|  |  |
| --- | --- |
| **S. No.** | **Parameter** |
|  | **Code Formatting**   * **Proper Naming Convention** * **Comments in place**    + **No Needless Comments**   + **No Obsolete Comments**   + **No Redundant Comments**   + **Comments in consistent format** * **Readability** * **Eliminating warning** * **Removing all unused using** |
|  | **Coding**   * **Use of constants and configuration values – No hard coded values & magic numbers** * **Use of enums for grouping similar values** * **Following DRY principles** * **Following SOLID principles** * **Proper use of interface(maintain decoupling)** * **Disposing unmanaged resources** * **Eliminating unnecessary features(YAGNI)** * **Code Should be modular** * **Refactoring statements outside loop wherever applicable** * **Null checks wherever applicable** * **Use of framework feature wherever possible.** * **No unreachable code** * **Avoiding nested loops wherever possible** * **Use StringBuilder if multiple concatenations are required to save heap mem.** * **Avoid using out and ref.** * **If mapping is required introduce a mapper component.** |
|  | **Error Handling**   * **Validating files and other input parameters** * **Proper exception logs** * **Handling the edge cases** * **Complete and understandable edge cases** |
|  | **Performance**   * **Replacing code with library or built in functions if applicable** * **Removing logging if it is not required at a particular place.** * **Avoid type casting and type conversions** * **Avoid multiple calls to different layers; try to minimize the number of calls** * **Check if caching can be applied to frequently used data** * **Use asynchronous calls for long running operations** * **Ensure all calls are logged and configurable along with interception for logging; e.g., Use of App Insights** * **If mapping is required introduce a mapper component to system.** |
|  | **REST API**   * **API Versioning** * **Identifying and Hiding unused data from API response.** * **Returning appropriate status code** * **API Security** |
|  | **Security**   * **Use of various appropriate encryption algorithms during interaction with APIs and dependent systems** * **Best practices used for authentication and authorization be it at HTTPModules or be it at handler levels and the use of attributes** * **Check the level of sensitive data and how it is being transmitted** |
|  | **Scalability**   * **Use of patterns and practices that abstract functionality at higher level and have the application extensible for future changes** * **Make proper use of interfaces** * **Ensure the proper implementation of design patterns which do not affect performance** |
|  | **Performance Testing**   * **Use profilers for the code to understand chatty interfaces and long running processes** * **Use performance testing tools to understand the time taken under various levels of load.** |