C# SYLLABUS LEARN PATH:

Data Types

Subtopic	Description	Reference Links
Introduction to Data Types	Overview of data types in C#. Importance of choosing the correct data type.	Introduction to Data Types
Value Types	Definition and characteristics of value types. Examples: Integer Types (int, long, short, byte), Floating-point Types (float, double), bool, char, struct.	Value Types in C#
Reference Types	Definition and characteristics of reference types. Examples: String (string), Arrays, Classes (class), Delegates, Interfaces.	Reference Types in C#
Nullable Types	Explanation of nullable types. Using the ? operator for nullable value types. Handling null values in value types.	Nullable Types in C#
Implicitly Typed Variables	The var keyword and its use. How var determines the type of a variable at compile time.	Implicitly Typed Variables
Enumerations (enum)	What is an enum and how to define one. Assigning values to enum members. Casting between enum and underlying data types.	Enums in C#
Object Type (object)	object as the base type of all other types. Boxing and Unboxing operations.	Object Type in C#
Type Casting	Implicit Casting (automatic conversion between compatible types). Explicit Casting (using (type) or Convert methods). Differences between implicit and explicit casting.	Type Casting in C#
Constants and Read-only Types	Declaring constant values using const. Declaring readonly fields and differences from const.	Constants and Readonly in C#
Default Values	Default values for different data types. Using default keyword with value types and reference types.	Default Values in C#

Data Type Conversion	Type conversion methods (e.g., Convert.ToInt32(), Parse(), TryParse()). Parsing strings to numeric types and handling exceptions.	Data Type Conversion in C#
Custom Data Types	Creating custom data types using struct and class. Differences between struct and class (value vs reference types).	Structs and Classes in C#
Built-in Data Types	Overview of common built-in types in C#. Examples of built-in types such as DateTime, decimal, Guid, etc.	Built-in Types in C#
Tuple Types	Defining and using tuples. Accessing tuple elements by name or position.	Tuple Types in C#
Dynamic Type	Understanding dynamic type. Differences between dynamic and object. Using dynamic for late binding.	Dynamic Type in C#
Arrays	Definition and use of arrays. Declaring, initializing, and accessing arrays. Multidimensional arrays. Jagged arrays.	Arrays in C#

Operators

Sub-topic	Description	Reference Link
Arithmetic Operators	Basic operators for performing arithmetic calculations.	Learn more
Assignment Operators	Used to assign values to variables.	Learn more
Comparison Operators	Used to compare two operands.	Learn more
Logical Operators	Perform logical operations on expressions.	Learn more
Bitwise Operators	Perform bit-level operations on binary representations of numbers.	Learn more

Unary Operators	Work with a single operand to return a result.	Learn more
Ternary Operator	A conditional operator that takes three operands.	Learn more
Null-coalescing Operators	Returns the left-hand operand if it's not null; otherwise, the right.	Learn more
Null-conditional Operators	Short-circuiting operators for nullable types.	Learn more
Type Testing Operators	Checks or casts the types of objects.	Learn more
Range and Index Operators	Enables slicing and accessing elements in a collection.	Learn more
is and as Operators	Used for safe type casting and checking object compatibility with a type.	Learn more
Overloadable Operators	Operators that can be customized or overloaded in a class.	Learn more
Checked and Unchecked	Control overflow behavior in arithmetic operations.	Learn more

---

Methods

Subtopic	Description	Reference Link
Introduction to Methods	Purpose and usage of methods in C#, including syntax and structure	Microsoft - Methods in C# C# Station - Methods Tutorial
Method Parameters	Types of parameters: value parameters, reference parameters, and output parameters. ref and out keywords	Microsoft - Passing Parameters
Return Types and void Methods	Returning values from methods, void methods, and methods with return types	Return Type in C#

		C# Corner - Void and Non-Void Methods
Method Overloading	Concept of method overloading with different parameter types and counts	Microsoft - Method Overloading
		Guru99 - Method Overloading
Optional Parameters and Named Arguments	Using optional parameters with default values and named arguments for clearer method calls	Microsoft - Named and Optional Arguments
		DotNetPerls - Optional Parameters
Static vs Instance Methods	Difference between static and instance methods, and when to use each	Microsoft - Static Classes and Static Class Members
		Code Maze - Static and Non-static Methods
Recursive Methods	Concept of recursion, with examples of recursive methods for problem-solving	TutorialsTeacher - Recursive Method
		C# Corner - Recursive Methods
Local Functions (Nested Methods)	Local functions within a method, including their scope and lifetime	Microsoft - Local Functions
		DotNetCurry - Local Functions in C#
Extension Methods	Creating and using extension methods to add functionality to existing types	Microsoft - Extension Methods
		Code Maze - Extension Methods in C#
Async and Await in Methods	Using async and await keywords to write asynchronous methods	Microsoft - Asynchronous Programming

		C# Corner - Async and Await
Lambda Expressions and Anonymous Methods	Lambda expressions and anonymous methods, used for delegates and LINQ expressions	Microsoft - Lambda Expressions TutorialsTeacher - Anonymous Methods
Method Signatures and Overriding	Understanding method signatures and overriding methods in inheritance with override and virtual	Microsoft - Method Overriding GeeksforGeeks - Method Overriding
Access Modifiers in Methods	Public, private, protected, internal modifiers and their impact on method visibility	Microsoft - Access Modifiers TutorialsTeacher - Access Modifiers
Partial Methods	Partial methods in partial classes, including limitations and benefits	Microsoft - Partial Methods C# Corner - Partial Methods
Best Practices in Method Design	Method naming conventions, keeping methods focused, and optimizing for readability and performance	Microsoft - Method Design C# Station - Coding Standards

Object-Oriented Programming

Category	Subtopic	Reference Link

Classes and Objects	Introduction to Classes and Objects	C# Classes And Objects - In-depth Tutorial With Examples
	Creating and Using Constructors	Constructors in C# - Microsoft Learn
	Object Initialization and Object Lifecycle	Object Lifecycle in C# - Programiz
Abstraction	Introduction to Abstraction	Understanding Abstraction in C# - Programiz
	Abstract Classes and Methods	C# Abstract Class and Method (With Examples) - Programiz
	Abstraction vs Encapsulation	Abstraction and Encapsulation Differences - Microsoft Learn
Encapsulation	Encapsulation Principles	Encapsulation in C# - Dot Net Tutorials
	Getters and Setters	Using Getters and Setters in C# - C# Corner
	Encapsulation and Data Hiding	Data Hiding with Encapsulation - Programiz
	Access Modifiers	Understanding Access Modifiers in C# - Microsoft Learn
	Using Access Modifiers with Encapsulation	C# Access Modifiers with Encapsulation - Dot Net Tutorials
Inheritance	Basics of Inheritance	Inheritance in C# - An In-depth Guide
	Abstract Classes and Methods	C# Abstract Class and Method - Programiz
	Polymorphism with Inheritance	Polymorphism in C# - Microsoft Learn
	The Diamond Problem	Understanding the Diamond Problem in C# - CodeProject

Interfaces	Defining Interfaces and Implementing Interface Members	Interfaces in C#: A Beginner's Guide
	Explicit Interface Implementation	Explicit Interface Implementation in C# - Microsoft Learn
	Interface vs Abstract Class	Interface vs Abstract Class - Dot Net Tutorials
Polymorphism	Overview of Polymorphism and its Types	Understanding Polymorphism in C# - Programiz
	Runtime vs Compile-time Polymorphism	Polymorphism Explained - TutorialsTeacher
	Method Overloading and Overriding	Method Overloading and Overriding in C# - Microsoft Learn
Static Classes	Understanding Static Classes	Static Classes in C# - Microsoft Learn
	Static Members and Methods	Static Members in C# - Programiz
	When to Use Static Classes	Using Static Classes - C# Corner
Relationships	Association - Understanding Association	Association in C# - GeeksforGeeks
	Aggregation - Explaining Aggregation	Aggregation in C# - Programiz
	Composition - Understanding Composition	Composition in C# - Dot Net Tutorials
	Dependency - Explaining Dependency Relationships	Dependency Relationships in C# - TutorialsTeacher
Additional OOP Concepts	Static Classes and Members	Understanding Static Classes and Static Members - Microsoft Learn
	Constructors and Destructors	Constructors and Destructors in C# - Microsoft Learn

Formatting and Parsing

▯ **Type Conversions**

- **Implicit vs Explicit Type Conversions**: C# Type Conversion - Programiz
- **Using as and is for Type Checking**: Type Checking with 'as' and 'is' - Microsoft Learn
- **Conversion Methods and Parse**: Type Conversion Methods in C# - Tutorialspoint

▯ **Generics**

- **Introduction to Generics and Usage**: Generics in C#: A Comprehensive Guide
- **Generic Collections in .NET**: Generic Collections - Microsoft Learn
- **Constraints on Type Parameters**: Constraints on Type Parameters - Microsoft Learn

---

Exception Handling

Subtopic	Description	Reference Links
What is Exception Handling?	Overview of exception handling in C# and its importance for robust application development.	Microsoft Docs: Exception Handling GeeksforGeeks: Exception Handling in C#
try, catch, finally Blocks	Using try, catch, and finally blocks to handle exceptions and ensure clean-up operations.	Microsoft Docs: try-catch-finally TutorialsTeacher: Try-Catch Block
Throwing Exceptions	Understanding the throw statement to raise exceptions intentionally within code.	Microsoft Docs: Throwing Exceptions Dot Net Perls: Throwing Exceptions
Custom Exceptions	Creating custom exception classes that inherit from Exception to represent specific error cases.	Microsoft Docs: Custom Exceptions C# Corner: Custom Exceptions
Common Exception Types	Overview of common C# exceptions like NullReferenceException, ArgumentException, etc.	Microsoft Docs: Common Exceptions Stack Overflow: Common C# Exceptions

Nested try-catch Blocks	Handling exceptions in nested try-catch blocks and understanding their behavior.	Microsoft Docs: Nested try-catch TutorialsTeacher: Nested Try-Catch
Exception Propagation	How exceptions propagate through the call stack and how to handle them effectively.	Microsoft Docs: Propagating Exceptions C# Corner: Exception Propagation
Re-throwing Exceptions	Using throw without arguments in catch blocks to preserve original exception details.	Microsoft Docs: Re-throwing Exceptions Stack Overflow: Rethrowing Exceptions
Using Exception Filters	Using exception filters (when keyword) to conditionally handle exceptions in catch blocks.	Microsoft Docs: Using Exception Filters Dot Net Perls: Exception Filters
Global Exception Handling	Setting up global exception handling at the application level using AppDomain and TaskScheduler.	Microsoft Docs: Global Exception Handling Stack Overflow: Global Exception Handling
Using AggregateException	Handling multiple exceptions that may occur in parallel or asynchronous operations.	Microsoft Docs: AggregateException C# Corner: AggregateException
Exception Logging	Logging exceptions for debugging and monitoring purposes using various logging frameworks.	Microsoft Docs: Logging Exceptions Loggly: Exception Logging Best Practices
Best Practices for Exception Handling	Guidelines for effective exception handling, including performance considerations and user feedback.	Microsoft Docs: Best Practices for Exception Handling Dot Net Perls: Best Practices
Task-based Asynchronous Error Handling	Handling exceptions in async methods with async and await.	Microsoft Docs: Async Exception Handling C# Corner: Async Error Handling

Exception Handling in LINQ	Managing exceptions within LINQ queries and deferred execution contexts.	Microsoft Docs: Handling Errors in LINQ TutorialsTeacher: Exception Handling in LINQ
Throw vs Throw ex	Understanding the difference between throw and throw ex to preserve stack trace or not.	Microsoft Docs: Throw vs Throw ex Code Project: Throw vs Throw ex
Multiple Catches, try	Handling multiple exception types in separate catch blocks within the same try-catch structure.	Microsoft Docs: Multiple Catches TutorialsTeacher: Multiple Catches
finally Block	Using the finally block to execute cleanup code that runs regardless of whether an exception occurs.	Microsoft Docs: finally Block TutorialsTeacher: finally Block
Third-party Logging Frameworks	Using third-party libraries like NLog, log4net, Serilog for advanced exception logging.	NLog Documentation log4net Documentation Serilog Documentation

Collections

Sub-Topic	Reference Link	Description
Introduction to Collections	Collections in C#	An overview of collections in C#, explaining their role in storing and manipulating groups of objects.
	Introduction to Collections in C#	Provides an introduction to collections, types, and methods commonly used for managing data in C#.
Types of Collections in C#	Generic Collections in C#	Describes the generic collections, including types like List<T>, Dictionary<TKey, TValue>, Queue<T>, etc.
	List<T> in C#	Details the List<T> class for storing ordered elements and provides

		methods for adding, removing, and accessing elements.
	Dictionary<TKey, TValue> in C#	Explains how Dictionary<TKey, TValue> is used to store key-value pairs for quick lookups.
	Queue<T> in C#	Describes the Queue<T> collection, which implements a FIFO (First-In-First-Out) structure.
	Stack<T> in C#	Discusses the Stack<T> collection, which uses a LIFO (Last-In-First-Out) structure for storing elements.
	HashSet<T> in C#	Explains HashSet<T> and its unique element storage, ensuring that no duplicates are present.
	LinkedList<T> in C#	Covers LinkedList<T> for representing a doubly linked list of elements.
	Non-Generic Collections (ArrayList, Hashtable)	Describes non-generic collections like ArrayList and Hashtable, which store objects and can store mixed types.
	Non-Generic Queue and Stack	Provides an overview of the non-generic Queue and Stack collections, which don't enforce type safety.
Collection Interfaces	IEnumerable<T> Interface	Describes the IEnumerable<T> interface, which allows objects to be enumerated using a foreach loop.
	ICollection<T> Interface	Covers the ICollection<T> interface for providing methods that deal with elements in a collection.
	IList<T> Interface	Explains the IList<T> interface, which allows indexed access to a collection

		and supports modification operations.
	IDictionary<TKey, TValue> Interface	Discusses the IDictionary<TKey, TValue> interface, used for collections that store key-value pairs.
	ISet<T> Interface	Introduces the ISet<T> interface, which is used by collections that store unique elements like HashSet.
Collection Initialization	Collection Initialization in C#	Details different ways to initialize collections in C#, including collection initializers.
Iterating over Collections	Iterating over Collections with foreach	Covers how to iterate over collections using the foreach loop for better performance and readability.
	LINQ Methods for Collections	Introduces LINQ, a powerful way to query and manipulate collections in C#.
Sorting and Searching Collections	Sorting Collections in C#	Explains how to sort collections using the Sort method or LINQ queries for more complex sorting.
	Searching Collections in C#	Describes methods to search collections, such as Find and FindIndex, to locate specific elements.
Performance of Collections	Choosing the Right Collection Type	Discusses the performance characteristics of different collection types and how to choose the most efficient one for your needs.
	Performance Comparison of Collections	Provides a comparison of various collection types' performance in different scenarios.

Mutability of Collections	Immutable Collections in C#	Discusses immutable collections, which prevent modification of their contents after initialization.
	ReadOnlyCollection<T> in C#	Covers the ReadOnlyCollection<T> type, which provides read-only access to a collection.
Concurrency and Collections	Thread-Safe Collections in C#	Introduces thread-safe collections, which can be safely used in multi-threaded environments.
	ConcurrentDictionary<TKey, TValue> in C#	Explains how to use ConcurrentDictionary for thread-safe key-value storage in a multi-threaded environment.
Custom Collection Classes	Implementing a Custom Collection in C#	Guides on creating your own collection classes to implement specific functionality or behavior.
Collection and LINQ (Language Integrated Query)	Using LINQ with Collections	Shows how to use LINQ to query, filter, and modify collections in C#.
	LINQ Methods	Describes the available LINQ methods, such as Where, Select, OrderBy, and many more, that can be used with collections.
Advanced Collection Concepts	Serialization and Collections	Discusses how to serialize collections to save or transfer their data.
	Cloning Collections in C#	Explains how to clone collections to create deep or shallow copies.
	Asynchronous Programming with Collections	Covers how to work with collections in asynchronous programming.
Working with Collections in	Collections in ASP.NET Core	Explains how to use collections in ASP.NET Core applications, including

Specific Scenarios		dependency injection and model binding.

Generics

Subtopic	Description	Reference Links
Introduction to Generics in C#	Overview of generics, advantages over non-generic collections, and how generics improve type safety	Microsoft Docs - Generics
Generic Classes	How to define and use generic classes with T parameters	Microsoft Docs - Generic Classes
Generic Methods	Creating and using generic methods, benefits of method-level type parameters	Microsoft Docs - Generic Methods
Generic Interfaces	Implementing and using generic interfaces for flexible code design	Microsoft Docs - Generic Interfaces
Generic Delegates	Using generics with delegates, creating type-safe event handling	Microsoft Docs - Generic Delegates
Constraints on Type Parameters	Defining constraints like where T : class, struct, new(), or specific interfaces	Microsoft Docs - Constraints
Covariance and Contravariance in Generics	Understanding type variance, covariance with out and contravariance with in keywords	Microsoft Docs - Covariance and Contravariance
Nullable Types and Generics	Handling nullable types in generic collections and methods	Microsoft Docs - Nullable Types
Generic Collections in .NET	Overview of common generic collections like List<T>, Dictionary<TKey, TValue>, HashSet<T>	Microsoft Docs - Collections
Performance Benefits of Generics	How generics improve performance by reducing boxing/unboxing and increasing type safety	Microsoft Docs - Performance

Best Practices for Using Generics in C#	Guidelines on when to use generics, naming conventions, and tips for creating type-safe APIs	Microsoft Docs - Best Practices

Delegates and Events

Subtopic	Description	Reference Link
Introduction to Delegates	Definition, syntax, creation, and usage of delegates.	Official Microsoft Documentation on Delegates
Types of Delegates	Single-cast vs Multicast, Anonymous, Generic (Func, Action, Predicate).	Types of Delegates in C#
Delegates and Anonymous Methods	Using anonymous methods, comparison with lambda expressions.	Anonymous Methods
Delegates and Lambda Expressions	Introduction, syntax, and usage with lambda expressions.	Lambda Expressions in C#
Events in C#	Definition, declaration, event handlers, and best practices.	Microsoft: Events in C#
Event Handling in C#	Subscribing and unsubscribing to events, syntax.	C# Corner: Event Handling in C#
Delegates vs Events	Differences, use cases, and real-world scenarios.	Differences Between Delegates and Events
Real-World Examples of Delegates and Events	Creating callbacks, event-driven programming, observer pattern.	Examples
Common Pitfalls & Best Practices	Avoiding memory leaks, using weak references, best practices.	Microsoft: Handling Common Pitfalls

LINQ and Iterators

Subtopic	Description	Reference Link
Introduction to LINQ	Overview of LINQ, its purpose, and benefits over traditional iteration.	Microsoft Documentation - Overview of LINQ
	Differences between LINQ and traditional iteration using loops.	Read More
LINQ Syntax and Query Types	Differences between query syntax (SQL-like) and method syntax (extension methods).	LINQ Syntax: Query vs Method
	Basic LINQ queries: Select, Where, OrderBy, GroupBy.	Introduction to LINQ to Objects
Advanced LINQ Queries	How LINQ supports deferred execution and its implications.	Understanding Deferred Execution in LINQ
	Projection operations: Using Select, SelectMany to project data into new forms.	Advanced LINQ Operators
	Set operations: Distinct, Union, Intersect, Except.	
	Quantifiers: Any, All, Contains.	Link
LINQ and Iterators	Basics of iterating collections using foreach and IEnumerator.	C# Iterators and the yield Keyword
	Creating custom iterators using yield return.	Link
	Combining LINQ queries with custom iterators.	Iterators and Collections
LINQ and Performance Considerations	Efficiency of LINQ queries, including deferred execution and memory consumption.	Performance Considerations in LINQ
	Scenarios where traditional loops may be more efficient.	

	Techniques to optimize LINQ performance like .ToList() and avoiding multiple enumerations.	Optimizing LINQ Queries
Practical Examples and Use Cases	Examples of using LINQ with collections like List<T>, Dictionary<TKey, TValue>, etc.	LINQ with Collections - Examples
	How LINQ is used in data access scenarios like LINQ to SQL and Entity Framework.	Using LINQ for Data Access
	Using LINQ to parse and query data in text files and XML.	LINQ for File Operations
Debugging and Troubleshooting LINQ	Tools and techniques for debugging LINQ, including Immediate Window and LINQPad.	Debugging LINQ Queries in C#
	Common errors in LINQ queries like NullReferenceException.	LINQPad - The Ultimate LINQ Debugger

---

Memory Management and File I/O

Topic	Subtopic	Description	Reference Link
Memory Management in C#	Automatic Memory Management (Garbage Collection)	Overview of garbage collection and how it automatically manages memory in C#.	Garbage Collection - Microsoft Docs
	Value Types vs. Reference Types	Difference between value types (stack) and reference types (heap).	Value Types and Reference Types - Microsoft Docs
	Stack and Heap Memory	Explanation of stack and heap memory management in C#.	Stack and Heap Memory - C# Corner

	Garbage Collection Process and Generations	How the garbage collector works and the concept of generations.	Fundamentals of Garbage Collection - Microsoft Docs
	IDisposable Interface and using Statement	Use of IDisposable interface and using statement for resource management.	IDisposable Interface - Microsoft Docs
	Memory Leaks and Best Practices	Identifying and avoiding memory leaks in C#.	Avoiding Memory Leaks - Redgate
	Weak References	Using weak references to allow garbage collection.	WeakReference Class - Microsoft Docs
File I/O Operations in C#	File Handling Basics (System.IO Namespace)	Introduction to file handling in C#.	System.IO Namespace - Microsoft Docs
	Reading and Writing Text Files (StreamReader and StreamWriter)	How to read and write text files using StreamReader and StreamWriter.	Reading and Writing to a Text File - Microsoft Docs
	Working with Binary Files (BinaryReader and BinaryWriter)	Handling binary data with BinaryReader and BinaryWriter.	BinaryReader and BinaryWriter - Microsoft Docs
	File and Directory Management (File, Directory, FileInfo, DirectoryInfo)	Managing files and directories using relevant classes.	File and Directory - Microsoft Docs
	Asynchronous File Operations	Performing asynchronous file operations for responsiveness.	Asynchronous File I/O - Microsoft Docs
	Working with Streams (FileStream, MemoryStream)	Using FileStream and MemoryStream for file and memory operations.	FileStream Class - Microsoft Docs

File I/O Exception Handling	Handling exceptions during file operations.	Exception Handling - Microsoft Docs
Path Operations (Path Class)	Working with file paths using the Path class.	Path Class - Microsoft Docs
Compression and Decompression	Compressing and decompressing files using GZipStream and ZipArchive.	Compressing Files - Microsoft Docs

---

Threads

Subtopic	Description	Reference Links
Introduction to Threads in C#	Basic concepts, single vs multi-threading, thread anatomy, creating a thread	Microsoft Docs - Managed Threads
Creating and Managing Threads	Using Thread class, starting and naming threads, passing parameters	C# Corner - Creating a Thread, GeeksforGeeks - Creating a Thread
Thread Lifecycle in C#	Thread states, lifecycle methods (Abort, Join, Sleep), thread priority	Microsoft Docs - Thread Class
Synchronization in Threads	Avoiding race conditions, lock statement, Mutex, Semaphore, thread synchronization tools	Microsoft Docs - Synchronizing Threads
Thread Safety in C#	Best practices for thread-safe code, using Interlocked class, immutability	Microsoft Docs - Thread Safety
Working with ThreadPool in C#	Benefits of ThreadPool, creating and managing threads with ThreadPool.QueueUserWorkItem	Microsoft Docs - ThreadPool
Asynchronous Programming and Threads	Differences between threads and async programming, Task, async/await usage	Microsoft Docs - Asynchronous Programming with Async and Await

Advanced Threading Techniques	Background vs foreground threads, ThreadLocal<T>, CancellationToken, Parallel.For	Microsoft Docs - Background Threads
Debugging and Performance Optimization of Threads	Debugging tools and techniques, performance analysis, handling common multithreading issues	Link
Best Practices for Multithreading in C#	Avoiding common pitfalls, using high-level APIs, simplicity in code, best practices	Link

---

Serialization

Below are the key topics for Serialization in .NET, along with relevant reference links:

1. **What is Serialization?**
   - **Description**: Serialization is the process of converting an object into a format that can be stored or transmitted. Deserialization is the reverse process, converting the stored format back into an object.
   - **Reference**: Serialization Overview in .NET
2. **Binary Serialization**
   - **Description**: Converts an object into a binary format for compact storage, but is specific to .NET.
   - **Reference**: Binary Serialization in .NET
3. **XML Serialization**
   - **Description**: Converts an object into an XML format, which is readable and platform-independent, often used in web services.
   - **Reference**: XML Serialization in .NET
4. **JSON Serialization**
   - **Description**: Converts objects into JSON format, commonly used for web APIs and data exchange.
   - **Reference**: JSON Serialization in .NET
5. **Custom Serialization**

- **Description**: Provides control over the serialization process by implementing custom logic, such as using the ISerializable interface or attributes.
- **Reference**: Custom Serialization in .NET

6. **Data Contract Serialization**
   - **Description**: Used in WCF for controlling serialized data with fine-grained attributes.
   - **Reference**: Data Contract Serialization in .NET
7. **Serialization Attributes**
   - **Description**: Attributes like [Serializable], [NonSerialized], [DataContract], and [DataMember] control what and how data is serialized.
   - **Reference**: Serialization Attributes in .NET
8. **Security Considerations in Serialization**
   - **Description**: Managing serialization to avoid security risks like data exposure and vulnerabilities.
   - **Reference**: Security Considerations in Serialization

TPL

Subtopic	Description	Reference Link
Introduction to TPL	Overview of TPL, benefits over traditional threading, core concepts like tasks, parallelism, and concurrency.	Task Parallel Library (TPL) Overview
Creating and Managing Tasks	Creating tasks, managing task lifecycle, starting, waiting, cancellation, and handling exceptions.	Tasks and the Task Parallel Library (TPL)
Task Continuations	Using ContinueWith(), task chaining, continuation options, combining tasks with WhenAll and WhenAny.	Continuation Tasks
Task Scheduling and TaskSchedulers	Understanding default task scheduler, creating custom TaskScheduler, controlling task scheduling and execution.	Task Schedulers

Cancellation in TPL	Implementing task cancellation using CancellationToken, cooperative cancellation patterns.	Cancellation in Managed Threads
Exception Handling in TPL	Handling exceptions in tasks, aggregating exceptions using AggregateException, propagating exceptions.	Handling Exceptions in Tasks
Parallel LINQ (PLINQ)	Introduction to Parallel LINQ, executing LINQ queries in parallel, controlling parallel execution.	Parallel LINQ (PLINQ)
Data Parallelism	Using Parallel.For and Parallel.ForEach, managing loop state, optimizing and controlling parallel loops.	Data Parallelism
Asynchronous Programming with TPL	Implementing async programming using async and await, Task-based asynchronous patterns (TAP).	Asynchronous Programming with Async and Await
Synchronization and Concurrent Collections	Managing concurrency with locks, SemaphoreSlim, thread safety, using ConcurrentDictionary, BlockingCollection.	Thread Synchronization, Concurrent Collections
Best Practices for TPL	Best Practices for Task Parallel Library	Best Practices for Task Parallel Library
Task-Based Asynchronous Pattern (TAP)	Understanding TAP, converting existing patterns to TAP, implementing IAsyncResult to Task conversions.	Task-Based Asynchronous Pattern (TAP)

Reflection

Subtopic	Description	Reference Link
Introduction to Reflection	Overview of what Reflection is in .NET and the use of the System.Reflection namespace.	Introduction to Reflection

Accessing Metadata with Reflection	Accessing metadata like types, methods, and properties at runtime using Type.	Accessing Metadata Using Reflection
Working with Assemblies	Loading and exploring assemblies dynamically, using Assembly.Load, GetTypes, etc.	Assemblies in Reflection
Retrieving Type Information	Getting detailed information about types, such as fields, methods, and properties.	Retrieving Type Information
Invoking Methods Using Reflection	Calling methods dynamically at runtime using MethodInfo and the Invoke() method.	Invoking Methods Using Reflection
Accessing Properties and Fields	Reading and modifying properties and fields, including private members using BindingFlags.	Accessing Fields and Properties with Reflection
Creating Instances at Runtime	Creating objects dynamically using Activator.CreateInstance and handling constructors.	Creating Instances with Reflection
Reflection and Attributes	Retrieving and using custom attributes through Reflection, accessing metadata, and properties of attributes.	Reflection and Attributes
Dynamic Method Generation with Emit	Generating methods at runtime using System.Reflection.Emit for creating and executing dynamic assemblies.	Emitting Dynamic Methods
Performance Considerations of Reflection	Understanding the performance overhead, optimization tips, and alternatives to Reflection in performance-critical scenarios.	Performance Considerations with Reflection
Unit Testing with Reflection	Using Reflection for unit testing, testing private members, and understanding pros and cons in testing scenarios.	Unit Testing with Reflection - Practical Example

---

Attribute

Subtopic	Description	Reference Link

Introduction to Attributes	Overview of what attributes are in C# and their purpose.	Introduction to Attributes
Using Attributes	How to apply attributes to code elements and common uses for attributes.	Using Attributes
Defining Custom Attributes	Steps to define and use custom attributes in C#.	Defining Custom Attributes
Accessing Attributes with Reflection	How to retrieve and use attribute information at runtime using reflection.	Accessing Attributes with Reflection

---

ML 5 : ASP .NET MVC :

Category	Subtopic	Reference Link
MVC Architecture	Understanding MVC Architecture	Understanding MVC Architecture
Routing	ASP.NET MVC Routing	ASP.NET MVC Routing
Controllers	Creating Controllers in MVC	Creating Controllers in MVC
Views	Working with Views in MVC	Working with Views in MVC
Models	Defining Models in MVC	Defining Models in MVC
HTML Helpers	Using HTML Helpers in MVC	Using HTML Helpers in MVC
Data Access	Using Entity Framework in MVC	Using Entity Framework in MVC
Dependency Injection (DI)	Implementing DI in MVC	Implementing DI in MVC
State Management	State Management in MVC	State Management in MVC
Security	Security Best Practices in MVC	Security Best Practices in MVC
Filters	Using Filters in MVC	Using Filters in MVC
Custom Error Handling	Handling Errors in MVC	Handling Errors in MVC

Bundling and Minification	Optimizing MVC with Bundling	Optimizing MVC with Bundling
AJAX in MVC	AJAX with ASP.NET MVC	AJAX with ASP.NET MVC
API Integration	Integrating APIs in MVC	Integrating APIs in MVC
Localization and Globalization	Localization in MVC	Localization in MVC
Performance Optimization	Performance Tuning in MVC	Performance Tuning in MVC
Testing in MVC	Testing ASP.NET MVC Applications	Testing ASP.NET MVC Applications
Configuration Management	Managing Configuration in MVC	Managing Configuration in MVC
Deployment	Deploying MVC Applications	Deploying MVC Applications

---

ML 6 : .NET Core/Entity Framework/Dapper etc.

Category	Subtopic	Reference Link
Fundamentals	Introduction to .NET Core	Introduction to .NET Core
	ASP.NET Core Overview	ASP.NET Core Overview
	Dependency Injection in ASP.NET Core	Dependency Injection in ASP.NET Core
	Middleware in ASP.NET Core	Middleware in ASP.NET Core
Routing and Endpoints	Routing in ASP.NET Core	Routing in ASP.NET Core
	Endpoint Routing	Endpoint Routing in ASP.NET Core

	Attribute Routing	Attribute Routing in ASP.NET Core
Controllers and Actions	Controllers in ASP.NET Core	Controllers in ASP.NET Core
	Action Results	Action Results in ASP.NET Core
	Model Binding	Model Binding in ASP.NET Core
Razor Pages	Introduction to Razor Pages	Introduction to Razor Pages
	Razor Pages vs. MVC	Razor Pages vs MVC
	Building a Razor Page Application	Building a Razor Page Application
Data Access	Entity Framework Core Overview	Entity Framework Core Overview
	Getting Started with EF Core	Getting Started with EF Core
	DbContext and DbSet in EF Core	DbContext and DbSet
Security	ASP.NET Core Authentication	Authentication in ASP.NET Core
	Authorization in ASP.NET Core	Authorization in ASP.NET Core
	Identity in ASP.NET Core	Identity in ASP.NET Core
	JWT Authentication	JWT Authentication in ASP.NET Core
Configuration and Settings	Configuration in ASP.NET Core	Configuration in ASP.NET Core
	appsettings.json and Environment Variables	appsettings.json and Environment Variables
	Environment-specific Configuration	Environment-specific Configuration
Logging and Diagnostics	Logging in ASP.NET Core	Logging in ASP.NET Core
	ILogger in ASP.NET Core	ILogger in ASP.NET Core
	Diagnostics in .NET Core	Diagnostics in .NET Core

Testing	Testing with xUnit in .NET Core	Testing with xUnit in .NET Core
	Unit Testing Controllers	Unit Testing Controllers
	Mocking in .NET Core with Moq	Mocking in .NET Core with Moq
Performance and Caching	Caching in ASP.NET Core	Caching in ASP.NET Core
	MemoryCache and Distributed Cache	MemoryCache and Distributed Cache
	Performance Tips for ASP.NET Core	Performance Tips for ASP.NET Core
API Development	Creating REST APIs with ASP.NET Core	Creating REST APIs
	API Versioning in ASP.NET Core	API Versioning
	Securing APIs with OAuth2	Securing APIs with OAuth2
SignalR	Introduction to SignalR	Introduction to SignalR
	Building Real-Time Applications with SignalR	Building Real-Time Applications with SignalR
	SignalR Hubs	SignalR Hubs
File Handling	File Uploads in ASP.NET Core	File Uploads in ASP.NET Core
	Managing Static Files	Managing Static Files
	File Providers	File Providers in ASP.NET Core
Deployment and Hosting	Deploying ASP.NET Core Apps	Deploying ASP.NET Core Apps
	Hosting on IIS, Linux, and Docker	Hosting on IIS, Linux, and Docker
	Continuous Deployment with Azure	Continuous Deployment with Azure
Blazor	Blazor Overview	Blazor Overview
	Blazor Server vs. WebAssembly	Blazor Server vs WebAssembly
	State Management in Blazor	State Management in Blazor

ML 7 : Web API/REST

Category	Subtopic	Reference Link
Overview of Web APIs	Introduction to REST and HTTP	Introduction to REST and HTTP
	RESTful API Design Guidelines	RESTful API Design Guidelines
ASP.NET Core Fundamentals	Setting up an ASP.NET Core project	Setting up an ASP.NET Core Project
	ASP.NET Core Documentation	ASP.NET Core Documentation
Routing and Controllers	Understanding routing and controller actions	Understanding Routing and Controller Actions
	Routing in ASP.NET Core	Routing in ASP.NET Core
Data Access with Entity Framework Core	Setting up Entity Framework Core	Setting up Entity Framework Core
	Entity Framework Core Documentation	Entity Framework Core Documentation
Middleware and Dependency Injection	Understanding middleware and DI in ASP.NET Core	Understanding Middleware and DI in ASP.NET Core
	ASP.NET Core Middleware	ASP.NET Core Middleware
Authentication and Authorization	Implementing JWT and OAuth2	Implementing JWT and OAuth2 in ASP.NET Core
	Authentication and Authorization in ASP.NET Core	Authentication and Authorization in ASP.NET Core
API Documentation with Swagger	Setting up Swagger for API documentation	Setting up Swagger for API Documentation

	Swagger in ASP.NET Core	Swagger in ASP.NET Core
Error Handling and Logging	Implementing global error handling and logging	Implementing Global Error Handling and Logging
	Error Handling in ASP.NET Core	Error Handling in ASP.NET Core

ML 8 : OO Design Principles & Patterns

SOLID Principles	Understanding the SOLID Principles with Real-Time Examples
Design Patterns	- Design Patterns in C# (GeeksforGeeks) - Comprehensive Course on .NET Design Patterns
Microservices	- Microservices Architecture Overview - Microservices with .NET - Designing Microservices - Interservice Communication - Data Management in Microservices - Deployment and Orchestration - Security in Microservices - Distributed Transactions - Distributed Data Management - Awesome Microservices .NET - Microservices Best Practices
Clean Code	- Clean Code Principles - Clean Coders - Clean Code for .NET - .NET Microservices: Architecture for Containerized .NET Applications - Clean Coding Practices
UML Diagrams	- UML Tutorial (Guru99) - UML Class Diagrams Reference - UML Cheatsheets and Reference Guides

ML 9 : Cloud

Introduction to Azure and Cloud Fundamentals

- Introduction to Azure: Azure Fundamentals Documentation

- https://learn.microsoft.com/en-us/training/paths/microsoft-azure-fundamentals-describe-cloud-concepts/

Azure App Service: Deploy Web Applications

- https://learn.microsoft.com/en-us/azure/app-service/
- https://learn.microsoft.com/en-us/azure/app-service/quickstart-dotnetcore?tabs=net80&pivots=development-environment-vs

Azure Functions (Serverless): Azure Functions Overview

- https://learn.microsoft.com/en-us/azure/azure-functions/
- https://learn.microsoft.com/en-us/azure/azure-functions/functions-create-function-app-portal?pivots=programming-language-csharp
- https://learn.microsoft.com/en-us/azure/azure-functions/functions-triggers-bindings?tabs=isolated-process%2Cnode-v4%2Cpython-v2&pivots=programming-language-csharp
- https://learn.microsoft.com/en-us/training/modules/execute-azure-function-with-triggers/

Azure Logic Apps

- https://learn.microsoft.com/en-us/azure/logic-apps/
- https://learn.microsoft.co m/en-us/azure/logic-apps/quickstart-create-example-consumption-workflow

Azure SQL Database: Azure SQL Documentation

- https://learn.microsoft.com/en-us/azure/azure-sql/?view=azuresql
- https://learn.microsoft.com/en-us/azure/azure-sql/database/single-database-create-quickstart?view=azuresql&tabs=azure-portal

Azure Blob Storage: Blob Storage Overview

- https://learn.microsoft.com/en-us/azure/storage/blobs/storage-blobs-overview
- https://learn.microsoft.com/en-us/azure/storage/blobs/storage-quickstart-blobs-portal

Working with NoSQL in Azure: Azure Cosmos DB Overview

- https://learn.microsoft.com/en-us/azure/cosmos-db/
- https://learn.microsoft.com/en-us/azure/cosmos-db/nosql/quickstart-dotnet

Azure Service Bus

- https://learn.microsoft.com/en-us/azure/service-bus-messaging/
- https://learn.microsoft.com/en-us/azure/service-bus-messaging/service-bus-dotnet-get-started-with-queues?tabs=passwordless

Monitoring and Optimization

- https://learn.microsoft.com/en-us/azure/azure-monitor/
- https://learn.microsoft.com/en-us/azure/azure-monitor/app/app-insights-overview
- https://learn.microsoft.com/en-us/azure/azure-monitor/app/asp-net-core

Azure Active Directory (Azure AD)

- https://learn.microsoft.com/en-us/entra/identity/
- https://learn.microsoft.com/en-us/entra/fundamentals/whatis
- https://learn.microsoft.com/en-us/azure/azure-functions/functions-bindings-http-webhook-trigger?tabs=python-v2%2Cisolated-process%2Cnodejs-v4%2Cfunctionsv2&pivots=programming-language-csharp
- https://learn.microsoft.com/en-us/azure/logic-apps/logic-apps-securing-a-logic-app?tabs=azure-portal

Azure Key Vault

- **https://learn.microsoft.com/en-us/azure/key-vault/general/overview**
- https://learn.microsoft.com/en-us/azure/key-vault/general/basic-concepts

---

ML 10 : Client Side Technologies

JavaScript Fundamentals

- Syntax and Operators: MDN: JavaScript Syntax
- Variables and Data Types: MDN: Data Types
- Control Structures (if, switch, loops): MDN: Control Flow
- Functions (declaration, expression, arrow functions): MDN: Functions
- Scope (global vs. local, lexical scope): MDN: Scope
- Hoisting and Closures: MDN: Hoisting | MDN: Closures
- IIFE
- Prototype

- References: MDN JavaScript Guide

Object-Oriented JavaScript

- Objects and Prototypes: MDN: Objects
- Inheritance (constructor functions, prototype chain): MDN: Prototypes
- ES6 Classes: MDN: Classes
- Modules (import/export): MDN: JavaScript Modules
- Object Destructuring and Spread Operator: MDN: Destructuring assignment | MDN: Spread operator
- References: Understanding ECMAScript 6

Asynchronous JavaScript

- Callbacks: MDN: Callbacks
- Promises (creation, chaining): MDN: Promises
- Async/Await: MDN: Async/Await
- Error Handling in Asynchronous Code: MDN: Error Handling
- Fetch API and AJAX: MDN: Fetch API
- References: JavaScript.info - Async

Advanced JavaScript

- Functional Programming Concepts (higher-order functions, map, filter, reduce): MDN: Functional Programming
- The 'this' Keyword: MDN: this
- Event Delegation: MDN: Event Delegation
- JavaScript Design Patterns (module, factory, singleton): MDN: Design Patterns
- Memory Management and Garbage Collection: MDN: Memory Management
- References: JavaScript Design Patterns

JavaScript in the Browser

- Document Object Model (DOM) Manipulation: MDN: DOM Manipulation
- Event Handling: MDN: Events
- Local Storage and Session Storage: MDN: Web Storage API
- Web APIs (Geolocation, Fetch, etc.): MDN: Web APIs

- Responsive Design and Mobile Browser Considerations: MDN: Responsive Web Design
- References: MDN Web APIs

Angular Basics

- Introduction to Angular: Angular: Getting Started
- Setting Up an Angular Project with Angular CLI: Angular CLI
- Angular Architecture Overview: Angular: Architecture
- Components and Templates: Angular: Components
- Data Binding (One-way, Two-way): Angular: Data Binding
- Directives (Structural and Attribute): Angular: Directives
- References: Angular Official Guide

Services and Dependency Injection

- Creating and Using Services: Angular: Services
- Dependency Injection in Angular: Angular: Dependency Injection
- Observables and RxJS Basics: RxJS: Getting Started
- Using HttpClient for API Calls: Angular: HttpClient
- Managing Application State with Services: Angular: Services
- References: Angular Services

Routing and Navigation

- Angular Router Basics: Angular: Router
- Configuring Routes: Angular: Route Configuration
- Route Parameters and Query Parameters: Angular: Route Parameters
- Route Guards (CanActivate, CanDeactivate): Angular: Route Guards
- Lazy Loading Modules: Angular: Lazy Loading
- References: Angular Routing & Navigation

Forms in Angular

- Template-driven Forms: Angular: Template-driven Forms
- Reactive Forms: Angular: Reactive Forms

- Form Validation (built-in and custom validators): Angular: Form Validation
- Handling Form Events: Angular: Form Events
- Managing Form State and Submitting Data: Angular: Submitting Forms
- References: Angular Forms

Advanced Angular Concepts

- Change Detection Strategies: Angular: Change Detection
- Lifecycle Hooks (ngOnInit, ngOnChanges, etc.): Angular: Lifecycle Hooks
- Pipes and Custom Pipes: Angular: Pipes
- Managing State with NgRx: NgRx: Getting Started
- Optimizing Angular Applications: Angular: Performance
- References: Angular Advanced Guide

-------------------------------DONE ---