Daily Coding Assignment Wipro NGA .NET Cohort

Coding Assignment: Unit Testing Basics in .NET

Duration: 30 minutes

Objective: Develop a basic .NET class library and write unit tests using a testing framework.

Problem Statement

Create a simple .NET class library that implements a basic calculator with arithmetic operations. Then, write unit tests for this class using a testing framework of your choice (NUnit, MSTest, or XTest).

User Stories and Expectations

User Story 1: Implement Calculator

- **As a developer**, you need to create a simple calculator class that supports basic arithmetic operations.
- **Acceptance Criteria**:
  - Implement a class `Calculator` with methods for addition, subtraction, multiplication, and division.
  - Methods:
    - `Add(double a, double b)`
    - `Subtract(double a, double b)`
    - `Multiply(double a, double b)`
    - `Divide(double a, double b)`: Handle division by zero with an appropriate exception.

User Story 2: Setup Testing Framework

- **As a developer**, you need to set up a testing framework and write unit tests for the calculator methods.
- **Acceptance Criteria**:
  - Choose a testing framework (NUnit, MSTest, or XTest).
  - Create a test project in the same solution as your class library.
  - Write test cases for each method in the `Calculator` class.

User Story 3: Write Unit Tests

- **As a developer**, you need to write unit tests to verify that the calculator methods work correctly.
- **Acceptance Criteria**:
    - Write test cases to verify:
        - Correct results for valid inputs.
        - Handling of division by zero.
        - Correct results for edge cases (e.g., adding/subtracting zero).
    - Use assertions to validate the test results (e.g., `Assert.AreEqual`, `Assert.IsTrue`).

User Story 4: Test Execution and Reporting

- **As a developer**, you need to run the tests, analyze the results, and handle any failures.
- **Acceptance Criteria**:
    - Run the tests using the chosen framework.
    - Review the test results and ensure all tests pass.
    - If any tests fail, debug and fix the issues.

Assignment Steps

Task 1: Implement Calculator

1. **Create a Calculator Class**:
    - Develop a `Calculator` class with methods for addition, subtraction, multiplication, and division.
    - Ensure that the division method handles division by zero appropriately.

Task 2: Setup Testing Framework

1. **Create a Test Project**:
    - Add a new test project to your solution using your chosen testing framework.
2. **Install the Testing Framework**:
    - For NUnit: Install the `NUnit` and `NUnit3TestAdapter` packages.
    - For MSTest: Install the `MSTest.TestFramework` and `MSTest.TestAdapter` packages.
    - For XTest: Install the `xunit` and `xunit.runner.visualstudio` packages.

Task 3: Write Unit Tests

1. **Create Test Class**:
    - Write test cases for each method in the `Calculator` class to verify functionality.

- Ensure test cases cover valid inputs, division by zero, and edge cases.

Task 4: Test Execution and Reporting

1. **Run Tests**:
   - Use the Test Explorer in Visual Studio or command line tools to execute the tests.
   - Review the results to ensure all tests pass.
2. **Debug and Fix Issues**:
   - If any tests fail, debug the issues in the `Calculator` class and fix them.

Expectations

- **Code Implementation**: Your `Calculator` class should be correctly implemented to perform basic arithmetic operations.
- **Unit Tests**: Write comprehensive tests for each method, including edge cases and exception handling.
- **Testing Framework**: Correctly set up and use the chosen testing framework.
- **Execution**: Ensure that all tests pass and handle any failures appropriately.

This assignment will test your understanding of unit testing principles, ability to use a testing framework, and skill in writing effective unit tests.