Readme

Image Processing:

> This project aims to perform image processing using C++. The core image processing functionalities and effects are implemented in C++ within the Libraries directory present in the ImageEffectBackend directory. Each effect has a directory and contains a .cpp file which has the code for a specific image processing algorithm, a .h containing the declaration of the function, and an EffectInterafce.cpp file that has JNI implementation for the effect which has to be called from java. We have created the EffectImplement class which implements the required base effects. In the PhotoEffectServices class, there are methods for each effect. The method creates the object of the EffectImplemet class and then sets the attribute. Then the modified image pixel array stores the image generated after the processing.

Image Processing Algorithms:

> **BRIGHTNESS:**
>
> Takes a 2D vector image representing an image and a floating-point amount to adjust the brightness. The function iterates through each pixel in the image, adds the brightness adjustment to each color channel (red, green, and blue), and then clamps the resulting values to ensure they stay within the valid range of [0, 255]. The formula amount = 2.55 * amount - 255 is used to scale and shift the

> brightness level. The adjustments are performed in-place on the input image vector

> **CONTRAST:**
> Modifies the contrast of an image represented by a 2D vector of Pixel objects. It iterates through each pixel, adjusts the contrast based on the provided factor, and ensures the resulting values stay within the valid range of [0, 255]. The contrast adjustment is performed in-place on the input image vector.

> **FLIP:**
> flipHorizontal and flipVertical, flip an image represented by a 2D vector of Pixel objects. **flipHorizontal Function:**
>
> - Iterates through each row, swapping pixels from left and right.
> - Achieves horizontal flipping in-place.
>
> f**lipVertical Function:**
>
>   - Iterates through half of the rows, swapping entire rows between top and bottom halves.
>   - Achieves vertical flipping in-place.

> **GAUSSIANBLUR:**
> applies Gaussian blur to an image represented by a 2D vector of Pixel objects. It calculates a Gaussian kernel based on the specified blur amount, normalizes it, and then performs convolution separately in the horizontal and vertical directions. The function uses a temporary image to store intermediate results during the convolution process. The resulting blurred image is stored in the

> input image vector. The size of the Gaussian kernel is determined based on the specified blur amount, and the convolution is applied to each color channel (red, green, and blue) individually. The function is designed to modify the input image vector in-place.

> **GRAYSCALE:**
> Converts an image represented by a 2D vector of Pixel objects to grayscale. It iterates through each pixel in the image and calculates the grayscale value using the formula:
> **grayscale=0.299×R+0.587×G+0.114×B.** The resulting grayscale value is assigned to each color channel (red, green, and blue) of the pixel, effectively creating a grayscale representation of the original image. The function modifies the input image vector in-place.

> **HUE SATURATION:**
> Adjusts the hue and saturation of an image represented by a 2D vector of Pixel objects. It iterates through each pixel in the image, converts the RGB values to the HSV color space, applies the specified hue and saturation adjustments, and then converts the modified HSV values back to RGB. The function modifies the input image vector in-place.

> **INVERT:**
> inverts the colors of an image represented by a 2D vector of Pixel objects. It iterates through each pixel in the image and performs color inversion by subtracting each RGB component from 255. The function modifies the input image vector in-place.

> **ROTATE:**
> rotate an image represented by a 2D vector of Pixel objects by 90, 180, and 270 degrees clockwise, respectively. The rotations are achieved by transposing the matrix and reversing the order of rows. Additionally, there is a generic rotate function that calls the corresponding specific rotation function based on the specified rotation angle (value). The specific rotation functions modify the input image vector in-place.

> **SEPIA:**
> applies a sepia filter to an image represented by a 2D vector of Pixel objects. It iterates through each pixel in the image and calculates new RGB values based on the specified sepia filter coefficients. The formula for each color channel (red, green, and blue) is applied separately. The resulting values are then clamped to ensure they stay within the valid range of [0, 255]. The function modifies the input image vector in-place, creating a sepia-toned representation of the original image.

> **SHARPEN:**
> Applies a sharpening effect to an image represented by a 2D vector of Pixel objects. It iterates through each pixel in the image (excluding borders) and calculates the sharpened pixel values based on the surrounding pixels. The sharpening is controlled by the amount parameter, and the new pixel values are determined by adjusting the difference between the pixel and its immediate neighbor. The function clamps the resulting values to ensure they stay within the valid range of [0, 255]. The sharpening effect is applied in-place, modifying the input image vector.

Logging Service:

> The class LoggingService is responsible for logging the user activity with the service used for the input image with the selected image effect. The addLog function starts with retrieving the date and time at the time the service is used. It then creates an object of the type LogModel and then adds the relevant data. After it successfully creates the object with the parameter values, it appends the object in the local ArrayList of LogModel objects.

Basic Workflow:

> The process starts with uploading the image from the device. The constraint being that .png images cannot be uploaded. Once the image gets uploaded successfully, the Apply button can be used to apply the effect which generates the new image, available in .png format for download. When the Apply button is clicked, the apply<Name>Effect function gets triggered which uses the '.subscribe' method where the URL with the parameters is posted using the http.post method. This function handles the return value in Blob type. This function uses the corresponding function from the photoEffects object, which parses the parameters required by the effect function in the backend files using the FormData object. The URL is formed by taking the root path from the environment.ts file. This URL request is thus sent to the backend interface.
>
> The PhotoEffectService typescript file links to the PhotoEffectService file in the backend as the typescript file uses the root URL (from 'environments') to call the corresponding function. In

> the backend files, the PhotoEffectService.java returns the image which is received as a Blob type object by the frontend files. In the PhotoEffectService.java file, for a given effect, the parameter values are received by the corresponding variable names whose values are parsed from the PhotoEffectService typescript file in the FormData object type. In PhotoEffectService.java, for each of the effects, the imageFile in the format of MultipartFile is first converted into a 2D array of Pixel object using the ProcessingUtil’s preprocessing function. This array is passed into the implementation class files in the ‘effectImp’ directory which implements one of the interfaces from the ‘baseEffects’ directory. This helps in calling the apply function which changes the values in the 2D array.

Contributions:

> **Subham Agarwala IMT2022110**: PhotoeffectService.java and Effect implementation for contrast, flip, gaussian, grayscale, and hue saturation.

> **Soham Pawar IMT2022127:** Debugging, algorithm for sharpen, hue saturation, and read me.

> **Shashwat Chaturvedi IMT2022118:** Complete loggingServices.java code.

> **Nupur Patil IMT2022520:** Image processing algorithms for Sepia, Brightness, Contrast, and invert

> **Shreeya H IMT2022535:** Image processing algorithms for flip, Gaussian blur, and grayscale and read me.

> **Samyak Jain IMT2022071:** Effect Implementation for brightness, invert, rotation and sharpen effects.