A Simple Image Processing Library

Develop a simple image processing library using C++. Compared to C, C++ offers more advanced features, making it ideal for implementing a class to represent an image. This project will help you gain a deeper understanding of C++ features. **Ensure that the library is user-friendly and, most importantly, memory-safe.**

Requirements

1. Design a C++ class to represent images. If you are unsure how to design the class, you can refer to the `cv::Mat` class in OpenCV for inspiration.
2. Implement basic image processing functions, such as:
    - Adjusting brightness by adding a fixed value to all pixel values.
    - Blending two images by averaging their pixel values.

    While you do not need to implement many image processing functions, focus on the quality of your source code. The functions should be memory-safe (with proper memory management), robust (handling errors gracefully), and efficient (fast execution). Using SIMD and OpenMP is recommended to improve efficiency.
3. In your report, explain the rationale behind your class and function designs. Highlight the measures you took to ensure memory safety, robustness, and efficiency.
4. Use one `.h` file, several `.cpp` files, and a `CMakeLists.txt` file which is for generating a dynamic library. You should also provide a demo cpp sample to use the dynamic library.
5. You are **encouraged** to use AI tools, such as DeepSeek, to generate a framework for your source code and improve its quality. In your report, describe how you used the tool to enhance your code. However, avoid generating irrelevant or low-quality content.

Rules:

1. The project report and the source code must be submitted before the deadline. Any submission after the deadline (even by 1 second) will result in **a score of 0**. The deadline is 23:59 on May 11.
2. Submit the following files:
    - `report.pdf`
    - One `.h` file
    - Several `.cpp` files for the library
    - `CMakeLists.txt`
    - `demo.cpp`

Avoid submitting too many files. Use the exact filenames and extensions specified. The files should **NOT** be compressed into a single archive.

3. The score will depend on the quality of both the source code and the report. The report should be easy to understand and provide a clear description of the project, especially the highlights.

4. Attention should be paid to code style. Adequate time is given for code to be written correctly and with good style. Deductions will be made for poor code style. Code style guides, such as the Google C++ Style Guide (http://google.github.io/styleguide/cppguide.html), can be used as a reference.