IMAGE EDITOR

- YASH AGARWAL

OVERVIEW

This Image Editor enables the user to edit the image by providing the path and also give the name of the **modified image to be saved.** This feature helps the user to save the image in its desired form and allows it to perform various functions on the image namely:

- Changing the Brightness
- Changing to GrayScale
- Rotating the image clockwise
- Rotating the image anti-clockwise
- Vertical Flip
- Horizontal Flip
- Blurring the Image
- Color Inversion

The project also includes a user-friendly menu interface which gives users options to select from and exit from the program whenever the user wants.The concept of arrays has been used to play with the pixels of the image and change the RGB value of each image.The below given features will give a detailed information about the interface and each of its features.

USER INTERFACE

```
WELCOME!!
Enter the path of the image ::sample.jpg

Enter the path to save the image ::GrayScale

Enter 1 to convert the image to grayscale.
Enter 2 to change the brightness.
Enter 3 to rotate the image clockwise.
Enter 4 to rotate the image anti-clockwise.
Enter 5 to horizontally flip the image.
Enter 6 to vertically flip the image.
Enter 7 to blur the image.
Enter 8 to invert the image.
Enter 0 to exit.
Enter the choice::1
Image changed to GrayScale Successfully!!

Do you want to continue?(Enter Y to continue)Y
Enter 9 to work on the modified image
Enter 10 to work on original image.
Enter 11 to work on new image.
Enter the choice::9

Enter the path to save the image ::Check

Enter 1 to convert the image to grayscale.
Enter 2 to change the brightness.
Enter 3 to rotate the image clockwise.
Enter 4 to rotate the image anti-clockwise.
Enter 5 to horizontally flip the image.
Enter 6 to vertically flip the image.
Enter 7 to blur the image.
Enter 8 to invert the image.
Enter 0 to exit.
Enter the choice::3
Image Rotated clockwise Successfully!!

Do you want to continue?(Enter Y to continue)n
Thank You for visiting!!
```

The user interface is interactive and gives very special feature to the user to decide the name of the image to be saved. It gives a option menu for the user to choose from adding to this it gives the user to work on the modified image , the original image or the new image without closing the program .

At every step user can exit from the program . Also it prints a message which helps the user to know that the operation has been performed successfully.

FEATURES

1.Convert to Grayscale

Algorithm: It involves taking RGB values of each pixel and setting it to the output image whose type is TYPE_BYTE_GRAY. This type changes the pixel to grayscale giving the required output.

```
BufferedImage outputImage = new BufferedImage(width, height,BufferedImage.TYPE_BYTE_GRAY);
```

INPUT IMAGE

OUTPUT IMAGE

2.Changing Brightness

Algorithm : Here using the concept of arrays each pixel's RGB values are stored and then it is increased by the percentage value given by the user and stored in the output image in the corresponding coordinates . The function also keeps the RGB values in check and keeps it in the range of 0-255. Scanner is used to take the percentage as input from the user.

```
BufferedImage outputImage = new BufferedImage(width, height,BufferedImage.TYPE_3BYTE_BGR);
```

INPUT IMAGE

OUTPUT IMAGE

3.Rotating the image

Algorithm : Rotating an image involves a series of operations on the pixel values within the image array. The dimensions of this new array will be the width and height of the original image swapped to account for the clockwise rotation.

```
BufferedImage outputImage = new BufferedImage(height, width,BufferedImage.TYPE_INT_RGB);
```

Iterated through each pixel in the original image array.For all pixels in the original image, copying their values to the corresponding positions in the rotated image.The rotated image array now contains the pixels of the original image rotated clockwise by 90 degrees.

INPUT IMAGE

OUTPUT IMAGE CLOCKWISE

OUTPUT IMAGE ANTI-CLOCKWISE

4.Horizontal Flip

Algorithm: It involves swapping the pixel values between corresponding columns. Iterated through each row of the original image array.While copying pixel values from the original row to the flipped row, the order of the columns are reversed. Started by copying the pixel from the last column of the original row to the first column of the flipped row and so on until reaching the first column of the original row.The flipped image array now contains the pixels of the original image horizontally flipped.

```
BufferedImage outputImage = new BufferedImage(width,height,BufferedImage.TYPE_INT_RGB);
```

INPUT IMAGE

OUTPUT IMAGE

This algorithm creates a horizontally flipped version of the original image by reversing the order of columns in each row, effectively flipping the image along its vertical axis.

5.Vertical Flip

Algorithm: It involves swapping the pixel values between corresponding rows. Iteration is done through each row of the original image array and pixel values are copied from the original row to the flipped row, keeping the order of the columns the same.

```
BufferedImage outputImage = new BufferedImage(width,height,BufferedImage.TYPE_INT_RGB);
```

INPUT IMAGE

OUTPUT IMAGE

This algorithm creates a vertically flipped version of the original image by reversing the order of rows while keeping the order of columns the same, effectively flipping the image along its horizontal axis.

6.Blurring the Image

Algorithm: It involves taking the average of the RGB values of all the pixels in the given area by the user and copying the average pixelated values in all the positions of that given area.For this a function blurSquare has been created which takes input as starting and ending row and column values and blurs that region.

```
public static void blurSquare(BufferedImage inputImage, BufferedImage outputImage, int start_row, int end_row, int start_col, int end_col){
```

```
BufferedImage outputImage = new BufferedImage(width,height,BufferedImage.TYPE_INT_RGB);
```

INPUT IMAGE

OUTPUT IMAGE

This algorithm creates a blurred version of the original image by taking the averaged RGB values of the pixel , effectively blurring the image.

7.Color Inversion

Algorithm: It involves taking the RGB values of each pixel and subtracting the values from 255 and copying the values to the new image array.

```
BufferedImage outputImage = new BufferedImage(width,height,BufferedImage.TYPE_INT_RGB);
```

INPUT IMAGE

OUTPUT IMAGE

CONCLUSION

The journey in developing this image editor project has been exciting. We set out with the goal of creating a versatile tool that empowers users to enhance and transform their images, and we have achieved significant milestones in that regard.

Throughout the project, I successfully implemented a range of essential features, including image loading and saving, basic image adjustments, transformative operations like rotation,changing brightness,blurring,etc. These features offer users a wide array of options for editing and customizing their images.

ACKNOWLEDGEMENT

I would like to extend our heartfelt gratitude to everyone who played a crucial role in the successful completion of this image editor project. Without your support, dedication, and expertise, this project wouldn't have been possible.

I also want to express my appreciation to our mentor Kshitij Sir for his nvaluable guidance and insights. Your expertise and feedback were invaluable in refining the project and pushing it to new heights. I look forward to the continued evolution of this image editor and the exciting possibilities it holds in the world of digital image processing.