (e.g., using colorblind-friendly palettes, providing text alternatives). For example, students creating bar charts to visualize SAT scores (200-800 range) and ACT scores (1-36 range) should recognize that using the same axis scale (e.g., 0-800) for both tests would compress ACT score variation and make differences appear negligible, while using appropriate scales for each test (200-800 for SAT, 1-36 for ACT) accurately represents the variation within each test. Students are not expected to conduct formal user testing or collect empirical feedback on visualization effectiveness. Students are not expected to create interactive dashboards or use advanced visualization libraries. Students are not expected to master all accessibility standards but should demonstrate awareness of basic considerations.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Reflectiveness

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- **Data Science**
- Game Development
- X + CS

S2-DSC-12: Critique a data visualization for misleading elements and their role in spreading misinformation.

Boundary Statement(s)	Students should be able to analyze a data visualization and accompanying narrative from public sources (e.g., news articles, social media posts, advertisements, political campaigns) to identify misleading or deceptive elements. Students should explain how specific design choices (e.g., truncated or flipped axes, cherry-picked data ranges, counterintuitive color palettes) can distort viewers' understanding. Students should discuss how misleading visualizations contribute to spreading misinformation or disinformation. Students should identify misleading elements and understand their potential impact. For example, students might analyze a social media post showing a dramatic-looking trend line that uses a truncated y-axis and a selective time range, explain how these choices exaggerate the trend, identify what narrative the visualization promotes, and discuss potential real-world consequences of its spread. Students are not expected to verify data authenticity or trace the original source of visualizations. Students are not expected to determine the intent behind misleading visualizations (whether deliberate deception or honest mistake).
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Reflectiveness

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- **Data Science**
- Game Development
- X + CS

S2-DSC-13: Apply ethical, legal, and social considerations when working with large-scale datasets, predictive models, and emerging technologies.

Boundary Statement(s)	Students should be able to analyze the ethical, legal, and social implications of data science projects, particularly those that involve large datasets and predictive models, and apply principles of fairness, accountability, and transparency in their own work. This goes beyond identifying bias to actively considering how decisions at each stage—from data collection through model deployment—can cause harm and taking steps to prevent it. For example, when building a predictive model, students should identify potential sources of bias in the data, discuss the societal consequences of the model's predictions, and propose methods for ensuring that the model's decisions are fair and transparent. Students are not expected to read or interpret specific data privacy laws and regulations (e.g., CCPA, GDPR). Students are not expected to use mathematical techniques to measure fairness or to conduct statistical tests to validate model performance.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Ethics and Social Responsibility: 2. Respect others' rights when creating computational technologies.
Disposition(s)	Critical Thinking, Reflectiveness

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- **Data Science**
- Game Development
- X + CS

S2-DSC-14: Communicate technical results for diverse stakeholders in written reports, presentations, and interpersonal communication.

Boundary Statement(s)	Students should be able to translate data analyses into clear, actionable insights tailored for different audiences with varying levels of technical expertise. This goes beyond simply presenting findings to strategically crafting a message that resonates with the audience's interests and needs. For example, students could create a presentation for a city council, emphasizing the policy implications of a traffic data analysis, while simultaneously preparing a detailed technical report and code library so other scientists can reproduce the results. Students are not expected to be polished public speakers or to create publication-ready infographics with custom illustrations.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Reflectiveness, Resourcefulness

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- **Data Science**
- Game Development
- X + CS

S2-DSC-15: Evaluate protective measures in data collection, usage, and governance for privacy, security, and fairness.

Boundary Statement(s)	Students should be able to analyze and evaluate protective measures used in data collection, usage, and governance, including privacy safeguards, security protocols, consent mechanisms, and bias prevention strategies. This goes beyond simply identifying these measures to assessing their effectiveness in specific contexts and understanding their interconnectedness. For example, students could evaluate a mobile application's data practices by analyzing its privacy policy and requested permissions, identifying potential privacy risks (e.g., unnecessary location tracking) and proposing alternative approaches that would better protect user privacy and prevent bias. Students are not expected to implement cryptographic security protocols (e.g., encryption algorithms) or to develop and deploy consent management systems (e.g., software that tracks user consent choices, enforces data usage permissions, and generates compliance audit trails).
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Ethics and Social Responsibility: 2. Respect others' rights when creating computational technologies.
Disposition(s)	Critical Thinking, Resourcefulness

Focus Area

Software Development

Cybersecurity

Artificial Intelligence

Physical Computing

Data Science

Game Development

X + CS

Game Development

Specialty I

S1-GMD-01: Analyze the fundamental components of games, including players, rules, actions, and outcomes.

Boundary Statement(s)	Students should identify and document the core components of simple games, including player roles, available actions, game rules that govern play, win/loss conditions, and how game state changes based on player actions. Students should represent these components using appropriate models (e.g., state diagrams, flowcharts, rule tables). For example, students analyzing a simple card game could document the initial setup, legal moves on each turn, how card plays affect game state, and the conditions that end the game. Students are not expected to analyze complex games with extensive rule sets, probabilistic elements, or real-time mechanics that would require sophisticated implementation techniques.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Reflectiveness, Curiosity

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- Data Science
- **Game Development**
- X + CS

S1-GMD-02: Enhance existing rule-based logic to control Non-Playable Characters (NPC).

Boundary Statement(s)	Students should be able to read and understand rule-based code controlling Non-Playable Character (NPCs). They should identify strengths and limitations of the controlling logic and suggest and implement improvements in the algorithm. For example, students might analyze a simple rule-based NPC, such as a guard that patrols, chases, or attacks based on player distance, and explain how each conditional statement controls its behavior. They would then identify gaps, like the NPC never retreating or reacting to low health, and modify the algorithm to add more realistic or strategic decision-making. Students are not expected to implement AI-based systems nor formally evaluate algorithmic limitations.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Computational Thinking: 7. Develop and use abstractions.
Disposition(s)	Critical Thinking, Resourcefulness

S1-GMD-03: Create a storyboard to plan and communicate game narratives and interactive experiences.

Boundary Statement(s)	Students should be able to create storyboards that visually outline the narrative flow and interactive elements of a game. Their storyboards should illustrate key scenes, player choices, decision points, and how the game responds to those choices. Storyboards should use simple sketches and annotations to clarify on-screen actions, intended player interactions, and the emotions or reactions the designer wants to evoke. The emphasis is on using storyboards as a planning and communication tool. Basic, low-fidelity sketches that clearly convey the game's structure and interactivity are sufficient. Students are not expected to create polished or artistically detailed drawings or depict every possible branching path.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems.
Disposition(s)	Creativity

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- Data Science
- **Game Development**
- X + CS

S1-GMD-04: Develop an interactive experience to support gameplay.

Boundary Statement(s)	Students should be able to design a cohesive game experience. The design should leverage the unique strengths of various input devices to create an enriched or unique interactive experience, not simply map controls one-to-one. Students should use existing or commonly available hardware in their designs. For example, students could design a game where a player uses a joystick for character movement while also using a touchscreen or keyboard to manage a separate inventory system, creating a multi-modal interface. Students are not expected to create new input devices or build drivers to support their designs.
Pillar(s) and Practice(s)	Computational Thinking: 8. Create computational artifacts. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Creativity, Critical Thinking

S1-GMD-05: Conduct a basic usability test on a game by identifying key user flows, collecting observable data, and translating that data into actionable design revisions.

Boundary Statement(s)	Students should be able to conduct a basic usability test on a game by identifying key user flows, observing how users interact with the experience, and recording meaningful data about where users succeed or struggle. They should use this evidence to recommend clear, actionable design revisions. As part of the testing process, students should consider whether any aspects of the game create barriers for certain users or unintentionally introduce bias or exclusion. For example, students might examine whether an onboarding tutorial is confusing, whether controls are difficult for users with different abilities, or whether characters or narrative elements reinforce stereotypes. Students should use feedback from a small group of peers with varied perspectives or abilities to propose design improvements that enhance usability and inclusivity. Students are not expected to run large-scale user studies, use professional usability-testing tools, or conduct formal accessibility or ethical audits.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Reflectiveness, Critical Thinking

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- Data Science
- **Game Development**
- X + CS

S1-GMD-06: Create a user-friendly interface for a game, considering accessibility, usability, and aesthetics.

Boundary Statement(s)	Students should be able to design and prototype user interfaces (UIs) that are not only functional but also intuitive, aesthetically pleasing, and accessible to a diverse range of users. They should understand how design choices for interactive media, such as button placement, color schemes, and font readability, impact the user experience. Students should go beyond basic user interaction and demonstrate a more sophisticated application of design principles. For example, students could design and prototype a UI for a simple mobile game that includes a colorblind-friendly mode, adjustable font sizes for readability, and clear, universally recognized icons. Students are not expected to implement complex, dynamic UIs that require advanced knowledge of front-end frameworks or backend data management.
Pillar(s) and Practice(s)	Computational Thinking: 8. Create computational artifacts. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Creativity, Reflectiveness

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- Data Science
- **Game Development**
- X + CS

S1-GMD-07: Describe the key architectural features of modern GPUs and their implications for game development.

Boundary Statement(s)	Students should be able to describe at least two key architectural features of modern GPUs and explain how each one directly impacts game development. Students should demonstrate a conceptual understanding of the main architectural components and how they influence game design and performance that goes beyond defining a GPU. For example, students could describe the large number of specialized cores or stream processors that allow a GPU to handle thousands of calculations simultaneously, enabling complex lighting and particle effects in games. They could also explain the importance of dedicated, high-bandwidth memory (VRAM) in GPU architecture for efficiently loading and rendering high-resolution textures. Students are not expected to understand the low-level microarchitecture of specific GPU manufacturers or perform hardware-level analysis.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Resourcefulness

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- Data Science
- **Game Development**
- X + CS

S1-GMD-08: Analyze the ethical implications of copyright and intellectual property in game development.

Boundary Statement(s)	Students should be able to analyze the ethical and social implications of copyright and intellectual property in games, applying their understanding to common scenarios (e.g., modding, fan art, using third-party assets). They should be able to articulate the balance between a creator's right to their work and a community's desire to build upon that work, specifically examining how fair use doctrines and Creative Commons licensing models provide legal and ethical frameworks for this interaction. For example, students might analyze a case where a game company issues a cease and desist order to a fan-made game and discuss the ethical arguments for and against the company's action, taking into account principles of fair use and the potential for Creative Commons alternatives. Students are not expected to review laws or contracts.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 2. Respect others' rights when creating computational technologies. Inclusive Collaboration: 3. Communicate effectively about computing.
Disposition(s)	Reflectiveness, Resourcefulness

S1-GMD-09: Collaborate effectively within diverse teams to plan, develop, and iterate on game development projects.

Boundary Statement(s)	Students should be able to collaborate effectively within a team to plan, develop, and iterate on a game development project by defining roles, setting clear objectives and milestones, and establishing a version control system (e.g., Git) for managing project assets and code. They should collaboratively implement core game mechanics, integrate assets, and use an agile workflow (e.g., short sprints and regular check-ins) to maintain forward progress. Team members should collect feedback from peers or target users, identify bugs, and refine the game design and code based on these critiques. Students are not expected to implement project management methodologies typically used in professional studios (e.g., advanced Scrum or Kanban techniques) or design and create all assets from scratch, provided they appropriately credit third-party or open-source assets used in the project.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Inclusive Collaboration: 5. Act responsibly in computing collaborations.
Disposition(s)	Sense of Belonging in CS, Persistence

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- Data Science
- **Game Development**
- X + CS

Specialty II

S2-GMD-01: Apply responsible design principles to the design and development of engaging and meaningful game experiences.

Boundary Statement(s)	Students should be able to design and develop game experiences by applying responsible design principles (e.g., ethical considerations, inclusivity and accessibility, sustainability, transparency and trust) that specifically address player motivation, emotion, and behavior, considering a diverse range of human identities. Students should develop game experiences that emphasize inclusive design decisions, iterative development, and user empathy. For example, students could design a simulation tailored for a specific audience, or develop an interactive product with adaptive controls or alternative input methods that enhance accessibility and engagement. Students are not expected to conduct research studies to measure user motivation, emotion, or behavior, nor are they required to develop interactive game engines or game production pipelines.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Creativity, Critical Thinking, Resourcefulness, Reflectiveness

S2-GMD-02: Construct a rule-based AI to control an NPC.

Boundary Statement(s)	Students should be able to create a functional, rule-based AI for a Non-Playable Character (NPC) using programming or visual scripting. Students' implementation must be based on a recognized AI method (e.g., finite-state machine, behavior tree). Students should implement an AI that can transition between different behaviors based on specific, pre-defined conditions. For example, students could implement a simple AI for a guard NPC that switches from a "Patrolling" state to a "Chasing" state when the player enters its line of sight, and then to an "Attacking" state when the player is within a certain distance. Students are not expected to implement machine learning models or neural networks to control NPCs. Students are not expected to create an AI that learns from the player or adapts its behavior in non-deterministic ways.
Pillar(s) and Practice(s)	Computational Thinking: 7. Develop and use abstractions. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Persistence, Critical Thinking

S2-GMD-03: Implement basic 2D and 3D animations for game assets using keyframing and foundational animation principles.

Boundary Statement(s)	Students should be able to create short, purposeful animations for game assets by applying keyframing and principles of animation such as anticipation, squash and stretch, and timing. The animations can be for either 2D sprites or 3D models. The goal is to make the assets feel dynamic and believable within a game. For example, a student could animate a simple 2D or 3D character performing an action, such as jumping, where the animation shows the character "squashing" down before the jump (anticipation) and "stretching" as it reaches its peak height (squash and stretch). Students are not expected to create complex, cinematic animations or master advanced techniques like inverse kinematics, character rigging, or cloth simulation.
Pillar(s) and Practice(s)	Computational Thinking: 7. Develop and use abstractions. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Creativity, Persistence

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- Data Science
- **Game Development**
- X + CS

S2-GMD-04: Evaluate game interactions that use a variety of input devices to enhance immersion and player experience.

Boundary Statement(s)	Students should be able to evaluate a game that uses a variety of input devices, specifically to measure and improve the player's sense of immersion and overall experience. Students should take a methodical, iterative approach to collecting both quantitative (e.g., number of errors) and qualitative (e.g., player feedback) data to identify strengths and weaknesses in the input design. Students should then use the data to inform design changes. For example, a student could evaluate a game that uses a motion controller by observing players' physical reactions and gathering their verbal feedback on the "feel" and responsiveness of the controls, then use that information to refine the sensitivity or timing of the inputs. Students are not expected to use statistical analysis. Students are not expected to collect data from large numbers of participants or recruit participants from outside the school context. Students are not required to produce formal research reports.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Reflectiveness, Critical Thinking

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- Data Science
- **Game Development**
- X + CS

S2-GMD-05: Compare two versions of the same game to determine which version performs better based on defined metrics.

Boundary Statement(s)	Students should be able to design and execute a usability study, like A/B testing, to compare two versions of a single game element (e.g., a button's color, a level's layout, a character's ability) while considering responsible design principles (e.g., ethical considerations, inclusivity and accessibility, sustainability, transparency and trust). Students should use study findings to make a data-driven decision on the game's design. For example, a student could run a test with two small, distinct groups of players to determine which of two different tutorial pop-ups results in a higher completion rate of the first level. Students are not expected to work with large-scale data sets, use statistical analysis software, or implement testing on a live product.
Pillar(s) and Practice(s)	Computational Thinking: 6. Define computational problems. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Critical Thinking, Persistence, Reflectiveness

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- Data Science
- **Game Development**
- X + CS

S2-GMD-06: Explain the role of graphics processing units (GPUs) in game development, including their impacts on rendering performance, visual fidelity, and the overall gaming experience.

Boundary Statement(s)	Students should be able to explain the fundamental difference between a Central Processing Unit (CPU) and a Graphics Processing Unit (GPU), and articulate how the GPU's architecture, particularly its reliance on parallel processing, makes it uniquely suited for the task of rendering. They should understand the GPU's role in the gaming pipeline and connect this understanding to tangible impacts on a game's visual presentation (e.g., complexity of lighting, quality of textures, fluidity of animation). For example, students could describe how a GPU's thousands of cores can simultaneously calculate the shading for millions of individual pixels on a screen, enabling realistic lighting and shadows that would be impossible for a sequential-processing CPU to achieve in real-time. Students are not expected to understand the low-level architecture of specific GPU models, write platform-specific code, or perform hardware benchmarking.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Resourcefulness

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- Data Science
- **Game Development**
- X + CS

S2-GMD-07: Apply ethical principles in the design and development of games.

Boundary Statement(s)	Students should create a functional game that demonstrates ethical design principles by integrating accessibility features, promoting inclusivity, avoiding bias, and protecting user well-being and data. Students should apply these principles from the beginning of the design process rather than adding them afterward. Students should also be able to justify their design choices from an ethical standpoint. For example, students could design a game with colorblind-friendly palettes and remappable controls (accessibility), avoid stereotypical character representations (inclusivity and avoiding bias), include content warnings for sensitive topics (user well-being), and implement appropriate data collection practices (data protection). Students are not expected to address every possible accessibility need or ethical consideration, nor are they expected to create polished, market-ready games.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Reflectiveness, Creativity

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- Data Science
- **Game Development**
- X + CS

S2-GMD-08: Collaborate effectively in a project team by defining and executing assigned roles, communicating clearly, and managing shared resources to deliver a game project.

Boundary Statement(s)	Students should participate in a collaborative team environment to complete a game project by communicating effectively, managing their assigned roles and responsibilities, and contributing to shared decision-making. Students should listen to and integrate teammates' ideas, work through disagreements constructively, and be willing to contribute outside their primary expertise when it benefits the team. For example, a student who prefers coding might help brainstorm art concepts with a teammate who is more visually inclined, demonstrating willingness to engage with different aspects of game development to support the team's goals. Students are not expected to use formal project management methodologies or maintain perfect team dynamics throughout the project.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Inclusive Collaboration: 5. Act responsibly in computing collaborations.
Disposition(s)	Persistence, Sense of Belonging in CS

X + CS

S1-XCS-01: Identify and explain connections between CS concepts and practices and those from a non-CS discipline (X).

Boundary Statement(s)	Students should be able to identify and explain, at a conceptual level, how foundational computer science principles, such as data representation, algorithmic thinking, or abstraction, serve as can support understanding or problem-solving within a non-CS field (X) like economics, linguistics, or art history. For a course paired with a literature class, this could look like students drawing a parallel between the concept of procedural abstraction in programming and the role of narrative patterns in literature such as a "Hero's Journey". Students are not expected to develop software or models that computationally solve advanced domain-specific problems.
Pillar(s) and Practice(s)	Inclusive Collaboration: 3. Communicate effectively about computing. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Creativity

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- Data Science
- Game Development
- **X + CS**

S1-XCS-02: Apply computational thinking to reinterpret a problem and design a solution within a non-CS discipline (X).

Boundary Statement(s)	Students should apply computational thinking skills to a problem from a non-CS discipline (X) and design a feasible computational solution. Students should identify a problem from X that can be solved computationally, decompose it into sub-problems, extract common features and patterns (i.e., abstraction), and create step-by-step procedures (i.e., algorithms) to solve it. For example, Biology students studying population dynamics could predict how a prey population will change over time by identifying key variables (e.g., birth rate, predation rate, available resources, carrying capacity), decomposing the problem into sub-problems (e.g., calculating population growth during different seasons, accounting for predator-prey interactions, determining resource limitations), and designing an algorithm that models population changes across multiple generations based on these factors. Students should focus on the design process and generating design documentation (e.g., pseudocode, flowcharts). Students are not expected to create, test, and refine a full computational solution.
Pillar(s) and Practice(s)	Computational Thinking: 7. Develop and use abstractions. Computational Thinking: 8. Create computational artifacts.
Disposition(s)	Critical Thinking, Persistence, Resourcefulness

S1-XCS-03: Investigate a computer science innovation in a non-CS discipline (X).

Boundary Statement(s)	Students should investigate a specific computer science innovation, detailing the foundational computing concepts involved, how it was developed, and its real-world impact on the chosen non-CS discipline. For example, students could investigate how machine learning algorithms are used in medical imaging to detect tumors, detailing the training process using labeled image datasets, the pattern recognition methods that identify abnormalities, and how this innovation has improved diagnostic accuracy and speed. Students are not expected to reproduce the innovation, implement it from scratch, or conduct comprehensive analyses of its societal, economic, or policy implications beyond describing its direct impact on the discipline.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 1. Use computing for positive social impact. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Resourcefulness

S1-XCS-04: Model relationships within a non-CS discipline using data, visualizations, and computational methods.

Boundary Statement(s)	Students should select and apply computational methods (e.g., specific machine learning algorithms, statistical tests) to structure, analyze, and visualize a substantial, real-world dataset from a non-CS discipline (X), revealing patterns and relationships within that discipline. For example, Political Science students could transform raw, disorganized voting records into a structured database, apply a clustering algorithm to identify voter groups, and create a network graph to visualize relationships between political actors or the flow of political influence. Students are not expected to develop novel computational methods or algorithms, nor are they expected to work with datasets so large that they require distributed computing resources.
Pillar(s) and Practice(s)	Computational Thinking: 7. Develop and use abstractions. Human-Centered Design: 10. Understand and involve diverse users in design decisions.
Disposition(s)	Reflectiveness, Resourcefulness, Critical Thinking

Focus Area

- Software Development
- Cybersecurity
- Artificial Intelligence
- Physical Computing
- Data Science
- Game Development
- **X + CS**

S1-XCS-05: Assess how well an algorithm solves problems within a non-CS discipline (X) by analyzing their accuracy, efficiency, and relevance to the intended goal.

Boundary Statement(s)	Students should select and apply appropriate metrics to evaluate an algorithm's performance and suitability for solving a specific problem in a non-CS field (X). Students should assess accuracy (e.g., error rates, precision/recall for classification tasks), efficiency (e.g., runtime, memory usage, scalability), and relevance (e.g., whether the algorithm addresses the actual problem requirements). Students should be able to interpret complexity classifications, compare the efficiency of different algorithmic approaches, and explain trade-offs between accuracy and efficiency in the context of discipline X. For example, Healthcare students assessing a machine learning algorithm to predict patient readmission risk could analyze its prediction accuracy on a test dataset, measure its runtime when processing patient records, and evaluate whether it appropriately accounts for the clinical factors that healthcare providers consider important. Students are not expected to formally derive algorithmic complexity using Big O notation.
Pillar(s) and Practice(s)	Ethics and Social Responsibility: 2. Respect others' rights when creating computational technologies. Computational Thinking: 6. Define computational problems.
Disposition(s)	Critical Thinking, Reflectiveness

References

Association for Computing Machinery (ACM). (2018). *ACM code of ethics and professional conduct.* https://www.acm.org/code-of-ethics

Interaction Design Foundation (IDF). (n.d.a). *Human-centered design (HCD).* https://www.interaction-design.org/literature/topics/human-centered-design

International Technology and Engineering Educators Association (ITEEA). (2020). *Standards for technological and engineering literacy: The role of technology and engineering in STEM education.* https://www.iteea.org/stel

K–12 Computer Science Framework. (2016). http://www.k12cs.org

Learning for Justice. (n.d.). *Social justice standards: The Learning for Justice anti-bias framework.* https://www.learningforjustice.org/frameworks/social-justice-standards

National Governors Association Center for Best Practices (NGA Center) & Council of Chief State School Officers (CCSSO). (2010). *Common Core State Standards for Mathematics.* https://corestandards.org/wp-content/uploads/2023/09/Math_Standards1.pdf

National Institute of Standards and Technology (NIST). (2021). *Human centered design (HCD).* https://www.nist.gov/itl/iad/visualization-and-usability-group/human-factors-human-centered-design

NGSS Lead States. (2013). *Next generation science standards: For states, by states.* The National Academies Press. http://www.nextgenscience.org

Partnership for 21st Century Skills. (2009). *P21 framework definitions.* https://files.eric.ed.gov/fulltext/ED519462.pdf

Tedre, M., Denning, P. J., & Toivonen, T. (2021). CT 2.0. In *Proceedings of the 21st Koli Calling International Conference on Computing Education Research (Koli Calling '21)*. Association for Computing Machinery. https://doi.org/10.1145/3488042.3488053

Vinney, C. (2023). *What is human-centered design? Everything you need to know.* UX Design Institute. https://www.uxdesigninstitute.com/blog/what-is-human-centered-design/

