**EVOLUTION OF .NET FRAMEWORK**  
Microsoft started development on the [.NET Framework](https://en.wikipedia.org/wiki/.NET_Framework) in the late 1990s originally under the name of Next Generation Windows Services. The first version of .NET Framework was released on 13 February 2002, bringing [managed code](https://en.wikipedia.org/wiki/Managed_code) to [Windows NT 4.0](https://en.wikipedia.org/wiki/Windows_NT_4.0), [98](https://en.wikipedia.org/wiki/Windows_98), [2000](https://en.wikipedia.org/wiki/Windows_2000), [ME](https://en.wikipedia.org/wiki/Windows_ME) and [XP](https://en.wikipedia.org/wiki/Windows_XP).

Since the first version, Microsoft has released nine more upgrades for .NET Framework, seven of which have been released along with a new version of [Visual Studio](https://en.wikipedia.org/wiki/Visual_Studio). Two of these upgrades, .NET Framework 2.0 and 4.0, have upgraded [Common Language Runtime](https://en.wikipedia.org/wiki/Common_Language_Runtime) (CLR).

.NET Framework 4.8 was announced as the final version of .NET Framework, with future work going into the rewritten and [cross-platform](https://en.wikipedia.org/wiki/Cross-platform_software) [.NET Core](https://en.wikipedia.org/wiki/.NET) platform (later, simply *.NET*). However, .NET Framework 4.8.1 was released in August 2022.  
  
 **EVOLUTION OF C#**The Microsoft was released the first version of C# 1.0 bundle with .NET Framework 1.0 in Visual Studio.NET 2002 at January 2002. C# was introduced programming language to develop .NET Application and it’s supported managed code initially.  
  
The Microsoft was released the second version of C# 2.0 bundle with .NET Framework 2.0 in Visual Studio.NET 2005 at November 2005. The Microsoft was released version of C # 4.0 bundles with .NET Framework 4.0 in Visual Studio.NET 2010 at April 2010.

This version introduced couple of features like Dynamic binding, Optional parameters, Named arguments and more COM interoperability.  
  
The Microsoft was released version of C # 5.0 bundles with .NET Framework 5.0 in Visual Studio.NET 2012 at August 2012. This version introduced few features Asynchronus and Caller information.

2. **MONO: Sponsored by Microsoft, Mono is** an open source implementation of Microsoft's .NET Framework as part of the .NET Foundation and based on the ECMA standards for C# and the Common Language Runtime.

ii. **Xamarin:** is a developer's tool for cross-platform mobile application development. When Microsoft acquired the platform and made it an open-source product, it became a more popular cross-development platform tool. It is for building mobile applications for Android, iOS, and Windows with C# and . NET framework.

iii. **COM**: The Component Object Model (COM) lets an object expose its functionality to other components and to host applications on Windows platforms. To help enable users to interoperate with their existing code bases.

iv. .**NET Core**: Net core is used for application building for different operating systems like Mac, Windows, and Linux. And is much faster than .Net. It is a scalable system and is high-performance without the UI.

V. **UNITY C#:** The language that's used in Unity is called C# (pronounced C-sharp). All the languages that Unity operates with are object-oriented scripting languages.

VI. **REST**: The word REST stands for REpresentational State Transfer. In simple terms it's a pattern for creating an API. API stands for Application Programming Interface.

3. i. **Memory Management:** In the common language runtime (CLR), the garbage collector (GC) serves as an automatic memory manager. The garbage collector manages the allocation and release of memory for an application. Therefore, developers working with managed code don't have to write code to perform memory management tasks.

ii. **Exception Handling**: Exception, the CLR execution engine can raise exceptions, and unmanaged code can raise exceptions as well. Exceptions raised on a thread of execution follow the thread through native and managed code, across AppDomains, and, if not handled by the program, are treated as unhandled exceptions by the operating system.

iii. **Safety and Thread management**: These thread-safe collections ensure that data access is synchronized to prevent data inconsistencies. Some examples of thread-safe collections include ConcurrentDictionary, ConcurrentQueue, ConcurrentStack, and ConcurrentBag.