**PROJECT REPORT**

TOPIC:

**SIMPLE IMAGE EDITOR**

Name: Majo Augustine

Roll no: 30

MCA

INTRODUCTION:

This Simple Image Editor with Saturation and Brightness Adjustment is a Java application designed to allow users to manipulate the saturation and brightness of images. This project provides a graphical user interface (GUI) built using the Java Swing library, enabling users to choose an image file, adjust its saturation and brightness, and view the modified image.

AIM OF THE PROJECT:

The aim of the Simple Image Editor project is to develop a Java application that enables users to manipulate the saturation and brightness levels of images through a user-friendly graphical interface. This project aims to provide a convenient tool for users to enhance and modify their images according to their preferences without the need for complex image editing software. The primary goal is to create an intuitive and efficient platform for basic image processing tasks, facilitating the adjustment of image attributes such as colour intensity and brightness with ease. Through this project, users can gain practical experience in image processing techniques while enjoying a seamless and interactive editing experience.

FUNCTIONALITIES OF THE PROJECT:

1. **Choose Image:**
   * Users can select an image file from their system using a file chooser dialog.
   * Supported image formats include JPG, JPEG, PNG, and GIF.
2. **Adjust Saturation:**
   * Users can modify the saturation level of the selected image using a slider control.
   * Saturation adjustment allows users to enhance or reduce the intensity of colours in the image.
3. **Adjust Brightness:**
   * Users can adjust the brightness level of the image using another slider control.
   * Brightness adjustment enables users to control the overall lightness or darkness of the image.
4. **Display Modified Image:**
   * The modified image with adjusted saturation and brightness levels is displayed to the user within a separate window.
   * Users can view the changes made to the image in real-time.
5. **Back Button:**
   * Provides the option for users to return to the image selection screen after viewing the modified image.
   * Ensures a smooth and intuitive navigation experience within the application.
6. **Error Handling:**
   * The application includes error handling mechanisms to handle exceptions that may occur during image processing.
   * Users are notified of any errors encountered, such as invalid image files or processing errors.
7. **Intuitive User Interface:**
   * The graphical user interface (GUI) is designed to be user-friendly and intuitive, with clear labels and controls.
   * Users can easily understand and interact with the various functionalities provided by the application.
8. **Real-Time Preview:**
   * As users adjust the saturation and brightness sliders, the modified image is updated in real-time, providing instant feedback on the changes made.
   * Allows users to visualize the effects of saturation and brightness adjustments before applying them to the image.
9. **Cross-Platform Compatibility:**
   * The application is developed in Java, ensuring cross-platform compatibility, and allowing users to run the program on different operating systems without modification.
10. **Scalability and Extensibility:**

* The project codebase is structured in a modular and extensible manner, making it easy to add new features or expand existing functionalities in the future.
* Provides a solid foundation for incorporating additional image processing techniques or advanced editing options.

Top of Form

**SOURCE CODE:**

import javax.swing.\*;

import javax.swing.filechooser.FileNameExtensionFilter;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.image.BufferedImage;

import java.io.File;

import javax.imageio.ImageIO;

import java.awt.image.RescaleOp;

public class CISaturationGUI extends JFrame {

    private JFrame chooseImageFrame;

    private JFrame adjustSaturationFrame;

    private JFrame adjustBrightnessFrame;

    private JFrame displayImageFrame;

    private JLabel inputImagePathLabel;

    private JTextField inputImagePathField;

    private JButton chooseImageButton;

    private JLabel saturationFactorLabel;

    private JSlider saturationFactorSlider;

    private JButton adjustSaturationButton;

    private JLabel brightnessLabel;

    private JSlider brightnessSlider;

    private JButton adjustBrightnessButton;

    private JLabel outputImageLabel;

    private JButton backButton;

    private BufferedImage outputImage;

    public CISaturationGUI() {

        createChooseImageFrame();

        createAdjustSaturationFrame();

        createAdjustBrightnessFrame();

        createDisplayImageFrame();

        chooseImageFrame.setVisible(true);

    }

    private void createChooseImageFrame() {

        chooseImageFrame = new JFrame("Image Editor");

        chooseImageFrame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        chooseImageFrame.setLayout(new GridLayout(5, 5));

        chooseImageFrame.getContentPane().setBackground(Color.YELLOW);

        inputImagePathLabel = new JLabel("Set Image Path:");

        inputImagePathField = new JTextField();

        chooseImageButton = new JButton("Choose Image");

        chooseImageButton.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                JFileChooser fileChooser = new JFileChooser();

                FileNameExtensionFilter filter = new FileNameExtensionFilter("Images", "jpg", "jpeg", "png", "gif");

                fileChooser.setFileFilter(filter);

                int returnVal = fileChooser.showOpenDialog(chooseImageFrame);

                if (returnVal == JFileChooser.APPROVE\_OPTION) {

                    File file = fileChooser.getSelectedFile();

                    inputImagePathField.setText(file.getAbsolutePath());

                    chooseImageFrame.setVisible(false);

                    adjustSaturationFrame.setVisible(true);

                }

            }

        });

        chooseImageFrame.add(inputImagePathLabel);

        chooseImageFrame.add(inputImagePathField);

        chooseImageFrame.add(chooseImageButton);

        chooseImageFrame.pack();

        chooseImageFrame.setLocationRelativeTo(null);

    }

    private void createAdjustSaturationFrame() {

        adjustSaturationFrame = new JFrame("Adjust Saturation");

        adjustSaturationFrame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        adjustSaturationFrame.setLayout(new GridLayout(5, 5));

        adjustSaturationFrame.getContentPane().setBackground(Color.GREEN);

        saturationFactorLabel = new JLabel("Saturation Factor:");

        saturationFactorSlider = new JSlider(0, 200, 100);

        saturationFactorSlider.setMajorTickSpacing(10);

        saturationFactorSlider.setPaintTicks(true);

        saturationFactorSlider.setPaintLabels(true);

        adjustSaturationButton = new JButton("Adjust Saturation");

        adjustSaturationButton.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                String inputImagePath = inputImagePathField.getText();

                double saturationFactor = saturationFactorSlider.getValue() / 100.0;

                try {

                    BufferedImage inputImage = ImageIO.read(new File(inputImagePath));

                    outputImage = adjustSaturation(inputImage, saturationFactor);

                    adjustSaturationFrame.setVisible(false);

                    adjustBrightnessFrame.setVisible(true);

                } catch (Exception ex) {

                    JOptionPane.showMessageDialog(adjustSaturationFrame, "Error processing image: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

                }

            }

        });

        adjustSaturationFrame.add(saturationFactorLabel);

        adjustSaturationFrame.add(saturationFactorSlider);

        adjustSaturationFrame.add(adjustSaturationButton);

        adjustSaturationFrame.pack();

        adjustSaturationFrame.setLocationRelativeTo(null);

    }

    private void createAdjustBrightnessFrame() {

        adjustBrightnessFrame = new JFrame("Adjust Brightness");

        adjustBrightnessFrame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        adjustBrightnessFrame.setLayout(new GridLayout(5, 5));

        adjustBrightnessFrame.getContentPane().setBackground(Color.BLUE);

        brightnessLabel = new JLabel("Brightness:");

        brightnessSlider = new JSlider(-255, 255, 0);

        brightnessSlider.setMajorTickSpacing(50);

        brightnessSlider.setPaintTicks(true);

        brightnessSlider.setPaintLabels(true);

        adjustBrightnessButton = new JButton("Adjust Brightness");

        adjustBrightnessButton.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                int brightnessValue = brightnessSlider.getValue();

                try {

                    outputImage = adjustBrightness(outputImage, brightnessValue);

                    displayImageFrame.repaint();

                    displayImageFrame.revalidate();

                    adjustBrightnessFrame.setVisible(false);

                    displayImageFrame.setVisible(true);

                } catch (Exception ex) {

                    JOptionPane.showMessageDialog(adjustBrightnessFrame, "Error adjusting brightness: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

                }

            }

        });

        adjustBrightnessFrame.add(brightnessLabel);

        adjustBrightnessFrame.add(brightnessSlider);

        adjustBrightnessFrame.add(adjustBrightnessButton);

        adjustBrightnessFrame.pack();

        adjustBrightnessFrame.setLocationRelativeTo(null);

    }

    private void createDisplayImageFrame() {

        displayImageFrame = new JFrame("Display Image");

        displayImageFrame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        outputImageLabel = new JLabel();

        backButton = new JButton("Back");

        backButton.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                displayImageFrame.setVisible(false);

                chooseImageFrame.setVisible(true);

            }

        });

        JScrollPane outputImageScrollPane = new JScrollPane(outputImageLabel);

        displayImageFrame.add(outputImageScrollPane, BorderLayout.CENTER);

        displayImageFrame.add(backButton, BorderLayout.SOUTH);

        displayImageFrame.pack();

        displayImageFrame.setLocationRelativeTo(null);

    }

    private BufferedImage adjustSaturation(BufferedImage image, double saturationFactor) {

        int width = image.getWidth();

        int height = image.getHeight();

        BufferedImage adjustedImage = new BufferedImage(width, height, BufferedImage.TYPE\_INT\_RGB);

        for (int y = 0; y < height; y++) {

            for (int x = 0; x < width; x++) {

                Color color = new Color(image.getRGB(x, y));

                float[] hsb = Color.RGBtoHSB(color.getRed(), color.getGreen(), color.getBlue(), null);

                // Adjust saturation

                hsb[1] \*= saturationFactor;

                hsb[1] = Math.min(hsb[1], 1.0f); // Ensure saturation is in the valid range

                int rgb = Color.HSBtoRGB(hsb[0], hsb[1], hsb[2]);

                adjustedImage.setRGB(x, y, rgb);

            }

        }

        ImageIcon icon = new ImageIcon(adjustedImage);

        outputImageLabel.setIcon(icon);

        outputImageLabel.repaint();

        outputImageLabel.revalidate();

        return adjustedImage;

    }

    private BufferedImage adjustBrightness(BufferedImage image, int brightnessValue) {

        RescaleOp op = new RescaleOp(1.0f, brightnessValue, null);

        BufferedImage adjustedImage = op.filter(image, null);

        return adjustedImage;

    }

    public static void main(String[] args) {

        SwingUtilities.invokeLater(new Runnable() {

            @Override

            public void run() {

                new CISaturationGUI();

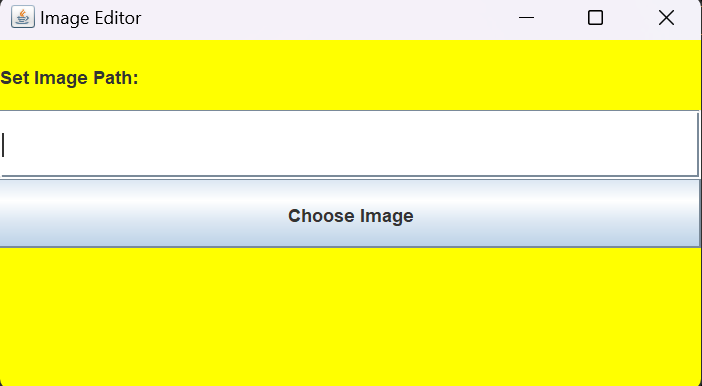
            }

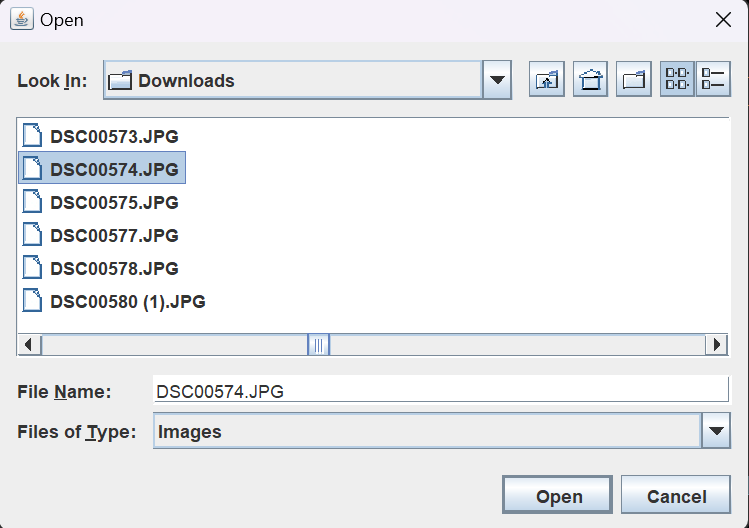
        });

    }

}

**SCREENSHOTS:**

****

****

