Coding Test: Edge Detection Backend in C#

Objective:

Implement a C# project that performs image edge detection using either the Sobel or Prewitt operator, based on user input.

Requirements:

Project Setup:

Create a public GitHub repository and initialize a C# project (e.g., a .NET Console Application or .NET Class Library).

Ensure that all source code, tests, and documentation are committed to this repository.

Functionality:

Implement an image edge detection function that can process a given image and produce an edge-detected output.

The user should be able to choose between Sobel and Prewitt operators before running the edge detection.

Input and output may be handled as image files or as byte arrays representing image data (your choice, but be consistent and document it).

Assume a simple grayscale input for demonstration purposes.

Architecture & Documentation:

Provide a UML class diagram of your solution. This diagram should illustrate the relationships between classes, interfaces, and any other relevant components.

The UML diagram can be in .drawio format or as a static image (e.g., .png) and must be included in the repository.

Write clean, well-structured, and maintainable code.

Include comments or a brief README to explain how to run the code and how your solution is organized.

Testing:

Write unit tests to cover:

- Operator selection logic (ensure the correct operator is applied based on user input).
- Correct handling of image data (test with at least one sample image or mock data).
- Aim for reasonable test coverage and ensure tests are easy to run.

Commit History:

Make incremental and meaningful commits with clear commit messages.

Show your thought process: initial setup, implementing edge detection logic, adding tests, fixes, and refinements.

Deliverables:

A link to the public GitHub repository containing:

- Source code for the edge detection logic.
- Unit test scripts.
- A UML class diagram (in .drawio or image format).
- A brief README with instructions.

Evaluation Criteria:

- Correctness and clarity of the edge detection implementation.
- Quality and organization of the code (e.g., structure, naming, documentation).
- Quality of tests and test coverage.
- Completeness and clarity of the UML diagram.
- Overall project hygiene (commit messages, directory structure, and clarity of documentation).
- This assignment is designed to assess your ability to implement an image processing algorithm in C#, produce clean and testable code, maintain good architectural documentation, and demonstrate effective use of version control.