Checklist

1. Make sure that there shouldn't be any **project warnings**.
2. It will be much better if ***Code Analysis*** is performed on a project (*with all Microsoft Rules enabled*) and then remove the warnings.
3. All **unused usings** need to be removed. Code cleanup for unnecessary code is always a good practice.
4. '**null**' check needs to be performed wherever applicable to avoid the *Null Reference Exception* at runtime.
5. Naming conventions to be followed always. Generally for variables/parameters, follow **Camel casing**and for method names and class names, follow **Pascal casing**.
6. Make sure that you are aware of SOLID principles.
7. **Code Reusability**: Extract a method if the same piece of code is being used more than once or you expect it to be used in future. Make some generic methods for repetitive task and put them in a related class so that other developers start using them once you intimate them. Develop user controls for common functionality so that they can be reused across the project.
8. **Code Consistency**: Let's say that an Int32 type is coded as int and String type is coded as string, then they should be coded in that same fashion across the application. But not like sometimes int and sometimes as Int32.
9. **Code Readability**: Should be maintained so that other developers understand your code easily.
10. **Disposing of Unmanaged Resources** like File I/O, Network resources, etc. They have to be disposed of once their usage is completed. Use usings block for unmanaged code, if you want to automatically handle the disposing of objects once they are out of scope.
11. Proper implementation of **Exception Handling** (try/catch and finally blocks) and logging of exceptions.
12. Making sure that methods have less number of lines of code. Not more than 30 to 40 lines.
13. Timely check-in/check-out of files/pages at source control (like TFS).
14. **Peer code reviews**. Swap your code files/pages with your colleagues to perform internal code reviews.
15. **Unit Testing**. Write developer test cases and perform unit testing to make sure that basic level of testing is done before it goes to QA testing.
16. **Avoid nested for/foreach loops** and nested if conditions as much as possible.
17. Use **anonymous types** if code is going to be used only once.
18. Try using **LINQ**queries and **Lambda** expressions to improve Readability.
19. Proper usage of var, object, and dynamic keywords. They have some similarities due to which most of the developers are confused or don’t know much about them and hence they use them interchangeably, which shouldn't be the case.
20. Use **access specifiers** (*private, public, protected, internal, protected internal*) as per the scope need of a method, a class, or a variable. Let's say if a class is meant to be used only within the assembly, then it is enough to mark the class as internal only.
21. Use interfaces wherever needed to maintain decoupling. Some design patterns came into existence due to the usage of interfaces.
22. Mark a class as sealed or static or abstract as per its usage and your need.
23. Use a Stringbuilder instead of string if multiple concatenations are required, to save **heap memory**.
24. Check whether any **unreachable code** exists and modify the code if it exists.
25. Write **comments**on top of all methods to describe their usage and expected input types and return type information.
26. Use a tool like **Silverlight Spy** to check and manipulate **rendered XAML in Runtime of a Silverlight**application to improve productivity. This saves lot of back & forth time between Design & Run views of the XAML.
27. Use **fiddler**tool to check the HTTP/network traffic and bandwidth information to trace the performance of web application and services.
28. Use *WCFTestClient.exe* tool if you want to verify the service methods out of the Visual Studio or by attaching its process to Visual Studio for debugging purposes.
29. Use **constants and readonly**wherever applicable.
30. Avoid **type casting and type conversions** as much as possible; because it is a performance penalty.
31. Override ToString (from Object class) method for the types which you want to provide with custom information.
32. Avoid straightaway ***copy/pasting of code*** from other sources. It is always recommended to hand write the code even though if you are referring to the code from some sources. By this, you will get good practice of writing the code yourself and also you will understand the proper usage of that code; finally you will never forget it.
33. Always make it a practice to read **books/articles,**upgrade and follow the *Best Practices* and Guidelines by industry experts like **Microsoft experts** and well-known authors like Martin Fowler, Kent Beck, Jeffrey Ritcher, Ward Cunningham, Scott Hanselman, Scott Guthrie, Donald E Knuth.
34. Verify whether your code have any **memory leakages**. If yes, make sure that they have been fixed.
35. Try attending **technical seminars by experts** to be in touch with the latest software trends and technologies and best practices.
36. Understand thoroughly the **OOPs concepts** and try implementing it in your code.
37. Get to know about your **project design and architecture** to better understand the *flow of your application* as a whole.
38. Take necessary steps to block and avoid any **cross scripting attacks, SQL injection**, and other security holes.
39. Always **encrypt** (by using good **encryption algorithms**) secret/sensitive information like passwords while saving to database and connection strings stored in *web.config* file(s) to avoid manipulation by unauthorized users.
40. Avoid using default keyword for the known types (primitive types) like int, decimal, bool, etc. Most of the times, it should be used in case of Generic types (T) as we may not be sure whether the type is a value type or reference type.
41. Usage of 'out' and 'ref' keywords be avoided as recommended by Microsoft (*in the Code analysis Rules and guidelines*). These keywords are used to pass parameters by reference. Note that 'ref' parameter should be initialized in the calling method before passing to the called method but for 'out' parameter this is not mandatory.