Lab Exercise: Implementing Exception Logging with NLog in C#

Objective:

Learn to configure and use the NLog framework to log exceptions in C# applications, enhancing debugging and monitoring capabilities.

Prerequisites:

- Basic knowledge of C# and .NET framework
- Visual Studio or any compatible IDE installed
- Basic understanding of exception handling in C#

Exercise Sections:

Part 1: Setting Up NLog

1. **Create a Console Application**
   - Start by creating a new Console Application in Visual Studio.
2. **Install NLog**
   - Use NuGet Package Manager to install the NLog package. You can do this by running `Install-Package NLog` in the Package Manager Console.
3. **Configure NLog**
   - Create an NLog configuration file named `NLog.config` in your project.
   - Define a simple configuration with a console target and a file target. Configure the log level to capture all levels of log messages.

Part 2: Basic Logging

1. **Implement Basic Logging**
   - In your `Main` method, initialize the NLog logger.
   - Add log statements at various levels (Debug, Info, Warn, Error, Fatal) to understand how different levels are logged.
2. **Logging an Exception**
   - Introduce a deliberate error in your code (e.g., divide by zero) to generate an exception.
   - Use a try-catch block to catch the exception and use the logger to log the exception with an error level log message.

Part 3: Advanced Logging Scenarios

1. **Custom Properties in Logs**
   - Modify your logging configuration to include custom properties in the log output (e.g., user ID or session ID).

- Demonstrate logging with these custom properties by adding them to log messages.

2. **Logging with Context**
    - Use NLog's structured logging to include contextual information in your logs (e.g., method name or error code) without explicitly formatting the log message.
3. **Log File Management**
    - Configure the file target in your `NLog.config` to enable log file rotation based on size or time. This ensures that log files are managed efficiently.

Part 4: Analyzing Logs

1. **Review Logged Exceptions**
    - Trigger various exceptions in your application and ensure they are logged appropriately.
    - Review the log files and console output to understand how different types of exceptions are logged.
2. **Advanced Configuration**
    - Experiment with advanced NLog features such as asynchronous logging, archival settings, and customizing log message formats.

Deliverables:

- Source code of the completed Console Application.
- NLog.config file used for configuring NLog.
- A report detailing the logging implementation, including how exceptions are logged and the benefits of using NLog for logging in applications.

Assessment Criteria:

- Correct implementation and configuration of NLog in a C# application.
- Effective logging of exceptions with appropriate log levels.
- Utilization of advanced NLog features to enhance logging capabilities.
- Understanding of log file management and structured logging practices.

This lab exercise will equip you with the practical skills needed to implement efficient logging in your C# applications using NLog, which is crucial for application maintenance, debugging, and monitoring