Massachusetts Department of Elementary & Secondary Education

Office for College, Career and Technical Education

Vocational Technical Education Framework

Information Technology Services Occupational Cluster

Programming & Web Development

CIP Code 110201

August 2022

Massachusetts Department of Elementary and Secondary Education
Office for College, Career, and Technical Education
75 Pleasant Street, Malden, MA 02148-4906
781-338-3910
www.doe.mass.edu/ccte/cvte/

Strand 2: Technical Knowledge and Skills

Computer Information Technology Fundamentals

Demonstrate safety and health in a computer environment.

List, define and demonstrate workplace ergonomics that reduce discomfort and strains while increasing productivity and efficiency.

Identify, list, and apply personal safety practices and precautions when working with electronic circuitry.

Define the effects of Electrostatic Discharge (ESD) on Printed Circuit Boards (PCBs) and demonstrate the proper use of wearing an electrostatic wrist strap while handling circuit boards.

Demonstrate safety practices and procedures associated with "Power strips", "Surge/Spike Protectors", or "Portable Outlets".

Performance Example:

Students will create a presentation on an aspect of computer environment safety.

Students will demonstrate the proper procedure for plugging in their laptop or desktop.

Explain the role of software applications in the workplace and community.

Describe the history of the software industry, including the contributions of traditionally underrepresented groups.

Research and classify professional certifications.

Analyze the harmful and beneficial impacts that the use software applications have on society.

Describe ways that software can make workplaces and communities more equitable.

Research and identify the harmful and beneficial uses of social media in workplaces and communities.

Research and identify uses of new and evolving technologies affecting the industry.

Explain the impact of the digital divide on access to critical information.

Research and identify the attributes of a digital footprint and its extent and impact on one's reputation.

Performance Example:

Students will create a digital artifact on an aspect of social media and its effect in the workplace and community.

Students will research current job postings and list the specific certifications and qualifications for the position.

Students will identify and describe emerging technologies from businesses within their communities, including traditionally underrepresented and untapped groups.

Given that facial recognition systems have been proven to misrecognize people of color, explain the effect that Artificial Intelligence systems can have on racially incomplete data.

Students will create a presentation on the uses of a new or evolving technology such as Artificial Intelligence, Big Data, Cybersecurity, Game Development, Cloud Technology, Robotics, Internet of Things, Block Chain and Extended Reality.
Students will create a presentation on the ways a specific software application can make workplaces and communities more equitable.

Cybersecurity Fundamentals

Research and identify Cybersecurity topics.

Identify security principles, vulnerability, and threats in software.

Identify security principles, vulnerability, and threats in network infrastructure.

Analyze threat maps to identify cyberattack targets and origins.

Explain what secure systems are from a Confidentiality, Integrity, availability (CIA triad) perspective.

Research and identify components of access control such as Identification, authentication, and authorization.

Identify and demonstrate the basics of cryptography.

Research and describe penetration testing methods in the context of ethical hacking.

Research and describe topics in cryptology, cryptography, and cryptanalysis.

Performance Example:
Students will create a presentation describing the evolving national and international legal issues related to software security.
Students will encrypt and decrypt a data file.
Students will create a presentation outlining common network vulnerabilities (e.g., cyberattacks, identity theft, privacy) and their associated responses.

Describe and classify computer hardware.

Classify hardware components, their functions, and relationships.

Analyze different hardware components required for software applications for a specific use.

Demonstrate programming a physical computing device.

Performance Example:
Students will identify the location of specific hardware components. Students will research and present the hardware requirements for a specific software application such as: Artificial Intelligence, Big Data, Cybersecurity, Game Development, Cloud Technology, Robotics, Internet of Things, Block Chain and Extended Reality
Performance Example:
Students will identify the location of specific hardware components. Students will research and present the hardware requirements for a specific software application such as: Artificial Intelligence, Big Data, Cybersecurity, Game Development, Cloud Technology, Robotics, Internet of Things, Block Chain and Extended Reality.
Students will control a robot or design an application using a microcontroller.

Explain concepts fundamental to networking.

Describe the purpose of a network.

Identify and explain the functions of network hardware components.

Explain cloud computing and its components. Explain how network topologies and protocols enable users, devices, and systems to communicate with each other.

Research and identify server-side networking software components. (Firewalls, Domain Name System (DNS), Visual Private Network (VPN), Proxy Servers, Dynamic Host Configuration Protocol (DHCP), File Transfer Protocol (FTP), Data Center (DC), and Active Directory (AD)

Identify security risks associated with storing information on network devices.

Performance Example:

Students will explain the different network classifications and terms such as Local Area Network (LANs), Metropolitan Area Network (MANs), Wide Area Network (WANs) and clouds.

Students will create a presentation demonstrating the use of multifactor authentication.

Students will propose a resolution to a problem with access control.

Students will create a presentation outlining the issues (e.g., latency, bandwidth, firewalls, server capability) that impact network functionality.

Operating System Fundamentals

Describe the purpose of an operating system.

Differentiate between desktop, server, and mobile device operating systems.

Differentiate between a Command Line Interface and a Graphical User Interface.

Demonstrate the ability to navigate a file system at the Command Line Interface.

Identify the content associated with different file types and why different file types exist (e.g., formats for word processing, images, video, sound, and three-dimensional objects.).

Research and compare data compression algorithms and their applicability to different contexts.

Performance Example:

Students will create a digital artifact recommending an operating system based on user needs.

Students will develop a script which demonstrates file handling.

Students will create a digital artifact comparing different operating systems.

Data Management Fundamentals

Research and identify data management concepts.

Explain the purpose of data management systems.

Demonstrate the planning, designing and creation and use of a relational database.

Research and describe the relationship between tables such as one-to-one and one-to-many.

Explain the purpose of data analysis.

Demonstrate the creation of database queries and the generation of data reports and visualization.

Performance Example:
Students will create a customer database with two tables (customer info and customer sales) and generate a report of total sales per customer.
Students will utilize an existing data set and perform meaningful data mining operations.
Students will create the database structure necessary to implement a library checkout system.
Students will use a programming language to cleanse, visualize and analyze data.

Software Development Life Cycle (SDLC)

SDLC Fundamentals

Explain the life cycle of software development.

List, describe, and classify different methodologies of project development.

Research and identify the major roles within a project team.

Performance Example:
Students will develop a problem statement.
Students will prepare a digital artifact that outlines the process of an SDLC methodology.
Student will investigate and outline how software is maintained following its release.
Students will create a digital artifact which compares Agile vs waterfall methodologies.

Determine requirements.

Collaborate with potential users or customers to determine software requirements.

Produce a prioritized list of features to meet the functional software requirements.

Research and identify appropriate technologies to meet software requirements.

Evaluate project resources and create a time and cost analysis.

Define and structure a project scope of work.

Performance Example:
Students will prepare and maintain use case requirements based on customer needs identified through interviews and research.
Students will develop a problem statement.
Students will identify their Minimum Viable Product (MVP) and future incremental sprints.
Students will prepare and role-play a requirements gathering meeting.

Demonstrate Project Planning

Determine project objectives and compose a project vision statement.

Identify stakeholders and explain their roles.

Consider project priorities and estimate solution timeframes.

Identify and document the required technical skills for the project.

Identify and document the personnel and technology required for the project.

Performance Example:
Students will role-play as customers and developers for a project.
Students will develop a project timeline based on a pre-determined set of project priorities.

Incrementally Design, Implement and Test Software

Demonstrate team collaboration during software development.

Create a technical design document.

Devise pseudocode algorithms for software to be developed.

Implement the design as a coded solution.

Identify various software testing techniques.

Create and implement a test plan to ensure the software meets project requirements.

Demonstrate coded solution to the customer, ensuring that all requirements have been met.

Document lessons learned.

Performance Example:
Students will articulate and document the theme of this sprint.
Students will document and implement a testing procedure for an application.
Students will participate in a sprint demonstration meeting.
Students will prepare and demonstrate a project proposal for a potential customer.
Students will create a test plan for a given application.

Create user software documentation.

Document how the end user should run the software.

Performance Example:
Students will prepare a multimedia tutorial describing how the user is to utilize the software.

Publish and maintain software.

Describe the purpose of source code management.

Demonstrate version control methods to maintain source code.

Describe the purpose of release management.

Describe the purpose of Continuous Integration and Continuous Deployment (CI/CD).

Describe the purpose of the different development environments (development, staging, production).

Publish the software solution.

Performance Example:
Students will describe the considerations involved with deploying a 3-tier application.
Students will list steps necessary to distribute a program on a given platform.
Students will use version control software to create a project repository.
Students will work collaboratively with local and remote repositories using branching, checkouts, etc. to manage code.

User Interface and User Experience Fundamentals

Accessibility

Research and identify the reasons for making software accessible.

Research and identify methods for implementing software accessibility.

Performance Example:

Students will create a digital artifact including software accessibility implementation strategies such as font size, color profile, text-to-speech, alternative metadata, and keyboard shortcuts.

Students will build an accessible data table.

User Experience Design

Document the user goals, needs, behaviors, and preferences.

Develop personas, research demographics, and identify user types for software.

Develop user experience models that demonstrate an understanding of different audiences.

Research and define where the software will provide feedback on defaults, status, error, validation, and instructions to the user.

Research and identify user research methodologies such as usability tests, user interviews, contextual inquiries, and user observations.

Performance Example:

Students will gather data and design three personas for typical users of a software or website.

Students will list strategies to simplify and enhance user experience for one device.

Students will create interview questions and conduct a real or simulated usability test for a software project.

User Interface Design

Describe design considerations for different display form factors.

Define different user interface modalities.

Create a simplified interface layout for part of a software application using an appropriate graphic design tool.

Build a storyboard of the user interface for a software application.

Create a prototype from an interface layout and storyboard.

Performance Example:

Students will create a slide presentation including simple graphics (circles, squares, arrows, etc.) representing objects and movement in a game scene.

Students will compare and contrast the differences between user experience and user interface design.

Students will assemble and justify an application wireframe.

Programming Fundamentals

Implement concepts fundamental to programming.

Research and describe what programs are and how they run.

Research and describe the four parts of computational thinking: Decomposition, Abstraction, Pattern Recognition, and Algorithm Design.

Utilize Integrated Development Environments.

Demonstrate the use of arithmetic operators.

Demonstrate the use of a debugger.

Demonstrate the use of logical operators.

Demonstrate the use of relational operators.

Demonstrate the use of compound conditions.

Demonstrate the use of conditional branching operators.

Demonstrate iterative loops.

Demonstrate user defined functions, methods, or procedures with and without return values including parameter passing.

Implement comments, formatting conventions, and naming conventions.

Differentiate between different types of programming paradigms.

Differentiate between compilers and interpreters and describe their role in the use of programming languages.

Differentiate between and demonstrate the use of variable type declarations and casting in various programming languages.

Demonstrate the creation and use of fixed length data structures such as arrays.

Demonstrate the creation and use of variable length data structures such as lists.

Demonstrate the use of functions, methods or procedures from system and shared libraries.

Differentiate between and demonstrate the use of sorting algorithms.

Differentiate between and demonstrate the use of searching algorithms.

Demonstrate data validation and error handling.

Demonstrate the use of file input and output.

Describe concepts of memory management.

Describe design patterns such as Model, View, Controller (MVC).

Performance Example:

Students will create programs using fundamental data types and logical, relational, and arithmetic operators.

Students will create programs using lists and arrays.

Students will create programs using built-in and user defined functions, methods, or procedures.

Students will create an interactive application using formatting conventions and fundamental programming concepts, that require data input, validation, and output.

Object Orientated Programming (OOP) Fundamentals.

Design a user defined class incorporating attributes, constructors, and methods.

Develop constructors.

Demonstrate object instantiation.

Demonstrate concepts of composition using a modeling language such as Universal Modeling Language (UML).

Demonstrate concepts of inheritance.

Demonstrate concepts of encapsulation.

Define or demonstrate concepts of polymorphism.

Define or demonstrate concepts of overloading.

Define or demonstrate concepts of overriding.

Demonstrate the creation and use of accessor and mutator methods within a class.

Performance Example:
Students will design a class modeling real-world objects.
Students will describe the relationship between objects using a modeling language such as UML.
Student will develop a program using a student developed class and implement some of the following OOP concepts: overloading, polymorphism, encapsulation, overriding, and inheritance.

Web Design and Development Fundamentals

Front-end Web Development Fundamentals

Research and identify components of the current Hyper Text Markup Language (HTML) standard for identification, processing, and presentation of information on web pages.

Research and identify current front-end tool kits used in industry and their role in web development.

Research and identify current web standards bodies.

Demonstrate creation of a static web page using the current HTML standard.

Demonstrate the use of the current Cascading Style Sheets (CSS) standard for style selectors and declarations including id, class, nested selectors, multiple selectors, and pseudo selectors.

Demonstrate the manipulation of elements in a web page using the box model.

Demonstrate various element positioning methods such as the CSS position property.

Research and identify current web page layout models.

Demonstrate a responsive web page using a fluid design and media queries.

Identify page loading efficiency best practices.

Demonstrate implementing a web page using a front-end toolkit.

Demonstrate creating and submitting a form.

Demonstrate client-side form validation.

Incorporate web media objects into a web page.

Demonstrate how to manipulate the Document Object Model (DOM) with a scripting language.

Demonstrate the use of scripting language events and event handlers.

Demonstrate synchronous and asynchronous scripting language calls.

Describe the use of cookies in web development including privacy and security concerns.

Research and identify current trends in Search Engine Optimization (SEO) techniques.

Performance Example:
Students will validate webpages using an industry standard validation tool.

Students will create a static website for a community organization demonstrating web accessibility best practices.
Students will create a responsive website for a business demonstrating web accessibility best practices.

Back-end Web Development Fundamentals

Research and identify the boundaries between and roles of front-end and back-end web development.

Compare and contrast various Hypertext Transfer Protocol (HTTP) request methods.

Demonstrate processing form data.

Develop a website implementing a server-side scripting language.

Demonstrate the ability to read from and write to a database server from a web page.

Describe Create Read Update Delete (CRUD) actions and identify examples.

Develop a website incorporating authentication and authorization.

Compare and contrast Application Program Interface (APIs) that can be used for e-commerce.

Compare and contrast website hosting options.

Deploy a website to a web server.

Develop a web application that utilizes data interchange requests such as Java Script Object Notation (JSON).

Compare and contrast pre-built Content Management Systems (CMS).

Demonstrate the use of a pre-built CMS.

Compare and contrast various web back-end framework.

Develop a website using a web framework

Performance Example:
Students will develop a web application tied to a database using version control.
Students will develop a website using a CMS such as WordPress incorporating customer requirements.