COMPUTER PROGRAMMING (.NET TECHNOLOGY)

TECHNICAL-PROFESSIONAL

Grade 12

Three-Term Budget Of Work (BOW)
for Learning Competencies

TECHNICAL-PROFESSIONAL
ICT SUPPORT AND COMPUTER PROGRAMMING TECHNOLOGIES
Grade 12

Three-Term Budget Of Work
For Learning Competencies

Please note that each Tech-Pro elective is completed in one term if delivered at Grade 12.

Elective	**Computer Programming (.Net Technology)**
Prerequisite	**None**
Performance Standard	The learners create a web portfolio using Hyper Text Mark-Up Language (HTML).
Content Standard	The learners demonstrate an understanding of fundamental concepts and principles of syntax and elements in Hyper Text Mark-Up Language (HTML) and the properties and values used in Cascading Style Sheet (CSS) in developing and designing a website.

Week	Learning Competency	Suggested Activities
1	• **Discuss the HTML fundamentals**	• **HTML Time Traveler: Build the Web Timeline** o Explore HTML versions, roles, and web editors by creating a simple interactive webpage timeline. o Students research HTML versions, explain the role of HTML in web development, and use a web editor to create a webpage timeline showcasing the evolution of HTML.

Week	Learning Competency	Suggested Activities
		• **Code & Create: My First Web Page Challenge** o One-liner Objective: Apply HTML fundamentals by designing a basic webpage using a web editor. o How: Students use a web editor to build a personal webpage that demonstrates basic HTML structure while explaining the purpose and importance of HTML in modern websites.
	• **Use paragraph formatting elements**	• **Paragraph Power: Text Styling Challenge** o Apply HTML paragraph formatting elements to create clear and visually organized webpage text. o Students create a webpage using paragraph tags and formatting elements such as bold, italic, underline, line breaks, and alignment to enhance text presentation. • **Web Writer's Studio** o Use HTML paragraph formatting elements to design an attractive and readable content page. **o** Students design a short article webpage using different paragraph formatting tags to improve readability, emphasis, and layout.

Week	Learning Competency	Suggested Activities
	• **Apply list attributes and values in web page designs**	• **List Master: Organize the Web** o Apply HTML list types, attributes, and values to create organized webpage content. o Students create a webpage using ordered, unordered, and descriptive lists, then customize appearance using different list attributes and values. • **Smart Menu Maker** o Use HTML lists with proper attributes and values to design a clear and attractive web menu. o Students design a navigation menu or checklist webpage using various HTML lists and modify styles through attributes and values for better presentation.
	• **Create HTML documents with multimedia elements**	• **Media Magic: Bring Webpages to Life** o Apply HTML multimedia elements by embedding images, videos, and audio into a webpage. o How: Students create a themed webpage that includes an image gallery, embedded video, and background or playable audio using proper HTML tags.

Week	Learning Competency	Suggested Activities
		• **Interactive Showcase: My Multimedia Page** o Use HTML multimedia features to design an engaging and interactive web page. o Students design a personal or promotional webpage using images, video clips, and audio elements to enhance user experience and presentation.
	• **Use table elements, attributes, and values in designing web pages**	• **Table Titan: Data Grid Challenge** o Apply HTML table elements, attributes, and values to organize information in a structured webpage layout. o Students create a webpage table containing sample data, then enhance it using rows, columns, borders, spacing, alignment, and other table attributes. • **Schedule Maker: Web Table Designer** o Use HTML table elements and attributes to design a clear and attractive schedule or planner page. o Students build a class schedule or event planner using HTML tables and customize it with proper attributes and values for readability.
	• **Create hyperlinks between web**	• **Link Quest: Connect the Web**

Week	Learning Competency	Suggested Activities
	pages	o Apply HTML link tags to connect webpages and online resources. o Create a webpage with text and image links that open internal pages and external websites. • **Navigation Builder"** **o Use HTML links to design a simple website navigation menu.** **o Build a multi-page mini website using hyperlinks to connect each page.**
	• **Use frames in designing web pages**	• **Frame Fusion: Multi-View Web Design"** o Apply HTML frames to organize multiple webpage contents within a single browser window. o Create a framed webpage with separate sections for navigation, content, and header. • **Split Screen Web Builder"** o Use HTML frames to design a structured and easy-to-navigate webpage layout. **o** Build a webpage using frames to display linked pages simultaneously in different sections.
2	• **Create HTML form web pages and provide fields where user can enter data**	• **Form Factory: Build a Smart Input Page"** o Apply HTML form elements to collect user information through a webpage. o Create an HTML form with text fields, buttons,

Week	Learning Competency	Suggested Activities
		checkboxes, and dropdown menus for user input. • **Survey Creator Challenge"** o Use HTML forms to design an interactive online survey or registration page. **o** Build a simple survey or registration form using different HTML input elements and submit buttons.
	• **Perform installation of Integrated Development Environment (IDE) application**	• **IDE Setup Sprint"** o Apply system requirements and installation procedures to successfully set up an Integrated Development Environment (IDE). o Check computer specifications and install the assigned IDE following the correct setup procedures. • **Tech Ready: Development Environment Challenge"** o Demonstrate proper IDE installation and configuration for programming development. o Install and configure an IDE, then document the system requirements and installation steps performed.
	• **Discuss Cascading Style Sheet (CSS) fundamentals**	• **Style Surge: CSS Makeover Challenge** o Apply CSS fundamentals by using proper

Week	Learning Competency	Suggested Activities
		syntax to enhance webpage design and appearance. o Create a simple webpage and apply CSS styles to change colors, fonts, spacing, and layout. • **Design It with CSS** o One-liner Objective: Demonstrate understanding of CSS definition, role, and syntax through webpage styling. o How: Develop a styled webpage using basic CSS rules and explain how CSS improves web presentation.
	• **Apply the different CSS Selectors**	• **Selector Showdown: Class vs ID"** o Apply CSS class and ID selectors to style webpage elements effectively. o Create a webpage using class and ID selectors to apply different styles to specific elements. • **Style Target Challenge** o Use CSS selectors to control and customize webpage design. o Design a webpage that demonstrates the proper use of class and ID selectors for formatting text and layout.

Week	Learning Competency	Suggested Activities
Suggested Performance Tasks	• **Individual Performance Task: Web Developer Bootcamp: From HTML to CSS Mastery** ○ **Apply HTML fundamentals, multimedia, tables, links, frames, forms, IDE setup,** and CSS selectors to create a fully functional styled website. o Install and configure an IDE, then create a multi-page website using HTML elements (paragraphs, lists, multimedia, tables, links, frames, and forms) and enhance its design using CSS fundamentals and selectors (class and ID). • **Individual Performance Task: Mini Web Agency Challenge: Build, Style, and Deploy** o Collaboratively design and develop a complete interactive website using HTML structure and CSS styling principles. o In groups, act as a web development team to plan, build, and style a website using HTML fundamentals, multimedia, tables, links, frames, forms, and CSS selectors, then present the finished project.	
Performance Standard	The learners create a web portfolio applying Cascading Style Sheet (CSS) rules.	
Content Standard	The learners demonstrate an understanding of fundamental principles, syntax, and the properties and values of Cascading Style Sheet (CSS) rule in designing and developing interactive webpages.	
2	• **Apply CSS text, font, and color in designing web pages**	• **Style Studio: Text Makeover Challenge** o Apply CSS text, font, and color properties to enhance webpage readability and design. o Create a webpage and use CSS to change text styles, font families, sizes, and colors to improve visual presentation**.** • **Brand Page Builder** o Use CSS typography and color styling to

Week	Learning Competency	Suggested Activities
		design a themed and visually appealing webpage. o Design a simple brand or profile webpage and apply consistent fonts, text formatting, and color schemes using CSS**.**
	• **Apply CSS gradient and opacity**	• **Gradient Glow-Up Challenge"** o Apply CSS gradients and opacity to create visually appealing webpage designs. o Design a webpage section using gradient backgrounds and opacity effects for images, text, or containers. • **Transparent Trends: Modern Web Styling** o Use CSS gradient and opacity properties to enhance webpage depth and visual effects. o Create a themed webpage applying linear or radial gradients and opacity settings to improve design **presentation.**
	• **Create layouts of web pages using the CSS box model**	• **Boxed Up: The CSS Layout Challenge** o Apply the CSS box model to organize webpage elements using margin, border, padding, and content properties. o Create a webpage layout and style different sections using box model properties for

Week	Learning Competency	Suggested Activities
		spacing and alignment. • **Spacing Master: Build the Perfect Web Layout** o Use the CSS box model to create balanced and visually structured webpage designs. **o** Design a simple webpage and adjust margins, padding, borders, and sizes to improve layout presentation.
3	• **Use the CSS list and link properties and values in creating web pages**	• **Link & List Designer Challenge** o Apply CSS list and link properties to create organized and interactive webpage navigation. o Create a webpage menu using lists and style links with colors, hover effects, and formatting using CSS. • **Smart Navigation Builder** o Use CSS to enhance lists and hyperlinks for a user-friendly web interface. **o** Design a navigation page using styled lists and customized links for better readability and interaction.
	• **Utilizes CSS Z-index, position, float properties and values in designing web pages**	• **Gallery Grid Genius** o Apply CSS techniques to design a structured and visually appealing image gallery layout.

Week	Learning Competency	Suggested Activities
		o Create an image gallery webpage using CSS grid or flexbox to properly arrange and style images. • **Visual Showcase Builder** o Use CSS styling to create an interactive and responsive image gallery webpage. o Design a themed gallery with images and apply CSS effects such as hover, spacing, and alignment for presentation.
	• **Create web pages with a CSS image gallery**	• **Hover Hero: Interactive Style Challenge** o Apply CSS pseudo-classes to create interactive and responsive webpage elements. o Create a webpage and use pseudo-classes like :hover, :active, and :focus to add interactive effects to buttons, links, and text elements. • **Detail Enhancer: Pseudo Magic Web Design** o Use CSS pseudo-elements to enhance webpage content and visual presentation. o Design a webpage and apply ::before and ::after pseudo-elements to add decorative content and improve styling without changing the HTML structure.
	• **Apply CSS pseudo class and elements**	• **Hover Hero: Interactive Style Challenge** o Apply CSS pseudo-classes to create interactive and responsive webpage elements.

Week	Learning Competency	Suggested Activities
		o Create a webpage and use pseudo-classes like :hover, :active, and :focus to add interactive effects to buttons, links, and text elements. • **"Detail Enhancer: Pseudo Magic Web Design"** o Use CSS pseudo-elements to enhance webpage content and visual presentation. o Design a webpage and apply ::before and ::after pseudo-elements to add decorative content and improve styling without changing the HTML structure.
	• **Apply CSS flexbox layout and media queries in web designs**	• **Flex Master: Responsive Layout Builder** o Apply CSS Flexbox to create flexible and well-aligned webpage layouts. o Create a webpage layout using Flexbox to arrange items in rows and columns with proper alignment and spacing. • **Responsive Web Hero: Mobile-Ready Design Challenge** o Use CSS Flexbox and media queries to design a responsive webpage for different screen sizes. o Design a webpage and apply media queries to adjust layout, font size, and structure for mobile, tablet, and desktop views.

Week	Learning Competency	Suggested Activities
	• **Create responsive websites using CSS Responsive Web Development (RWD) Layouts**	• **Responsive Grid Master: Build the Adaptive Page** o Apply RWD principles using CSS grid, media queries, images, and videos to create a responsive webpage layout. o Design a webpage using grid view and media queries, integrating properly sized images and videos that adjust across different screen sizes. • **Mobile-First Web Designer: Framework & Template Challenge** o Use responsive web design frameworks and templates to build a mobile-friendly website layout. o Create a responsive webpage using a CSS framework or template, integrating images, videos, and layout adjustments for different devices.
	• **Discuss the fundamentals of JavaScript**	• **JS Time Capsule: Code Through History** o Understand the history, versions, and basic syntax of JavaScript through simple webpage integration. o Create a webpage presenting JavaScript history and versions, and include sample scripts demonstrating syntax and comments.

Week	Learning Competency	Suggested Activities
		• **First Script Lab: Hello JavaScript** o Apply JavaScript syntax and comments to create a basic working script in a webpage. o Write a simple JavaScript program embedded in HTML that uses correct syntax and comments to display output or messages.
	• **Demonstrate the structures of statements, algorithm, flowchart, and pseudocode**	• **Logic Builder: From Idea to Flow** o Apply statements, algorithms, flowcharts, and pseudocode to design clear program logic. o Create a step-by-step algorithm, convert it into pseudocode, and draw a flowchart for a given problem, then express it using basic programming statements. • **Code Planner Challenge: Visual to Text Logic** o Use flowcharts, pseudocode, and algorithms to develop structured problem-solving solutions. o Design a solution for a real-life problem by creating a flowchart, writing pseudocode, and translating it into logical programming statements.
	• **Apply variables, data types, and functions in creating web applications**	• **Code Starter: Variables in Action** o Apply variables, data types, and functions to build simple working JavaScript programs.

Week	Learning Competency	Suggested Activities
		o Create a program that uses different variables and data types, then define and call functions to process and display results. • **Function Builder Challenge** o Use variables, data types, and functions to solve basic computational problems. **o** How: Design a JavaScript program that accepts inputs using variables, applies correct data types, and processes them using functions to produce output.
Suggested Performance Tasks	• **Individual Performance Task: "Full Stack Front-End Builder: Design to Logic Challenge"** o Apply CSS styling, responsive design, and JavaScript fundamentals to create a fully functional and interactive web application. o Design a complete responsive website using CSS (text, fonts, colors, gradients, box model, lists, links, positioning, flexbox, grid, pseudo-classes, image gallery, and media queries). Then integrate JavaScript basics using variables, data types, functions, and apply structured problem-solving using algorithm, pseudocode, flowchart, and statements. • **Group Individual Performance Task: "Web Development Studio: Build, Style, and Program"** o Collaboratively design and develop a responsive, interactive website integrating CSS design systems and JavaScript logic. o In groups, plan and build a responsive website using CSS (layout, styling, flexbox, grid,	

Week	Learning Competency	Suggested Activities
	RWD, and effects) and enhance it with JavaScript using variables, data types, and functions. Document the system logic using flowcharts, pseudocode, and algorithms, then present the working prototype.	
Performance Standard	The learners create a web application project using JavaScript and C# programming language.	
Content Standard	Demonstrate an understanding of the principles of web responsive development by using JavaScript functions and C# programming language.	
4	• **Use JavaScript operators in problem solving**	• **Operator Arena: JS Math Challenge** o Apply JavaScript operators to perform calculations and solve logical expressions in web programs. o Create a simple JavaScript program that uses arithmetic, comparison, and logical operators to compute results and display outputs on a webpage**.** • **Decision Maker: Interactive JS Logic Builder"** o Use JavaScript operators to build interactive decision-making web applications. o Design a webpage that uses comparison and logical operators to evaluate user inputs and display different outcomes based on conditions.
	• **Use event handlers and event listeners in creating web**	• **Event Trigger Lab: Make the Web React"** o Apply JavaScript events to create interactive

Week	Learning Competency	Suggested Activities
	applications	and responsive web elements. o Create a webpage with buttons, inputs, and images that respond to events like click, hover, and keypress using JavaScript. • **Interactive Action Page: JS Event Designer** o Use JavaScript events to build an interactive user-driven webpage. o Design a webpage that changes content or style based on user actions such as clicking buttons or typing in input fields**.**
	• **Use conditional and switch statements in creating web applications**	• **Logic Path Explorer: If-Else Adventure** o Apply JavaScript control flow to make decisions using conditional statements in web applications. o Create a program that uses if-else and switch statements to evaluate user input and display different outcomes on a webpage**.** • **Decision Engine: Smart Web Response System** o Use control flow structures to build interactive, decision-based web functionality. o Design a webpage that responds differently to user actions or inputs using nested conditions and logical branching.

Week	Learning Competency	Suggested Activities
	• **Apply loops in creating a program using while, do while, and for loops**	• **Loop Lab: Repeat & Render** o Apply JavaScript loops to generate repeated outputs and dynamic webpage content. o Create a webpage that uses for and while loops to display repeated numbers, text, or list items dynamically. • **Iteration Builder: Smart Repetition Challenge** o Use JavaScript loops to automate repetitive tasks in web applications. o Design a webpage that uses loops to generate tables, menus, or patterns based on user input.
5	• **Use JavaScript arrays in creating a program**	• **Array Adventure: Data Organizer Challenge"** o Apply JavaScript arrays to store, manage, and display multiple data values in a webpage. o Create a webpage that uses arrays to organize and display lists such as names, products, or scores dynamically. • **Smart List Builder: Array in Action** o Use JavaScript arrays to create interactive and dynamic web content. o Design a webpage that allows users to add, remove, or display items stored in an array

Week	Learning Competency	Suggested Activities
		using JavaScript.
	• **Create objects to display the set of properties and values**	• **Object Explorer: Build Your Digital Profile** o Apply JavaScript objects to organize and manage related data in web applications. o Create a webpage that uses JavaScript objects to store and display information such as profiles, products, or inventory details. • **Smart Object Creator: Interactive Data Challenge** o Use JavaScript objects to create dynamic and interactive webpage content. **o** Design a webpage where users can input data that will be stored and displayed using JavaScript object properties and methods.
	• **Apply Document Object Model (DOM) nodes in creating a web document**	• **DOM Master: Control the Web Page** o Apply DOM node manipulation to dynamically change webpage content and structure. o Create a webpage that uses JavaScript DOM methods to add, edit, or remove elements based on user interaction**.** • **Interactive Node Builder** o Use DOM nodes to create responsive and interactive webpage features.

Week	Learning Competency	Suggested Activities
		o Design a webpage that dynamically creates and updates text, images, or list items using DOM node manipulation techniques.
	• **Discuss the C# fundamentals**	• **C# Origins to Output: First Code Challenge** o Understand the history, features, and basic structure of C# through simple program creation. o Create a basic C# program demonstrating proper syntax and structure, then present key features and milestones of C# development. • **Build Your First C# App** o Apply C# fundamentals to write and execute a simple working program. o Develop a simple console application using correct C# syntax, program structure, and comments explaining its features.
6	• **Use C# data types, variables, strings, and constants in creating a web application**	• **Data Detective: C# Variable Challenge** o Apply C# data types, variables, strings, and constants in creating basic console programs. o Create a C# program that declares variables, uses different data types, manipulates strings, and applies constants in calculations or outputs. • **String & Variable Builder**

Week	Learning Competency	Suggested Activities
		o Use C# variables, strings, and constants to develop interactive program outputs. o Design a console application that accepts user input, stores data in variables, processes strings, and displays formatted results.
	• **Apply C# operators and decision making in problem solving**	• **Decision Engine: C# Logic Builder** o Apply C# operators and decision-making statements to control program flow. o Create a C# console program that uses arithmetic, relational, and logical operators with if-else or switch statements to make decisions based on user input. • **Smart Choice System: If-Else Challenge** o Use C# operators and conditional statements to build interactive decision-based applications. o Design a program that evaluates user inputs using operators and displays different outputs using nested or multiple decision-making conditions
	• **Use the C# iteration statements in creating a web application**	• **Loop Lab: Repeat the Logic** o Apply C# loops to automate repetitive tasks in console applications.

Week	Learning Competency	Suggested Activities
		o Create a C# program that uses for, while, or do-while loops to display repeated outputs such as numbers, patterns, or lists. • **Iteration Challenge: Smart Counter System** o Use C# iteration structures to solve problems involving repeated computations. o Design a program that uses loops to calculate totals, generate sequences, or process multiple user inputs dynamically.
7	• **Apply C# classes, methods and structure in creating a web application**	• **Class Creator: Build Your First C# Blueprint** o Apply C# classes, methods, and structures to organize and execute program logic. o Create a C# program that defines a class with attributes and methods, then use a structure to organize related data and display outputs. • **Method Master: Object-Oriented Challenge** o Use C# classes, methods, and structures to build modular and reusable programs. o Design a console application that uses multiple methods inside a class and a structure to handle and process user-defined data.
	• **Create web application programs that contains C# arrays**	• **Array Explorer: Data in Motion** o Apply C# arrays to store, process, and display multiple values in a structured program.

Week	Learning Competency	Suggested Activities
		o Create a C# program that uses arrays to store a list of values (e.g., names, scores, or items) and display them using loops. • **Smart Array System: Dynamic Data Handler** o Use C# arrays to manage and manipulate collections of data in a console application. o Design a program that allows input of multiple values into an array, then processes and outputs result such as sum, average, or search.
	• **Apply the C# Object-Oriented Programming (OOP) approach in creating a web application**	• **OOP Builder: Code Your Smart System** o Apply C# OOP concepts (encapsulation, inheritance, polymorphism, and interfaces) to design structured and reusable programs. o Create a C# program that defines classes using encapsulation, extends functionality through inheritance, implements interfaces, and demonstrates polymorphism through method overriding**.** • **Object-Oriented Challenge: Real-World Simulation** o Use C# OOP principles to model real-world systems with reusable and flexible code.

Week	Learning Competency	Suggested Activities
		o How: Design a console application that simulates a real-world scenario (e.g., school system or transport system) using classes, interfaces, inheritance, encapsulation, and polymorphism.
	• **Utilize the C# file input/output in creating, deleting, and reading web applications**	• **File Flow: Save & Retrieve Challenge** o Apply C# File Input/Output to read and write data using text files in console applications. o Create a C# program that writes user input to a file and retrieves the saved data for display using file handling methods. • **Data Logger: File Manager System** o Use C# File I/O to store, update, and retrieve structured data from external files. o Design a program that logs multiple user entries into a file and reads them back for viewing or processing.
Suggested Performance Tasks	• **Individual Performance Task: "Full-Stack Logic Builder: JavaScript to C# Web App Challenge"** o Integrate JavaScript programming (operators, control flow, loops, arrays, objects, events, and DOM) with C# fundamentals, OOP, and file handling to build a functional web application. o Develop a web application using JavaScript for frontend interactivity (operators, events, loops, arrays, objects, DOM manipulation, and control flow) and C# for backend logic (variables, data types, operators, loops, classes, OOP concepts, and file I/O for data	

Week	Learning Competency	Suggested Activities
	storage and retrieval). • **Group Performance Task "Web Dev Studio: Build, Code, and Store System"** o Collaboratively design and develop a web-based system using JavaScript interactivity and C# backend programming with file management. o In groups, build a complete web application where JavaScript handles UI interaction (events, DOM, loops, arrays, objects, and control flow) while C# manages backend processing (OOP, file I/O, data types, classes, and decision logic). Present a working prototype system with stored and retrieved data.	
Performance Standard	The learners create a web application following the procedures of managing the list of tasks of Creating, Reading, Updating, and Deleting (CRUD) operations with a database backend, conversion of web to mobile application, and computation of service costing.	
Content Standard	Demonstrate an understanding of the principles of responsive web development by exploring features of ASP.NET Model View Controller architecture and back-end database, conversion of web to mobile application, and computation of service costing	

Week	Learning Competency	Suggested Activities
8	• **Discuss the ASP.NET Model View Controller (MVC) fundamentals**	• **MVC Explorer: Web App Blueprint Challenge** o Understand ASP.NET MVC fundamentals by identifying its structure, history, advantages, and pattern components. o Create a visual diagram or presentation explaining ASP.NET history, advantages, and the MVC pattern (Model, View, Controller) with real-world examples. • **Mini MVC Builder: First Web App Design** o Apply ASP.NET MVC concepts to design a simple structured web application. o Sketch or prototype a basic MVC-based web application showing how Model, View, and Controller interact in a simple system (e.g., student or product management).
	• **Discuss the ASP.NET MVC environment and life cycle**	• **MVC Environment Explorer: Build the Flow Map** o Understand the ASP.NET MVC environment and application/request life cycles through visual mapping. o Create a flowchart or diagram showing the ASP.NET MVC environment, highlighting the application life cycle and request life cycle steps.

Week	Learning Competency	Suggested Activities
		• **Life Cycle Simulator: Request to Response"** o Apply knowledge of ASP.NET MVC life cycles to trace how a web request is processed. o Illustrate a step-by-step scenario of a user request in an MVC application, showing how it moves through the application and request life cycle until a response is generated.
	• **Perform configurations of the routing process in directing an http request in an ASP.NET MVC project using controllers and actions**	• **Route Finder: MVC Navigation Challenge** o Apply ASP.NET MVC routing to understand how URLs connect to controllers and actions. o Create a routing map or diagram that shows how different URLs are directed to specific controllers and actions in an MVC application. • **Controller Action Builder: Mini Web Flow** o Use controllers and actions in ASP.NET MVC to simulate web request handling. o Design a simple MVC flow where user requests are handled by controllers and actions, then displayed as a step-by-step process diagram or prototype**.**
	• **Use filters, selectors, and views in creating projects**	• **View Master: Filter & Display Challenge** o Apply ASP.NET MVC filters, selectors, and views to control and display web application data. o Create a simple MVC page that uses filters and

Week	Learning Competency	Suggested Activities
		selectors to process data and display results through a view. • **Smart View Builder: MVC Presentation Flow** o Use MVC filters, selectors, and views to design structured and dynamic web page outputs. o Design a workflow showing how data is filtered, selected, and rendered in a view within an ASP.NET MVC application.
9	• **Apply the ASP.NET MVC - data model**	• **Model Builder: Data Structure Challenge** o Apply ASP.NET MVC data models to design structured and organized application data. o Create a simple data model for a real-world scenario (e.g., student or product system) and define its properties and relationships. • **MVC Data Mapper: From Model to View** o Use ASP.NET MVC data models to connect and display structured data in an application. o Design a basic MVC flow where a data model is created and its data is passed to a view for display.
	• **Use ASP.NET MVC – helpers in creating realistic HTML controls**	• **Helper Hero: Build Faster Views** o Apply ASP.NET MVC HTML helpers to simplify and speed up view creation.

Week	Learning Competency	Suggested Activities
		○ Create a simple MVC view using built-in HTML helpers (e.g., textboxes, labels, and forms) to generate user input pages. • **Smart Form Builder: Helper Toolkit Challenge** ○ Use ASP.NET MVC helpers to design clean and structured web forms. ○ Design a user registration or data entry form using MVC helpers to automatically generate input fields and validation elements.
	• **Create bridges between the http request and the action methods (Model Binding)**	• **Data Binder Lab: Connect the Form** ○ Apply ASP.NET MVC model binding to automatically map form inputs to data models. ○ Create a simple MVC form where user inputs are automatically bound to a model and displayed after submission. • **Smart Input System: Auto Data Mapping Challenge** ○ Use model binding in ASP.NET MVC to simplify data input and processing in web applications. ○ Design a data entry page that uses model binding to capture user information and display it in a view without manual assignment.

Week	Learning Competency	Suggested Activities
	• **Discuss the framework of ASP.NET MVC – database and adding validation to a model**	• **Data Guard: MVC Validation Shield** o Apply ASP.NET MVC validation and database integration to ensure accurate and secure data entry. o Create a simple MVC form connected to a database and apply validation rules (required fields, formats, and constraints) before saving data. • **Smart Records System: Database & Validation Builder** o Use ASP.NET MVC database connectivity and validation techniques to manage reliable data storage. o Design a CRUD-style web page that stores user input in a database with proper validation to ensure clean and accurate records.
10	• **Execute security features in a web application**	• **Secure Cache Builder: MVC Safety & Speed Challenge** o Apply ASP.NET MVC security, caching, and data annotations to build a secure and optimized web application. o Create an MVC form with data annotations for validation, implement basic security checks, and use caching to improve page performance.

Week	Learning Competency	Suggested Activities
		• **NuGet Power App: Smart MVC Enhancement Lab** o Use NuGet packages, data annotations, security, and caching to enhance an ASP.NET MVC application. o Develop a simple MVC project and integrate a NuGet package while applying validation, security rules, and caching for improved functionality.
	• **Create ASP.NET MVC - web application program interface (API)**	• **API Explorer: First RESTful Call** o Understand ASP.NET MVC Web API by creating and testing basic API endpoints. o Create a simple Web API that returns data (e.g., student or product list) and test it using a browser or API testing tool. • **Data Exchange Builder: Web API Mini System** o Apply ASP.NET Web API to enable data communication between client and server. o Design a simple API that sends and receives JSON data, simulating real-world data exchange in a web application.
	• **Convert website to mobile version**	• **Web-to-App Converter: Mobile Makeover Challenge**

Week	Learning Competency	Suggested Activities
		o Apply third-party tools to convert a website into a functional mobile application. o Select a simple website and use Appgeyser, Android Studio, or Web2APK to convert it into a mobile app prototype. • **Mobile App Builder: Web Transformation Lab** o Use web conversion tools to design and deploy a mobile version of an existing website. o Redesign a website for mobile use and convert it using a chosen platform, then test its mobile functionality.
	• **Compute the service costs for web development, web hosting**	• **Cost Cruncher: Web Project Budgeting Challenge** o Apply service costing concepts by identifying direct and indirect costs in web development projects. o Create a simple web project budget and classify all expenses as direct or indirect costs. • **Smart Pricing Planner: Web Dev Cost Breakdown** o Use costing analysis to compute and organize web development service expenses.

Week	Learning Competency	Suggested Activities
		o Design a cost breakdown sheet for a web project showing direct and indirect costs with brief explanations for each item.

Suggested Performance Tasks	• **Individual Performance Task: "Full ASP.NET MVC Web Solutions: From Build to Mobile Deployment"** o Apply ASP.NET MVC fundamentals, routing, controllers, views, models, security, APIs, and deployment concepts to build a complete web application and estimate its development cost. o Develop an ASP.NET MVC web application implementing routing, controllers, filters, models, helpers, model binding, validation, security, caching, and Web API, then convert it into a mobile version using a third-party tool and compute its direct and indirect development costs. • **Group Performance Task: "Web Dev Agency Simulation: Design, Build, Secure, Deploy & Cost"** o Collaboratively design and develop a full ASP.NET MVC system with deployment and cost analysis. o In groups, simulate a web development company by planning and building an MVC-based application with routing, models, views, APIs, security, and validation, converting it into a mobile app, and presenting a complete project with computed service costs (direct and indirect).