Applied Technologies Descriptions of .NET Implementations

	OS	Open Source	Purpose
.NET Framework	Windows	No	Used for building Windows desktop applications and ASP.NET Web apps running on IIS.
.NET Core	Windows, Linux, macOS	Yes	Used for building cross-platform console apps and ASP.NET Core Web apps and cloud services.
Xamarin	iOS, Android, macOS	Yes	Used for building mobile applications for iOS and Android, as well as desktop apps for macOS.
.NET Standard	N/A	Yes	Used for building libraries that can be referenced from all .NET implementations, such as .NET Framework, .NET Core and Xamarin.

Following are the different applied technologies:

User-Interface APIs

User-interface–based applications can be divided into two categories: thin client, which amounts to a website, and rich client, which is a program the end user must download and install on a computer or mobile device.

For thin client applications, .NET provides ASP.NET and ASP.NET Core. For rich-client applications that target Windows 7/8/10 desktop, .NET provides the WPF and Windows Forms APIs. For rich-client apps that target iOS, Android, and Windows Phone, there's Xamarin, and for writing rich-client store apps for Windows 10 desktop and devices.

Finally, there is a hybrid technology called Silverlight, which has largely been abandoned since the rise of HTML5.

ASP.NET

Applications written using ASP.NET host under Windows IIS and can be accessed from any web browser. Here are the advantages of ASP.NET over rich-client technologies:

- There is zero deployment at the client end.
- Clients can run a non-Windows platform.
- The updates are easily deployed.

In writing your web pages, you can choose between the traditional Web Forms and the newer MVC (Model-View-Controller) API. Both build on the ASP.NET infrastructure. Web Forms has been part of the Framework since its inception; MVC was written much later in response

to the success of Ruby on Rails and MonoRail. It provides, in general, a better programming abstraction than Web Forms; it also allows more control over the generated HTML.

ASP.NET Core

A relatively recent addition, ASP.NET Core is similar to ASP.NET, but runs on both .NET Framework and .NET Core (allowing for cross-platform deployment). ASP.NET Core features a lighter-weight modular architecture, with the ability to self-host in a custom process, and an open source license. Unlike its predecessors, ASP.NET Core is not dependent on *System.Web* and the historical baggage of Web Forms. It is particularly suitable for micro-services and deployment inside containers.

Windows Presentation Foundation (WPF)

WPF was introduced in Framework 3.0 for writing rich-client applications. The benefits of WPF over its predecessor, Windows Forms, are as follows:

- It supports sophisticated graphics, such as arbitrary transformations, 3D rendering, multimedia, and true transparency. Skinning is supported through styles and templates.
- Its primary measurement unit is not pixel-based, so applications display correctly at any DPI (dots per inch) setting.
- It has extensive and flexible layout support, which means you can localize an application without danger of elements overlapping.
- Rendering uses DirectX and is fast, taking good advantage of graphics hardware acceleration.
- It offers reliable data binding.
- User interfaces can be described declaratively in XAML files that can be maintained independently of the "code-behind" files—this helps to separate appearance from functionality.

Windows Forms

Windows Forms is a rich-client API that's as old as the .NET Framework. Compared to WPF, Windows Forms is a relatively simple technology that provides most of the features you need in writing a typical Windows application. It also has significant relevancy in maintaining legacy applications. It has a number of drawbacks, though, compared to WPF:

- Controls are positioned and sized in pixels, making it easy to write applications that break on clients whose DPI settings differ from the developer's (although this has improved somewhat in Framework 4.7).

- The API for drawing nonstandard controls is GDI+, which, although reasonably flexible, is slow in rendering large areas (and without double buffering, may flicker).
- Controls lack true transparency.
- Most controls are non-compositional. For instance, you cannot put an image control inside a tab control header. Customizing list views and combo boxes is time- consuming and painful.
- Dynamic layout is difficult to get right reliably.

Xamarin

Xamarin, now owned by Microsoft, lets you write mobile apps in C# that target iOS and Android, as well as Windows Phone. Being cross-platform, this runs not on the .NET Framework, but its own framework (a derivation of the open source Mono framework).

UWP (Universal Windows Platform)

UWP is for writing apps that target Windows 10 desktop and devices, distributed via the Windows Store. Its rich-client API is designed for writing touch-first user interfaces, and was inspired by WPF and uses XAML for layout. The namespaces are *Windows.UI* and *Windows.UI.Xaml*.

Silverlight

Silverlight is also distinct from the .NET Framework, and lets you write a graphical UI that runs in a web browser, much like Macromedia's Flash. With the rise of HTML5, Microsoft has abandoned Silverlight.

Backend Technologies

ADO.NET

ADO.NET is the managed data access API. Although the name is derived from the 1990s-era ADO (ActiveX Data Objects), the technology is completely different. ADO.NET contains two major low-level components:

1. **Provider layer**:
   The provider model defines common classes and interfaces for low-level access to database providers. These interfaces comprise connections, commands, adapters, and readers (forward-only, read-only cursors over a database). The Framework ships with native support for Microsoft SQL Server, and numerous third-party drivers are available for other databases.
2. **DataSet model**
   A DataSet is a structured cache of data. It resembles a primitive in-memory database, which defines SQL constructs such as tables, rows, columns, relationships, constraints, and views. By programming against a cache of data, you can reduce the number of trips to the server, increasing server scalability and the responsiveness of a rich-client user interface. DataSets are serializable and are designed to be sent across the wire between client and server applications.

Sitting above the provider layer are three APIs that offer the ability to query databases via LINQ:

- Entity Framework (.NET Framework only)
- Entity Framework Core (.NET Framework and .NET Core)
- LINQ to SQL (.NET Framework only)

LINQ to SQL is simpler than Entity Framework, and has historically produced better SQL (although Entity Framework has benefited from numerous updates).

Entity Framework is more flexible in that you can create elaborate mappings between the database and the classes that you query (Entity Data Model), and offers a model that allows third-party support for databases other than SQL Server.

Entity Framework Core (EF Core) is a rewrite of Entity Framework with a simpler design inspired by LINQ to SQL. It abandons the complex Entity Data Model and runs on both .NET Framework and .NET Core.

Windows Workflow (.NET Framework only)

Windows Workflow is a framework for modelling and managing potentially long running business processes. Workflow targets a standard runtime library, providing consistency and interoperability. Workflow also helps reduce coding for dynamically controlled decision-making trees. Windows Workflow is not strictly a backend technology—you can use it anywhere.

Workflow came originally with .NET Framework 3.0, with its types defined in the *System.WorkFlow* namespace. Workflow was substantially revised in Framework 4.0; the new types live in and under the *System.Activities* namespace.

COM+ and MSMQ (.NET Framework only)

The Framework allows you to interoperate with COM+ for services such as distributed transactions, via types in the *System.EnterpriseServices* namespace. It also supports MSMQ (Microsoft Message Queuing) for asynchronous, one-way messaging through types in *System.Messaging*.

Distributed System Technologies

Windows Communication Foundation (WCF)

WCF is a sophisticated communications infrastructure introduced in Framework 3.0. WCF is flexible and configurable enough to make both of its predecessors— Remoting and (.ASMX) Web Services—mostly redundant. WCF, Remoting, and Web Services are alike in that they implement the following basic model in allowing a client and server application to communicate:

- On the server, you indicate what methods you would like remote client applications to be able to call.
- On the client, you specify or infer the signatures of the server methods you'd like to call.
- On both the server and the client, you choose a transport and communication protocol (in WCF, this is done through a binding).
- The client establishes a connection to the server.
- The client calls a remote method, which executes transparently on the server.

WCF further decouples the client and server through service contracts and data contracts. Conceptually, the client sends an (XML or binary) message to an end- point on a remote service, rather than directly invoking a remote method. One of the benefits of this decoupling

is that client have no dependency on the .NET platform or on any proprietary communication protocols.
For .NET-to-.NET communication, however, WCF offers richer serialization and better tooling than with REST APIs. It is also potentially faster as it is not tied to HTTP and can use binary serialization.
The types for communicating with WCF are in, and below, the *System.Service* Model

Web API
Web API runs over ASP.NET/ASP.NET Core and is architecturally similar to Microsoft's MVC API, except that it is designed to expose services and data instead of web pages. Its advantage over WCF is in allowing you to follow popular REST over HTTP conventions, offering easy interoperability with the widest range of platforms.
REST implementations are internally simpler than the SOAP and WS- protocols that WCF relies on for interoperability. REST APIs are also architecturally more elegant for loosely-coupled systems, building on de-facto standards and making excellent use of what HTTP already provides.

Remoting and .ASMX Web Services (.NET Framework only)
Remoting and .ASMX Web Services are WCF's predecessors. Remoting is almost redundant in WCF's wake, and .ASMX Web Services has become entirely redundant.
Remoting is geared toward tightly coupled applications. A typical example is when the client and server are both .NET applications written by the same company (or companies sharing common assemblies). Communication typically involves exchanging potentially complex custom .NET objects that the Remoting infrastructure serializes and deserializes without needing intervention.
The types for Remoting are in or under *System.Runtime.Remoting*; the types for Web Services are under *System.Web.Services*.

Scope of .Net Technology

Over the period, many software have evolved along with many new technologies being introduced. In this race, one, which caught the people's interest, was .Net. In a very short time, this new technology became the boon for the software developer community and now it has been considered as the most growing career option, which clearly indicates that .Net development still rules.

Its growing popularity has made it the first choice of many experienced and fresher and now one can think of having a great career start in this field outside India too. .Net is now part of

many international markets like USA, UAE, South Africa, UK and other developing countries and is heading forward with each passing day. With its every new version .Net technologies is evolving at a fast pace and creating amazing job opportunities for the developers.

The availability of RAD in.Net, which means the Rapid Application Development, is the reason behind its success. The plus point of learning this technology is that you can develop as many applications as you want for different platforms and environments. You can even use it for building XML web applications and web services that can excellently run on the Internet. .Net is best suited for developing window-based applications, web server programs and applications, which are both PC and mobile compatible. It is easy to transfer feature is what makes it the popular choice.

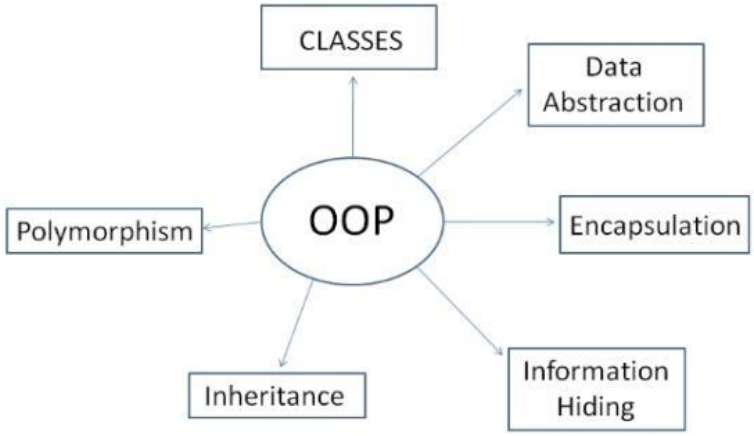
The biggest advantage of learning .Net is that one can get a job in various profiles like he/she can be absorbed as a software developer also or a.Net technician too. Today, there are an array of institutes and firms that offer certified and short-term course in .Net, which is a great move from career point of view. Whether you are a diploma holder, an Engineer, or an MCA, learning .Net will surely set your career and will offer it a right pace and track. There are ample of career options in this particular field. An interested candidate can go for MCTS(VB.net), MCTS(ASP.net) and MCPD, which are some of the international certifications. You can even choose from Cisco certifications like CCNA, CCNP, CCIE, which will give a new direction to your career.

With so many job prospects in this technology, choosing it will be an ideal choice for your career. This clearly illustrates the future scope of .Net, which is sure to offer you great future ahead in almost all the spheres, ranging from Desktop applications to mobile applications.

Feature of Object Oriented Programming:

Object-oriented programming (OOP) refers to a type of computer programming (software design) in which programmers define not only the data type of a data structure, but also the types of operations (functions) that can be applied to the data structure.

In this way, the data structure becomes an object that includes both data and functions. In addition, programmers can create relationships between one object and another. For example, objects can inherit characteristics from other objects.



Following are the features of OOPS:

Abstraction: The process of picking out (abstracting) common features of objects and procedures.

Class: A category of objects. The class defines all the common properties of the different objects that belong to it.

Encapsulation: The process of combining elements to create a new entity. A procedure is a type of encapsulation because it combines a series of computer instructions.

Information hiding: The process of hiding details of an object or function. Information hiding is a powerful programming technique because it reduces complexity.

Inheritance: a feature that represents the "is a" relationship between different classes.

Interface: the languages and codes that the applications use to communicate with each other and with the hardware.

Object: a self-contained entity that consists of both data and proceduresto manipulate the data.

Polymorphism: A programming language's ability to process objects differently depending on their data type or class.

Procedure: a section of a program that performs a specific task.

Procedure-Oriented vs. Object-Oriented Programming

Object-Oriented Programming (OOP) is a high-level programming language where a program is divided into small chunks called objects using the object-oriented model, hence the name. This paradigm is based on objects and classes.

- **Object** – An object is a self-contained entity that accumulates both data and procedures to manipulate the data. Objects are merely instances of classes.
- **Class** – A class, in simple terms, is a blueprint of an object, which defines all the common properties of one or more objects that are associated with it. A class can be used to define multiple objects within a program.

The OOP paradigm mainly eyes on the data rather than the algorithm to create modules by dividing a program into data and functions that are bundled within the objects. The modules cannot be modified when a new object is added restricting any non-member function access to the data. Methods are the only way to assess the data.

Objects can communicate with each other through same member functions. This process is known as message passing. This anonymity among the objects is what makes the program secure. A programmer can create a new object from the already existing objects by taking most of its features thus making the program easy to implement and modify.

Procedure-Oriented Programming (POP) follows a systematic (step-by-step) approach to break down a task into a collection of variables and routines (or subroutines) through a sequence of instructions. Each step is carried out in order in a systematic manner so that a computer can understand what to do. The program is divided into small parts called functions and then it follows a series of computational steps to be carried out in order.

It follows a top-down approach to actually solve a problem, hence the name. Procedures correspond to functions and each function has its own purpose. Dividing the program into functions is the key to procedural programming. So a number of different functions are written in order to accomplish the tasks.

Difference between OOP and POP is shown below:

Procedure Oriented Programming (POP)	**Object Oriented Programming (OOP)**
In POP, program is divided into small parts called **functions**.	In OOP, program is divided into parts called **objects.**
In POP, Importance is not given to **data** but to functions as well as **sequence** of actions to be done.	In OOP, Importance is given to the data rather than procedures or functions because it works as a **real world.**
POP follows **Top Down approach**.	OOP follows **Bottom Up approach.**
POP does not have any access specifier.	OOP has access specifiers named Public, Private, Protected, etc.
In POP, Data can move freely from function to function in the system.	In OOP, objects can move and communicate with each other through member functions.
To add new data and function in POP is not so easy.	OOP provides an easy way to add new data and function.

In POP, most function uses Global data for sharing that can be accessed freely from function to function in the system.	In OOP, data cannot move easily from function to function, it can be kept public or private so we can control the access of data.
POP does not have any proper way for hiding data so it is **less secure**.	OOP provides Data Hiding so provides **more security.**
In POP, Overloading is not possible.	In OOP, overloading is possible in the form of Function Overloading and Operator Overloading.
Example of POP are : C, VB, FORTRAN, Pascal.	Example of OOP are : C++, JAVA, VB.NET, C#.NET.