Maryland
STATE DEPARTMENT OF EDUCATION

Program of Study Guide: Computer Programming - DRAFT

Comprehensive guidelines and course standards for the Computer Programming pathway

Division of College and Career Pathways

July 2025

Table of Contents

Document Control Information

Title:	Program of Study Guide: Computer Programming
Security Level:	Not for Distribution
File Name:	Computer Programming Program Guide.docx

DOCUMENT HISTORY

Document Version	Date	Summary of Change
1.0	October 2024	Initial Document

Purpose

The purpose of this document is to communicate the required Career and Technical Education (CTE) academic standards for the Computer Programming Program of Study. The academic standards in this document are theoretical and performance based. The standards contain content from multiple state departments of education, the College Board, and the Computer Science Teachers Association (CSTA) and have been reviewed and vetted by members of the Maryland business and industry community.

In addition to academic standards, the Maryland State Department of Education (MSDE) has incorporated into this document Labor Market Information (LMI) definitions and explanations for the Program of Study; program aligned Industry Recognized Credentials; and Work-Based Learning resources and requirements by course level.

Standards Sources

The following sources collectively support a progression of standards from foundational to advanced programming concepts in a high school context, preparing students for industry-aligned certifications like Python and Java and providing them with the necessary knowledge and skills for career readiness in computer programming fields.

1. **CodeHS Pro - AP Computer Science Principles with Python**
   A. **Description**: Complete curriculum including Python programming fundamentals, interactive exercises, auto-graded assignments, and project-based learning materials aligned with AP CSP framework
   B. **Use**: Prepares students for both AP CSP exam and Certiport IT Specialist Python certification through structured Python programming instruction
   C. **Source**: CodeHS.com curriculum offerings

2. **Oracle Academy's Java Foundations**
   A. **Description**: Comprehensive Java curriculum including object-oriented programming concepts, hands-on labs, and project materials aligned with AP CSA
   B. **Use**: Builds foundational Java skills needed for AP Computer Science A and Certiport IT Specialist Java certification
   C. **Source**: Oracle Academy educational resources

3. **Certiport's Python Learning Suite**
   A. **Description**: Official exam preparation materials including practice tests, study guides, and lesson plans specifically aligned to certification objectives
   B. **Use**: Direct preparation for IT Specialist Python certification with exam-aligned content and practice assessments
   C. **Source**: Certiport certification preparation materials

4. **Certiport's Java Learning Suite**
   A. **Description**: Official Java certification preparation materials including practice exams, tutorials, and aligned curriculum resources
   B. **Use**: Focused preparation for IT Specialist Java certification using official exam-aligned materials
   C. **Source**: Certiport certification preparation materials

5. **GitHub Learning Lab**
   A. **Description**: Interactive platform teaching Git version control, collaborative development, and project management through hands-on exercises
   B. **Use**: Supports advanced coursework by teaching industry-standard development tools and practices
   C. **Source**: GitHub educational resources

6. **Android Developer Fundamentals**
   A. **Description**: Google's official curriculum for Android development including course materials, codelabs, and practical exercises
   B. **Use**: Provides structure for mobile development portion of advanced coursework
   C. **Source**: Google Android Developers training resources

7. **W3Schools Web Development Tutorial Library**
   A. **Description**: Comprehensive web development tutorials covering HTML, CSS, JavaScript, and full-stack development with interactive examples
   B. **Use**: Supports web development instruction with structured learning paths and hands-on practice
   C. **Source**: W3Schools online learning platform

8. **JetBrains Academy**
   A. **Description**: Project-based learning platform with tracks in Python and Java, including integrated development environments and real-world projects
   B. **Use**: Reinforces programming concepts through practical application and project development
   C. **Source**: JetBrains educational platform

9. **IT Specialist Python Exam Objectives (Certiport)**
   A. **Description**: Comprehensive exam objectives covering Python programming fundamentals, control flow, data structures, functions, error handling, and modules.
   B. **Use**: Primary framework for the Python Foundations course to ensure students develop skills required for certification.
   C. **Source**: Certiport website (https://certiport.pearsonvue.com/Certifications/IT-Specialist/Certification/Python)

10. **AP Computer Science Principles Course and Exam Description (College Board)**
    A. **Description**: College Board's framework for computer science principles, including computational thinking, algorithms, programming, data, and computing impacts.
    B. **Use**: Integrated with Python content to create a course that prepares students for both the IT Specialist Python exam and the AP CSP exam.
    C. **Source**: College Board website (https://apcentral.collegeboard.org/courses/ap-computer-science-principles)

11. **App Development with Swift Certified User Exam Objectives (Certiport)**
    A. **Description**: Certification objectives covering Swift fundamentals, Xcode tools, SwiftUI, and iOS app development.
    B. **Use**: Foundation for iOS App Development with Swift course to prepare students for the Swift Certified User exam.
    C. **Source**: Certiport website (https://certiport.pearsonvue.com/Certifications/Apple/App-Dev-with-Swift/Overview)

12. **Apple Swift Fundamentals Curriculum**
    A. **Description**: Apple's official curriculum introducing Swift programming, interface design, and basic app development.
    B. **Use**: Structured the iOS App Development course to align with Apple's teaching approach and learning progression.
    C. **Source**: Apple Education website (https://www.apple.com/education/k12/teaching-code/)

13. **Apple Swift Data Collections Curriculum**
    A. **Description**: Advanced Swift curriculum focusing on data structures, persistence, networking, and complex app development.
    B. **Use**: Framework for the Advanced iOS Development course's scaffolded learning progression.

C. **Source**: Apple Developer website (https://developer.apple.com/education/teaching-app-development-with-swift/)

14. **Xcode and SwiftUI Documentation**
    A. **Description**: Official technical documentation for Xcode features, Swift language, and SwiftUI framework.
    B. **Use**: Reference for specific technical concepts and capabilities in the iOS development courses.
    C. **Source**: Apple Developer Documentation (https://developer.apple.com/documentation/)

15. Python Software Foundation Documentation
    A. **Description**: Comprehensive documentation of Python language features, standard library, and best practices.
    B. **Use**: Reference for Python language concepts and implementation details in the Python Foundations course.
    C. **Source**: Python.org (https://docs.python.org/)

This document is intended for use by educational administrators and practitioners. A similar document is available for each state-approved CTE Program of Study.

Course Descriptions

Course Level	Course Information	Description
Core: Course 1	Computer Programming I SCED: <XX> Grades: 9-12 Prerequisite: None Credit: 1	This foundational course introduces students to the breadth of computer science through problem-solving, working with data, understanding the internet, cybersecurity, and programming. Students learn computational thinking skills by developing programs in Python, building toward the AP Computer Science Principles exam and the Certiport IT Specialist Python certification. Through hands-on experiences, students explore technology's impact on society while developing fundamental coding skills.
Core: Course 2A	Computer Programming II - Java SCED: <XX> Grades: 10-12 Prerequisite: Computer Programming I Credit: 1	Building on previous programming knowledge, this course provides an in-depth exploration of object-oriented programming using Java. Students master advanced programming concepts including classes, inheritance, algorithms, and data structures. The course emphasizes problem-solving, software engineering principles, and the development of sophisticated computer programs. Students prepare for both the AP Computer Science A exam and the Certiport IT Specialist Java certification while developing projects that demonstrate their programming expertise.
Core: Course 2B	Computer Programming II – Swift SCED: <XX> Grades: 10-12 Prerequisite: Computer Programming I Credit: 1	This course introduces students to iOS app development using the Swift programming language. Students will navigate Xcode, master Swift fundamentals, and build fully-functional iOS applications through hands-on projects. They'll learn essential skills including variables, functions, data structures, interface design with Interface Builder and SwiftUI, multi-screen navigation, and debugging techniques. Following Apple's official curriculum, the course prepares students for the App Development with Swift Certified User certification, demonstrating their ability to create iOS apps that adhere to standard practices with proper UI elements, layouts, and navigation interfaces.

Course Level	Course Information	Description
Optional Flex: Course 1A	Computer Programming III - Capstone SCED: <XX> Grades: 11-12 Prerequisite: Computer Programming I and II Credit: 1	This capstone course allows students to apply their Python and Java programming skills to real-world challenges through project-based learning. Students choose either a Python or Java specialization track to deepen their expertise while working on an industry-mentored project. The course emphasizes software development lifecycle, professional coding practices, collaborative development using Git, and creating portfolio-worthy applications. Students document their development process, present their projects to industry professionals, and may participate in internships or work-based learning experiences. This culminating experience bridges classroom learning with industry expectations, preparing students for both higher education and career opportunities in software development.
Optional Flex: Course 1B	Computer Programming III – Swift Data Collection SCED: <XX> Grades: 11-12 Prerequisite: Computer Programming I and II Credit: 1	The Swift Data Collections course builds on foundational Swift programming skills to develop sophisticated iOS applications with complex data management. Students will master higher-order collection functions, implement persistent data storage solutions, and create network-connected apps through hands-on projects. The curriculum explores protocol-oriented programming, advanced SwiftUI techniques, and integration with core iOS frameworks. Students will design and develop a capstone iOS application incorporating these advanced concepts, demonstrating their ability to create professional-grade mobile software. Following Apple's Swift Data Collections curriculum, this course prepares students for advanced iOS development roles and provides the skills needed to create apps ready for App Store submission.

Course Level	Course Information	Description
Optional Flex: Course 2	Career Connected Learning I SCED: <XX> Grades: 11-12 Prerequisite: Computer Programming II Credit: 1	This flexible, work-based learning course introduces students to real-world applications of classroom knowledge and technical skills through on-the-job experiences and reflective practice. Students engage in career exploration, skill development, and professional networking by participating in youth apprenticeships, registered apprenticeships, pre-apprenticeships, internships, capstone projects, or other approved career-connected opportunities. Variable credit (1–3) accommodates the required on-the-job training hours and related instruction. By integrating industry standards, employability skills, and personalized learning goals, Career Connected Learning I equips students to make informed career decisions, develop a professional portfolio, and build a strong foundation for success in postsecondary education, training, or the workforce.
Optional Flex: Course 3	Career Connected Learning II SCED: <XX> Grades: 11-12 Prerequisite: Career Connected Learning I Credit: 1	Building on the foundational experiences of Career Connected Learning I, this advanced work-based learning course provides students with deeper on-the-job practice, leadership opportunities, and refined career exploration. Students continue to enhance their technical and professional skills, expanding their industry networks and aligning personal goals with evolving career interests. Variable credit (1–3) remains aligned with the required training hours and related instruction. Through elevated responsibilities and skill application, Career Connected Learning II prepares students to confidently transition into higher-level postsecondary programs, apprenticeships, or the workforce.

Dual Enrollment and Career Connected Learning Experiences Must be Aligned to the CTE Core.

Industry-Recognized Credentials and Work-Based Learning

Industry-Recognized Credentials – The standards in this document are aligned to the following certifications:
By the end of Computer Programming I: IT Specialist: Python **By the end of Computer Programming II:** IT Specialist: Java or Apple Development with Swift: Certified User **Optional Credentials (via the Flex Course options):** IT Specialist: Software Development (C), IT Specialist: JavaScript, IT Specialist: HTML and CSS

Work-Based Learning Examples and Resources		
Computer Programming I: Career Awareness	**Computer Programming II: Career Preparation**	**Flex Courses: Career Preparation**
• Industry Visits • Guest Speakers • Participation in Career and Technical Student Organizations • Postsecondary Visits – Program Specific Site Tours • Mock Interviews	• All of Career Awareness plus the following: • Job Shadow • Paid and Unpaid Internships	• Paid and Unpaid Internships • Apprenticeships

Labor Market Information: Definitions and Data

Labor market information (LMI) plays a crucial role in shaping Career and Technical Education (CTE) programs by providing insights into industry demands, employment trends, and skills gaps. This data helps education leaders assess the viability of existing programs and identify opportunities for new offerings. By aligning CTE programs with real-time labor market needs, schools can better prepare students for in-demand careers and ensure that resources are effectively utilized to support pathways that lead to high-quality, sustainable employment.

Standard Occupational Code (SOC) and Aligned Industry:

Indicator	Definition	Pathway Labor Market Data
High Wage[1]	Those occupations that have a 25th percentile wage equal to or greater than the most recent MIT Living Wage Index for one adult in the state of Maryland, and/or leads to a position that pays at least the median hourly or annual wage for the DC-VA-MD-WV Metropolitan Statistical Area (MSA). *Note: A 25th percentile hourly wage of $24.74 or greater is required to meet this definition.*	**Standard Occupational Code:** 15-1251: Computer Programmers **Hourly Wage/Annual Salary:** 25th Percentile: $38.47/$80,018 50th Percentile: $49.70/$103,376 75th Percentile: $64.50/$134,160
High Skill	Those occupations located within the DC-VA-MD-WV Metropolitan Statistical Area (MSA) with the following education or training requirements: completion of an apprenticeship program; completion of an industry-recognized certification or credential; associate's degree, bachelor's degree, or higher.	**Typical Entry-Level Education:** Computer programmers typically need a bachelor's degree to enter the occupation. Most programmers specialize in several programming languages.
In-Demand	Annual growth plus replacement, across all Maryland occupations, is 405 openings between 2024-2029.	**Annual Openings**

[1] Living Wage Calculator: https://livingwage.mit.edu/states/24

Labor Market Information Data Source

Lightcast Q4 2024 Data Set. Lightcast occupation employment data are based on final Lightcast industry data and final Lightcast staffing patterns. Wage estimates are based on Occupational Employment Statistics (QCEW and Non-QCEW Employees classes of worker) and the American Community Survey (Self-Employed and Extended Proprietors). Occupational wage estimates are also affected by county-level Lightcast earnings by industry. Foundational data for the state of Maryland is collected and reported by the Maryland Department of Labor.

Methodology for High Wage Calculations

To combine labor market data across multiple Standard Occupational Classifications (SOCs), a weighted average approach was used to ensure accurate representation of the marketplace. Median wages for each SOC were weighted based on their respective employment levels, reflecting the relative demand for each occupation. This method ensures that occupations with higher employment contribute proportionately to the overall wage calculation. Additionally, job openings from all relevant SOCs were summed to determine the total projected demand. For example, if Mechanical Engineers account for 67% of total employment and Electrical Engineers for 33%, their respective wages are weighted accordingly, and job openings are aggregated to provide a comprehensive view of labor market opportunities. This approach delivers a balanced and accurate representation of both wages and employment demand for the program.

Methodology for In-Demand Calculations

The baseline for annual job openings, taking into account new positions and replacement positions, was determined by taking the average of all annual job openings between 2024 and 2029 across all 797 career sectors at the 5-digit SOC code level. For the 2024-2029 period, average job openings (growth + replacement) is 405.

Course Standards: Computer Programming I

1. **GENERAL REQUIREMENTS.** This course is recommended for students in Grades 9-12.

2. **INTRODUCTION**
   - A. Career and Technical Education (CTE) instruction provides content aligned with challenging academic standards and relevant technical knowledge and skills for students to further their education and succeed in current or emerging professions.
   - B. The Digital Technology Career Cluster focuses on building linkages in IT occupations for entry level, technical, and professional careers related to the design, development, support, and management of hardware, software, multimedia, and systems integration services.
   - C. The Computer Programming Program of Study provides high school students with comprehensive software development skills through mastery of Python and Java programming languages. Students progress from fundamental coding concepts to complex development projects while earning industry certifications and building professional portfolios. Graduates emerge prepared for both college computer science programs and immediate entry into software development careers, equipped with practical experience in current programming practices and software engineering principles.
   - D. Computer Programming I introduces students to Python programming through hands-on development of algorithms and programs. Students master fundamental concepts like variables, data structures, and control flow while developing essential problem-solving and debugging skills. The course emphasizes professional practices including code documentation and collaboration, culminating in Python certification preparation. This foundation prepares students for advanced programming courses and establishes core competencies needed in software development.
   - E. Students will participate in at least two Career-Connected Education and Work-Based Learning experiences in this course, which might include informational interviews or job shadowing relevant to the program of study.
   - F. Students are encouraged to participate in extended learning experiences through aligned Career and Technical Student Organizations (CTSOs). CTSOs are a cocurricular requirement in the Carl D. Perkins Act, and alignment to CTSO activities is an expectation for CTE programs in the state of Maryland.

3. KNOWLEDGE AND SKILLS

A. The student demonstrates the necessary skills for career development, maintenance of employability, and successful completion of course outcomes. The student is expected to:

1. Identify and demonstrate positive work behaviors that enhance employability and job advancement, such as regular attendance, promptness, proper attire, maintenance of a clean and safe work environment, and pride in work.
2. Demonstrate positive personal qualities such as flexibility, open-mindedness, initiative, active listening, and a willingness to learn.
3. Employ effective reading, writing, and technical documentation skills.
4. Solve problems using critical thinking techniques and structured troubleshooting methodologies.
5. Demonstrate leadership skills and collaborate effectively as a team member.
6. Implement safety procedures, including proper handling of hardware and following cybersecurity guidelines.
7. Exhibit an understanding of legal and ethical responsibilities in the IT field, following data privacy laws and best practices for security.
8. Demonstrate time-management skills and the ability to prioritize tasks in a technical setting.

B. The student identifies various career pathways in the Computer Programming field. The student is expected to:

1. Develop a career plan that includes the necessary education, certifications, job skills, and experience for specific roles in Computer Programming.
2. Create a professional resume and portfolio that reflect skills, projects, certifications, and recommendations.
3. Demonstrate effective interview skills for roles in IT and Computer Programming.

C. The student develops technology and digital literacy skills. The student is expected to:

1. Use technology as a tool for research, organization, communication, and problem-solving.
2. Use digital tools, including computers, mobile devices, collaboration platforms, and cloud services, to access, manage, and create information.
3. Demonstrate proficiency in using emerging and industry-standard technologies, including virtualization tools, network management software, and cybersecurity applications.
4. Understand ethical and legal considerations for technology use, including the principles of data protection, copyright, and responsible technology use.

D. The student integrates core academic skills into computer programming practices. The student is expected to:

1. Demonstrate the use of clear communication techniques, both written and verbal, that are consistent with industry standards.
2. Apply fundamental mathematical operations and expressions, including basic arithmetic, variables, constants, and order of operations. Analyze patterns and relationships in data structures. Develop logical thinking through Boolean operations and flow control.
3. Investigate systematic problem-solving approaches through algorithmic thinking and debugging. Apply iterative testing methods to validate solutions.

E. The student can explain basic computing concepts. The student is expected to:

1. Identify how computers represent information using binary.
2. Describe how computers follow instructions in programs.
3. Explain what an algorithm is and give everyday examples.
4. Discuss how the internet connects people and enables sharing of ideas.
5. Recognize how technology impacts daily life and future careers.

F. The student can discuss how computing affects society. The student is expected to:

1. Identify positive ways technology is used in school, home, and community.
2. Recognize potential risks of sharing personal information online.
3. Discuss how apps and websites are designed to keep users engaged.
4. Identify technology access differences among different communities.
5. Describe how innovations in computing create new career opportunities.

G. The student can use Python programming tools. The student is expected to:

1. Set up and open the Python programming environment.
2. Write and run simple Python statements.
3. Recognize when a program has errors and identify the error messages.
4. Add comments to explain what a program does.
5. Save and organize Python files for a project.

H. The student can work with basic Python elements. The student is expected to:

1. Create variables to store information in a program.
2. Use different data types for different kinds of information (numbers, text, true/false).
3. Convert data from one type to another when needed.
4. Perform basic math operations in Python programs.
5. Create and display messages to users.

I. The student can create interactive programs. The student is expected to:

1. Get information from users with the input() function.
2. Create clear prompts that explain what input is needed.
3. Format output to make information easy to read.
4. Convert user input to the correct data type for processing.
5. Create a complete program that gets input, processes it, and shows results.

J. The student can create programs that make decisions. The student is expected to:

1. Use if statements to run code only when certain conditions are true.
2. Add else statements to provide alternative actions.
3. Create programs with multiple conditions using elif.
4. Combine conditions using and, or, and not.
5. Create programs that respond differently based on user input.

K. The student can create programs that repeat actions. The student is expected to:

1. Use for loops to repeat actions a specific number of times.
2. Create while loops that continue until a condition changes.
3. Use loops to process lists of information.
4. Avoid common loop problems like infinite loops.
5. Create programs that collect and process multiple pieces of information.

L. The student can create and use lists. The student is expected to:

1. Create lists to store multiple pieces of related information.
2. Access specific items in a list using their position (index).
3. Add, remove, and change items in a list.
4. Find the highest, lowest, or average value in a list of numbers.
5. Sort lists to organize information.

M. The student can process data in lists. The student is expected to:

1. Use loops to work with each item in a list.
2. Use lists to keep track of information throughout a program.
3. Create simple programs that analyze data in lists.
4. Search for specific information in a list.
5. Create a program that solves a real problem using lists.

N. The student can create and use functions. The student is expected to:

1. Break down a complex problem into smaller tasks that can be solved with functions.
2. Create functions that perform specific tasks when called.
3. Pass information to functions using parameters.
4. Return results from functions.
5. Document what a function does and how to use it.

O. The student can use Python libraries. The student is expected to:

1. Import and use built-in Python modules.
2. Use the random module to create games or simulations.
3. Apply the math module to solve mathematical problems.
4. Use the datetime module to work with dates and times.
5. Identify when to use existing modules instead of writing new code.

P. The student can read and write files. The student is expected to:

1. Open and read information from text files.
2. Write information to new text files.
3. Add information to existing files.
4. Process data from CSV files (like spreadsheet data).
5. Create programs that read, process, and write data to files.

Q. The student can make programs more reliable. The student is expected to:

1. Identify common types of errors in programs.
2. Use try-except blocks to handle errors gracefully.
3. Provide helpful error messages to users.
4. Test programs with different kinds of input.
5. Create programs that can recover from common errors.

R. The student can create complete Python programs. The student is expected to:

1. Plan a program that solves a specific problem.
2. Use appropriate Python features to implement a solution.
3. Test the program to make sure it works correctly.
4. Explain how their program works and why they made certain choices.
5. Create a final project that demonstrates their Python programming skills.

Course Standards: Computer Programming II - Java

1. **GENERAL REQUIREMENTS.** This course is recommended for students in Grades 10-12.

2. **INTRODUCTION**
   A. Career and Technical Education (CTE) instruction provides content aligned with challenging academic standards and relevant technical knowledge and skills for students to further their education and succeed in current or emerging professions.
   B. The Digital Technology Career Cluster focuses on building linkages in IT occupations for entry level, technical, and professional careers related to the design, development, support, and management of hardware, software, multimedia, and systems integration services.
   C. The Computer Programming Program of Study provides high school students with comprehensive software development skills through mastery of Python and Java programming languages. Students progress from fundamental coding concepts to complex development projects while earning industry certifications and building professional portfolios. Graduates emerge prepared for both college computer science programs and immediate entry into software development careers, equipped with practical experience in current programming practices and software engineering principles.
   D. Computer Programming II advances students' expertise through Java programming and object-oriented development principles. Students apply inheritance, polymorphism, and data structures to create complex applications while mastering professional debugging and testing strategies. The course emphasizes software engineering practices and collaborative development, culminating in Java certification preparation. This comprehensive foundation prepares students for both college computer science programs and industry roles.
   E. Students will participate in at least two Career-Connected Education and Work-Based Learning experiences in this course, which might include informational interviews or job shadowing relevant to the program of study.
   F. Students are encouraged to participate in extended learning experiences through aligned Career and Technical Student Organizations (CTSOs). CTSOs are a cocurricular requirement in the Carl D. Perkins Act, and alignment to CTSO activities is an expectation for CTE programs in the state of Maryland.

3. **KNOWLEDGE AND SKILLS**

A. The student demonstrates the necessary skills for career development, maintenance of employability, and successful completion of course outcomes. The student is expected to:

1. Identify and demonstrate positive work behaviors that enhance employability and job advancement, such as regular attendance, promptness, proper attire, maintenance of a clean and safe work environment, and pride in work.
2. Demonstrate positive personal qualities such as flexibility, open-mindedness, initiative, active listening, and a willingness to learn.
3. Employ effective reading, writing, and technical documentation skills.
4. Solve problems using critical thinking techniques and structured troubleshooting methodologies.
5. Demonstrate leadership skills and collaborate effectively as a team member.
6. Implement safety procedures, including proper handling of hardware and following cybersecurity guidelines.
7. Exhibit an understanding of legal and ethical responsibilities in the IT field, following data privacy laws and best practices for security.
8. Demonstrate time-management skills and the ability to prioritize tasks in a technical setting.

B. The student identifies various career pathways in the computer programming field. The student is expected to:

1. Develop a career plan that includes the necessary education, certifications, job skills, and experience for specific roles in computer programming.
2. Create a professional resume and portfolio that reflect skills, projects, certifications, and recommendations.
3. Demonstrate effective interview skills for roles in IT and computer programming.

C. The student develops technology and digital literacy skills. The student is expected to:

1. Use technology as a tool for research, organization, communication, and problem-solving.
2. Use digital tools, including computers, mobile devices, collaboration platforms, and cloud services, to access, manage, and create information.
3. Demonstrate proficiency in using emerging and industry-standard technologies, including virtualization tools, network management software, and cybersecurity applications.
4. Understand ethical and legal considerations for technology use, including the principles of data protection, copyright, and responsible technology use.

D. The student integrates core academic skills into computer programming practices. The student is expected to:

1. Demonstrate the use of clear communication techniques, both written and verbal, that are consistent with industry standards.
2. Evaluate algebraic functions and equations, implement Boolean algebra and logic, apply basic statistics for data analysis, utilize different number systems (binary, decimal), calculate percentages and proportions, and employ basic geometry for graphics problems.
3. Apply systematic investigation and controlled testing methodologies. Analyze data collection methods and pattern recognition. Implement documentation processes for replicable results.

E. The student can analyze Java language fundamentals. The student is expected to:

1. Compare and contrast primitive and reference data types in program implementation.
2. Construct programs using appropriate variable declarations and initialization.
3. Apply proper syntax for arithmetic operators and mathematical computations.
4. Evaluate type conversion requirements and implement casting.
5. Create programs utilizing the Scanner class for user input.
6. Design programs that format and display output effectively.

F. The student can create object-oriented programming solutions. The student is expected to:

1. Design classes that model real-world objects with appropriate attributes and methods.
2. Implement multiple constructors for flexible object creation.
3. Apply encapsulation principles using public and private access modifiers.
4. Develop methods that effectively manipulate object state.
5. Create toString methods that properly represent object state.
6. Construct equals methods for object comparison

G. The student can apply program control structures. The student is expected to:

1. Implement selection statements to control program flow.
2. Design iteration solutions using appropriate loop structures.
3. Create programs that process one-dimensional arrays.
4. Develop algorithms for searching and sorting arrays.
5. Construct nested loop structures for complex iterations.
6. Generate solutions using two-dimensional arrays.

H. The student can evaluate inheritance relationships. The student is expected to:

1. Design class hierarchies that model real-world relationships.
2. Create abstract classes and methods for code reuse.
3. Implement interfaces to define class behavior.
4. Apply polymorphic principles in program design.
5. Analyze when to use inheritance versus composition.
6. Develop programs utilizing method overriding.

I. The student can develop data structure implementations. The student is expected to:

1. Create and manipulate ArrayList objects.
2. Implement common collection methods.
3. Compare efficiency of different data structures.
4. Design programs using appropriate data structures.
5. Apply iteration techniques for data processing.
6. Implement sorting and searching algorithms.

J. The student can implement exception handling. The student is expected to:

1. Design robust error handling mechanisms.
2. Create custom exception classes.
3. Apply try-catch blocks appropriately.
4. Develop input validation routines.
5. Handle file I/O exceptions effectively.
6. Implement proper exception hierarchies.

K. The student can synthesize file operations. The student is expected to:

1. Create programs that read from text files.
2. Develop programs that write to text files.
3. Implement file processing algorithms.
4. Design data persistence solutions.
5. Apply proper file closing procedures.
6. Handle file-related exceptions.

L. The student can evaluate program correctness. The student is expected to:

1. Design comprehensive test cases.
2. Implement debugging strategies.
3. Apply code review techniques.
4. Analyze program efficiency.
5. Create program documentation.
6. Validate program output.

M. The student can create professional-quality code. The student is expected to:

1. Apply Java naming conventions.
2. Implement proper code organization.
3. Create clear documentation using Javadoc.
4. Design reusable code components.
5. Follow object-oriented design principles.
6. Apply best practices for code maintenance.

Course Standards: Computer Programming II - Swift

1. **GENERAL REQUIREMENTS.** This course is recommended for students in Grades 10-12.

2. **INTRODUCTION**
   A. Career and Technical Education (CTE) instruction provides content aligned with challenging academic standards and relevant technical knowledge and skills for students to further their education and succeed in current or emerging professions.
   B. The Digital Technology Career Cluster focuses on building linkages in IT occupations for entry level, technical, and professional careers related to the design, development, support, and management of hardware, software, multimedia, and systems integration services.
   C. The Computer Programming Program of Study provides high school students with comprehensive software development skills through mastery of Python and Swift programming languages. Students progress from fundamental coding concepts to complex development projects while earning industry certifications and building professional portfolios. Graduates emerge prepared for both college computer science programs and immediate entry into software development careers, equipped with practical experience in current programming practices and software engineering principles.
   D. Computer Programming II – Swift introduces students to iOS app development using the Swift programming language. Students will navigate Xcode, master Swift fundamentals, and build fully-functional iOS applications through hands-on projects. They'll learn essential skills including variables, functions, data structures, interface design with Interface Builder and SwiftUI, multi-screen navigation, and debugging techniques. Following Apple's official curriculum, the course prepares students for the App Development with Swift Certified User certification, demonstrating their ability to create iOS apps that adhere to standard practices with proper UI elements, layouts, and navigation interfaces.
   E. Students will participate in at least two Career-Connected Education and Work-Based Learning experiences in this course, which might include informational interviews or job shadowing relevant to the program of study.
   F. Students are encouraged to participate in extended learning experiences through aligned Career and Technical Student Organizations (CTSOs). CTSOs are a cocurricular requirement in the Carl D. Perkins Act, and alignment to CTSO activities is an expectation for CTE programs in the state of Maryland.

3. **KNOWLEDGE AND SKILLS**

   A. **The student demonstrates the necessary skills for career development, maintenance of employability, and successful completion of course outcomes. The student is expected to:**
      1. Identify and demonstrate positive work behaviors that enhance employability and job advancement, such as regular attendance, promptness, proper attire, maintenance of a clean and safe work environment, and pride in work.
      2. Demonstrate positive personal qualities such as flexibility, open-mindedness, initiative, active listening, and a willingness to learn.
      3. Employ effective reading, writing, and technical documentation skills.
      4. Solve problems using critical thinking techniques and structured troubleshooting methodologies.
      5. Demonstrate leadership skills and collaborate effectively as a team member.
      6. Implement safety procedures, including proper handling of hardware and following cybersecurity guidelines.
      7. Exhibit an understanding of legal and ethical responsibilities in the IT field, following data privacy laws and best practices for security.
      8. Demonstrate time-management skills and the ability to prioritize tasks in a technical setting.

   B. **The student identifies various career pathways in the computer programming field. The student is expected to:**
      1. Develop a career plan that includes the necessary education, certifications, job skills, and experience for specific roles in computer programming.
      2. Create a professional resume and portfolio that reflect skills, projects, certifications, and recommendations.
      3. Demonstrate effective interview skills for roles in IT and computer programming.

   C. **The student develops technology and digital literacy skills. The student is expected to:**
      1. Use technology as a tool for research, organization, communication, and problem-solving.
      2. Use digital tools, including computers, mobile devices, collaboration platforms, and cloud services, to access, manage, and create information.
      3. Demonstrate proficiency in using emerging and industry-standard technologies, including virtualization tools, network management software, and cybersecurity applications.
      4. Understand ethical and legal considerations for technology use, including the principles of data protection, copyright, and responsible technology use.

   D. **The student integrates core academic skills into computer programming practices. The student is expected to:**
      1. Demonstrate the use of clear communication techniques, both written and verbal, that are consistent with industry standards.
      2. Evaluate algebraic functions and equations, implement Boolean algebra and logic, apply basic statistics for data analysis, utilize different number systems (binary, decimal), calculate percentages and proportions, and employ basic geometry for graphics problems.
      3. Apply systematic investigation and controlled testing methodologies. Analyze data collection methods and pattern recognition. Implement documentation processes for replicable results.

E. The student can describe the iOS app development ecosystem. The student is expected to:

1. Explain what Swift is and how it's used in iOS development.
2. Identify the components of the Apple development platform.
3. Describe the process of creating, testing, and deploying iOS apps.
4. Understand the purpose of the App Store and app distribution methods.
5. Recognize how Swift relates to Apple's hardware and software ecosystem.

F. The student can set up and navigate the development environment. The student is expected to:

1. Install and configure Xcode on a Mac computer.
2. Navigate the Xcode workspace including the navigator, editor, and inspector areas.
3. Create, open, and manage Swift projects and playgrounds.
4. Use the iOS simulator to test applications.
5. Access documentation and resources within Xcode.

G. The student can understand and apply Swift programming fundamentals. The student is expected to:

1. Declare and use basic Swift types (Int, Double, String, Bool).
2. Demonstrate when to use constants (let) versus variables (var).
3. Apply arithmetic, comparison, and logical operators to create expressions.
4. Interpret and predict the results of Swift expressions.
5. Use Swift playgrounds to experiment with code and see immediate results.

H. The student can implement control flow in Swift programs. The student is expected to:

1. Create decision structures using if, else if, and else statements.
2. Apply logical operators (&&, ||, !) to form compound conditions.
3. Use switch statements to handle multiple potential states.
4. Implement the guard statement for early exits from functions.
5. Develop programs that respond to different conditions.

I. The student can use loops for repetitive tasks in Swift. The student is expected to:

1. Create for-in loops to iterate through ranges and collections.
2. Implement while loops for condition-based execution.
3. Apply loop control statements (break, continue) to control execution flow.
4. Iterate through characters in strings and elements in collections.
5. Choose appropriate loop structures for specific programming tasks.

J. The student can manage data using Swift collection types. The student is expected to:

1. Create and initialize arrays to store ordered collections of data.
2. Access and modify array elements using index notation.
3. Perform common array operations (append, insert, remove, count).
4. Create and use dictionaries to store key-value pairs.
5. Access and modify dictionary values using key subscripting.

K. The student can define and use functions in Swift. The student is expected to:

1. Create functions with parameters and return values.
2. Implement functions with default parameter values.
3. Customize internal and external parameter names in functions.
4. Understand function scope and variable lifetime.
5. Organize code using functions for reusability and maintainability.

L. The student can work with optional types in Swift. The student is expected to:

1. Explain the purpose of optionals in Swift programming.
2. Unwrap optionals safely using if let and guard let statements.
3. Implement optional binding to safely access optional values.
4. Apply optional chaining to access properties and methods of optional values.
5. Use nil coalescing operators to provide default values for optionals.

M. The student can create and use Swift structures and classes. The student is expected to:

1. Define structures with properties and methods.
2. Initialize structures using memberwise and custom initializers.
3. Define classes with properties and methods.
4. Differentiate between structures and classes in Swift.
5. Implement property observers (willSet, didSet).

N. The student can debug Swift code effectively. The student is expected to:

1. Set breakpoints to pause execution at specific points in code.
2. Use the debugger to step through code line by line.
3. Inspect variable values during debugging sessions.
4. Implement logging to track program execution.
5. Diagnose and fix common errors in Swift programs.

O. The student can create user interfaces with Interface Builder. The student is expected to:

1. Add and configure UI elements from the Object Library.
2. Set properties for UI components using the Attributes Inspector.
3. Create connections between Interface Builder and Swift code using outlets and actions.
4. Design interfaces using storyboards and view controllers.
5. Implement Auto Layout for responsive interfaces.

P. The student can create simple SwiftUI views. The student is expected to:

1. Understand the declarative syntax of SwiftUI.
2. Create and customize basic SwiftUI views.
3. Apply modifiers to change the appearance and behavior of views.
4. Implement stacks (VStack, HStack, ZStack) to arrange views.
5. Create simple layouts that adapt to different screen sizes.

Q. The student can build navigation and multi-screen apps. The student is expected to:

1. Implement navigation between screens using NavigationStack and NavigationLink.
2. Present modal views using sheets.
3. Create tab-based interfaces for organizing app content.
4. Pass data between different screens in an app.
5. Design user flows that follow iOS interface guidelines.

R. The student can complete guided app projects to build practical skills. The student is expected to:

1. Build a simple flashlight app (Light) that demonstrates basic UI interaction.
2. Create a word-guessing game app (Apple Pie) using UIKit components and Auto Layout.
3. Develop a personality quiz app that showcases navigation between screens.
4. Plan, prototype, and implement a personal app project.
5. Document the development process and design decisions.

Course Standards: Computer Programming III – Capstone

1. **GENERAL REQUIREMENTS.** This course is recommended for students in Grades 10-12.

2. **INTRODUCTION**
   A. Career and Technical Education (CTE) instruction provides content aligned with challenging academic standards and relevant technical knowledge and skills for students to further their education and succeed in current or emerging professions.
   B. The Digital Technology Career Cluster focuses on building linkages in IT occupations for entry level, technical, and professional careers related to the design, development, support, and management of hardware, software, multimedia, and systems integration services.
   C. The Computer Programming Program of Study provides high school students with comprehensive software development skills through mastery of Python and Java programming languages. Students progress from fundamental coding concepts to complex development projects while earning industry certifications and building professional portfolios. Graduates emerge prepared for both college computer science programs and immediate entry into software development careers, equipped with practical experience in current programming practices and software engineering principles.
   D. Computer Programming IV advances students' programming skills through practical experience in full-stack development, focusing on web and Android applications using frameworks like Flask, Spring Boot, and Android Studio. Students master modern development practices including API design, cloud deployment, and agile methodologies while creating production-ready applications. Through hands-on projects and industry-standard tools, students build a professional portfolio that demonstrates expertise in both web and mobile development. This course prepares students for careers such as full-stack developer, mobile app developer, DevOps engineer, or API specialist, while providing a strong foundation for computer science degrees with a focus on modern application development.
   E. Students will participate in at least two Career-Connected Education and Work-Based Learning experiences in this course, which might include informational interviews or job shadowing relevant to the program of study.
   F. Students are encouraged to participate in extended learning experiences through aligned Career and Technical Student Organizations (CTSOs). CTSOs are a cocurricular requirement in the Carl D. Perkins Act, and alignment to CTSO activities is an expectation for CTE programs in the state of Maryland.

3. **KNOWLEDGE AND SKILLS**

A. The student demonstrates the necessary skills for career development, maintenance of employability, and successful completion of course outcomes. The student is expected to:

1. Identify and demonstrate positive work behaviors that enhance employability and job advancement, such as regular attendance, promptness, proper attire, maintenance of a clean and safe work environment, and pride in work.
2. Demonstrate positive personal qualities such as flexibility, open-mindedness, initiative, active listening, and a willingness to learn.
3. Employ effective reading, writing, and technical documentation skills.
4. Solve problems using critical thinking techniques and structured troubleshooting methodologies.
5. Demonstrate leadership skills and collaborate effectively as a team member.
6. Implement safety procedures, including proper handling of hardware and following cybersecurity guidelines.
7. Exhibit an understanding of legal and ethical responsibilities in the IT field, following data privacy laws and best practices for security.
8. Demonstrate time-management skills and the ability to prioritize tasks in a technical setting.

B. The student identifies various career pathways in the computer programming field. The student is expected to:

1. Develop a career plan that includes the necessary education, certifications, job skills, and experience for specific roles in computer programming.
2. Create a professional resume and portfolio that reflect skills, projects, certifications, and recommendations.
3. Demonstrate effective interview skills for roles in IT and computer programming.

C. The student develops technology and digital literacy skills. The student is expected to:

5. Use technology as a tool for research, organization, communication, and problem-solving.
6. Use digital tools, including computers, mobile devices, collaboration platforms, and cloud services, to access, manage, and create information.
7. Demonstrate proficiency in using emerging and industry-standard technologies, including virtualization tools, network management software, and cybersecurity applications.
1. Understand ethical and legal considerations for technology use, including the principles of data protection, copyright, and responsible technology use.

D. The student integrates core academic skills into computer programming practices. The student is expected to:

1. Demonstrate the use of clear communication techniques, both written and verbal, that are consistent with industry standards.
2. Apply mathematical concepts such as Algebraic functions and equations, Boolean algebra and logic, basic statistics for data analysis, order of operations, number systems (binary, decimal), percentages and proportions, basic geometry (for graphics and spatial problems) and pattern recognition.
3. Use scientific principles, such as data collection, pattern recognition, systematic investigation, controlled testing, and documentation of process.

E. The student can analyze professional software development practices. The student is expected to:

1. Evaluate various software development methodologies (Agile, Scrum, Waterfall).
2. Apply version control systems using Git for code management.
3. Implement project planning and tracking tools.
4. Design comprehensive documentation for software projects.
5. Create effective technical specifications and requirements.
6. Assess project risks and develop mitigation strategies.

F. The student can create advanced programming solutions. The student is expected to:

1. Develop complex applications using chosen language (Python/Java).
2. Implement design patterns appropriate to project requirements.
3. Create modular, reusable code components.
4. Design efficient algorithms for complex problems.
5. Build robust error handling and logging systems.
6. Integrate third-party libraries and APIs effectively.

G. The student can evaluate software testing methodologies. The student is expected to:

1. Design comprehensive test plans and test cases.
2. Implement unit testing frameworks.
3. Perform integration testing across system components.
4. Create automated testing scripts.
5. Apply debugging strategies for complex issues.
6. Document testing procedures and results.

G. The student can develop database integration solutions. The student is expected to:

1. Design database schemas for applications.
2. Implement CRUD operations in applications.
3. Create efficient database queries.
4. Apply database security best practices.
5. Handle data validation and sanitization.
6. Manage database connections effectively.

H. The student can synthesize user interface design principles. The student is expected to:

1. Create intuitive user interfaces.
2. Implement responsive design principles.
3. Apply accessibility standards.
4. Design effective user feedback systems.
5. Develop input validation mechanisms.
6. Create consistent design patterns.

I. The student can manage project development cycles. The student is expected to:

1. Plan project timelines and milestones.
2. Coordinate team member responsibilities.
3. Conduct code reviews and provide feedback.
4. Manage project dependencies.
5. Track and resolve project issues.
6. Present project progress to stakeholders.

J. The student can apply security best practices. The student is expected to:

1. Implement secure coding practices.
2. Create authentication and authorization systems.
3. Protect against common security vulnerabilities.
4. Manage sensitive data appropriately.
5. Perform security testing.
6. Document security measures.

K. The student can create professional portfolios. The student is expected to:

1. Document project development processes.
2. Create technical presentations.
3. Develop project demonstrations.
4. Write clear project documentation.
5. Present solutions to stakeholders.
6. Maintain professional online presence.

Course Standards: Computer Programming III – Swift Data Collections

1. **GENERAL REQUIREMENTS.** This course is recommended for students in Grades 10-12.

2. **INTRODUCTION**
   A. Career and Technical Education (CTE) instruction provides content aligned with challenging academic standards and relevant technical knowledge and skills for students to further their education and succeed in current or emerging professions.
   B. The Digital Technology Career Cluster focuses on building linkages in IT occupations for entry level, technical, and professional careers related to the design, development, support, and management of hardware, software, multimedia, and systems integration services.
   C. The Computer Programming Program of Study provides high school students with comprehensive software development skills through mastery of Python and Swift programming languages. Students progress from fundamental coding concepts to complex development projects while earning industry certifications and building professional portfolios. Graduates emerge prepared for both college computer science programs and immediate entry into software development careers, equipped with practical experience in current programming practices and software engineering principles.
   D. Computer Programming III – Swift Data Collections course builds on foundational Swift programming skills to develop sophisticated iOS applications with complex data management. Students will master higher-order collection functions, implement persistent data storage solutions, and create network-connected apps through hands-on projects. The curriculum explores protocol-oriented programming, advanced SwiftUI techniques, and integration with core iOS frameworks. Students will design and develop a capstone iOS application incorporating these advanced concepts, demonstrating their ability to create professional-grade mobile software. Following Apple's Swift Data Collections curriculum, this course prepares students for advanced iOS development roles and provides the skills needed to create apps ready for App Store submission.
   E. Students will participate in at least two Career-Connected Education and Work-Based Learning experiences in this course, which might include informational interviews or job shadowing relevant to the program of study.
   F. Students are encouraged to participate in extended learning experiences through aligned Career and Technical Student Organizations (CTSOs). CTSOs are a cocurricular requirement in the Carl D. Perkins Act, and alignment to CTSO activities is an expectation for CTE programs in the state of Maryland.

3. **KNOWLEDGE AND SKILLS**

A. The student demonstrates the necessary skills for career development, maintenance of employability, and successful completion of course outcomes. The student is expected to:

1. Identify and demonstrate positive work behaviors that enhance employability and job advancement, such as regular attendance, promptness, proper attire, maintenance of a clean and safe work environment, and pride in work.
2. Demonstrate positive personal qualities such as flexibility, open-mindedness, initiative, active listening, and a willingness to learn.
3. Employ effective reading, writing, and technical documentation skills.
4. Solve problems using critical thinking techniques and structured troubleshooting methodologies.
5. Demonstrate leadership skills and collaborate effectively as a team member.
6. Implement safety procedures, including proper handling of hardware and following cybersecurity guidelines.
7. Exhibit an understanding of legal and ethical responsibilities in the IT field, following data privacy laws and best practices for security.
8. Demonstrate time-management skills and the ability to prioritize tasks in a technical setting.

B. The student identifies various career pathways in the computer programming field. The student is expected to:

4. Develop a career plan that includes the necessary education, certifications, job skills, and experience for specific roles in computer programming.
5. Create a professional resume and portfolio that reflect skills, projects, certifications, and recommendations.
6. Demonstrate effective interview skills for roles in IT and computer programming.

C. The student develops technology and digital literacy skills. The student is expected to:

1. Use technology as a tool for research, organization, communication, and problem-solving.
2. Use digital tools, including computers, mobile devices, collaboration platforms, and cloud services, to access, manage, and create information.
3. Demonstrate proficiency in using emerging and industry-standard technologies, including virtualization tools, network management software, and cybersecurity applications.
4. Understand ethical and legal considerations for technology use, including the principles of data protection, copyright, and responsible technology use.

D. The student integrates core academic skills into computer programming practices. The student is expected to:

1. Demonstrate the use of clear communication techniques, both written and verbal, that are consistent with industry standards.
2. Apply mathematical concepts such as Algebraic functions and equations, Boolean algebra and logic, basic statistics for data analysis, order of operations, number systems (binary, decimal), percentages and proportions, basic geometry (for graphics and spatial problems) and pattern recognition.
3. Use scientific principles, such as data collection, pattern recognition, systematic investigation, controlled testing, and documentation of process.

E. The student can extend their Swift collection knowledge. The student is expected to:

1. Review and apply basic collection types (arrays, dictionaries, sets).
2. Create and manipulate nested collections for complex data structures.
3. Implement custom collection initialization techniques.
4. Compare and select appropriate collection types for specific scenarios.
5. Apply appropriate access patterns to retrieve data efficiently.

F. The student can implement higher-order collection functions. The student is expected to:

1. Use the map function to transform collection elements.
2. Apply filter to create subsets of collections based on criteria.
3. Implement reduce to combine collection elements into a single value.
4. Create custom sorting with closure expressions.
5. Chain multiple higher-order functions for complex data transformations.

G. The student can implement persistent data storage. The student is expected to:

1. Save and load basic data using the file system.
2. Store user preferences using UserDefaults.
3. Implement data serialization using Codable protocol.
4. Handle data persistence errors appropriately.
5. Design data models that support effective serialization.

H. The student can design and implement protocols and extensions. The student is expected to:

1. Create custom protocols to define shared behavior.
2. Extend existing types with new functionality.
3. Implement protocol conformance for custom types.
4. Apply protocol-oriented design principles to solve problems.
5. Use protocol extensions to provide default implementations.

I. The student can perform basic networking operations. The student is expected to:

1. Create network requests using URLSession.
2. Handle network responses and errors appropriately.
3. Parse JSON data from web services.
4. Implement asynchronous data fetching.
5. Update user interfaces based on network results.

J. The student can create advanced user interface components. The student is expected to:

1. Implement custom table and collection views.
2. Create custom cells with dynamic content.
3. Handle user interaction with complex UI components.
4. Apply proper data source and delegate patterns.
5. Implement search and filtering in collection-based interfaces.

K. The student can build advanced SwiftUI interfaces. The student is expected to:

1. Create custom view components for reusability.
2. Implement state management with property wrappers.
3. Design responsive layouts that adapt to different screen sizes.
4. Apply animations and transitions between view states.
5. Share data across views using environment objects.

L. The student can integrate system frameworks. The student is expected to:

1. Implement basic location services using Core Location.
2. Display maps and annotations with MapKit.
3. Create and schedule local notifications.
4. Access and use the device camera and photo library.
5. Implement basic device authentication features.

M. The student can apply app architecture patterns. The student is expected to:

1. Organize code using the Model-View-Controller pattern.
2. Implement proper separation of concerns in app design.
3. Apply the delegate pattern for component communication.
4. Create reusable components with clearly defined responsibilities.
5. Design data flow strategies for complex apps.

N. The student can perform testing and debugging. The student is expected to:

1. Write basic unit tests for Swift code.
2. Debug common issues using Xcode's debugging tools.
3. Implement logging for troubleshooting.
4. Test app functionality across different scenarios.
5. Create strategies for handling potential runtime errors.

O. The student can develop a capstone iOS application. The student is expected to:

1. Plan and design an app that solves a specific problem.
2. Create wireframes and prototypes for the user interface.
3. Implement core functionality using appropriate Swift features.
4. Test the app thoroughly on multiple device sizes.
5. Present and explain the technical implementation details.

Course Standards: Career Connected Learning I and II

Career connected learning is an educational approach that integrates classroom instruction with real-world experiences, enabling high school students to explore potential careers and develop relevant skills before graduation. By participating in work-based learning opportunities—such as apprenticeships, internships, capstone projects, and school-based enterprises—students apply academic concepts in authentic settings, gain practical industry knowledge, and build professional networks. This hands-on engagement helps students connect their studies to future career paths, strengthens their problem-solving and communication skills, and supports a smoother transition into college, vocational programs, or the workforce.

All Career and Technical Education Programs of Study include aspects of work-based learning, and almost all of the programs include two Career Connected Learning (CCL) courses. Below are the course descriptions for CCL I and CCL II. The CCL standards can be found via this link: