

Image Processing GUI Toolkit

1. Problem Statement

The objective of this project is to design and implement a user-friendly graphical interface for performing basic image processing operations such as enhancement, filtering, and transformation. The toolkit allows users to upload, process, and analyze images without requiring deep programming knowledge.

2. Introduction

Image processing is an essential domain in computer vision and artificial intelligence. This project introduces a GUI-based toolkit developed using Python, OpenCV, and Streamlit, allowing users to perform real-time image operations. The interface provides functionalities like histogram equalization, filtering, edge detection, and more.

3. Requirements

The system requires Python 3.10 or above along with essential libraries like OpenCV, NumPy, Pillow, and Streamlit. Hardware requirements include a standard PC with minimum 4GB RAM. The toolkit is lightweight and runs in any modern browser.

4. Methodology

The methodology involves four main steps: 1. Uploading an image using the GUI file uploader. 2. Selecting operations such as enhancement, histogram equalization, or edge detection. 3. Applying transformations using OpenCV and NumPy functions. 4. Displaying results side-by-side for comparison. Streamlit is used for frontend interface, while OpenCV handles the backend image processing tasks.

5. Results

The following screenshots demonstrate the working of the Image Processing Toolkit. Users can upload an image, perform operations such as Histogram Equalization, and view processed results instantly.

Original vs Processed Images:

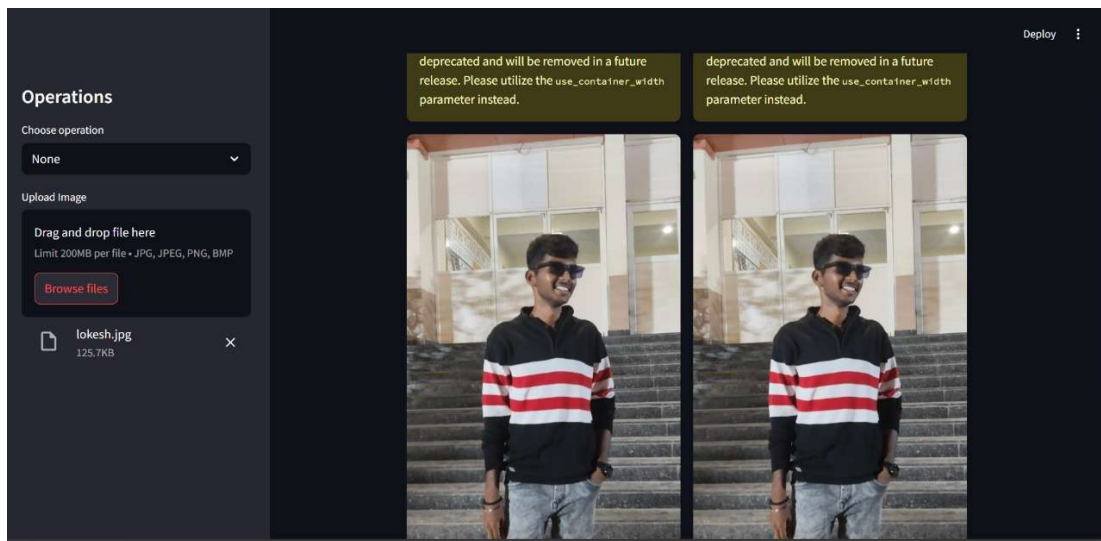
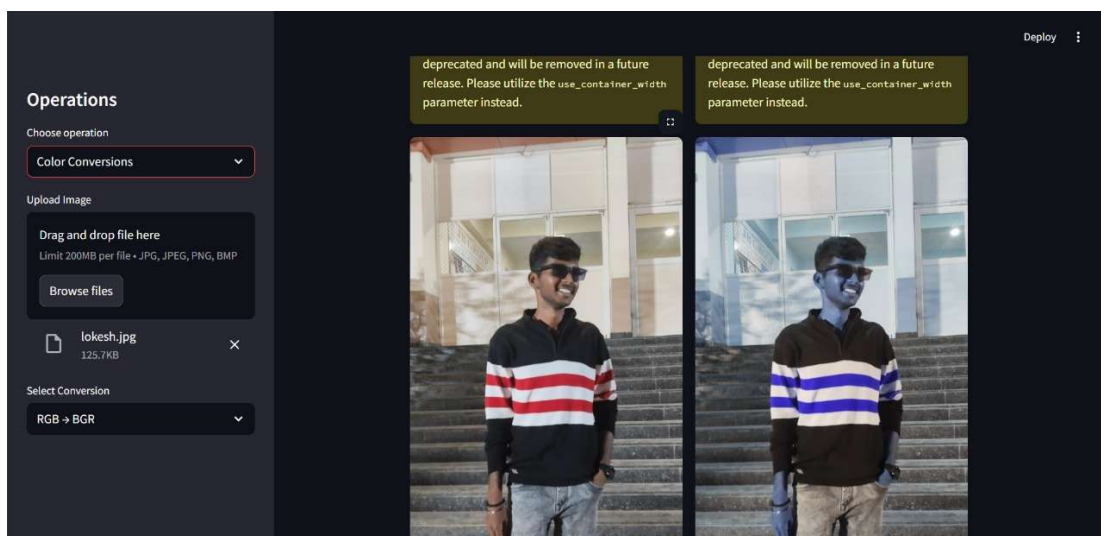
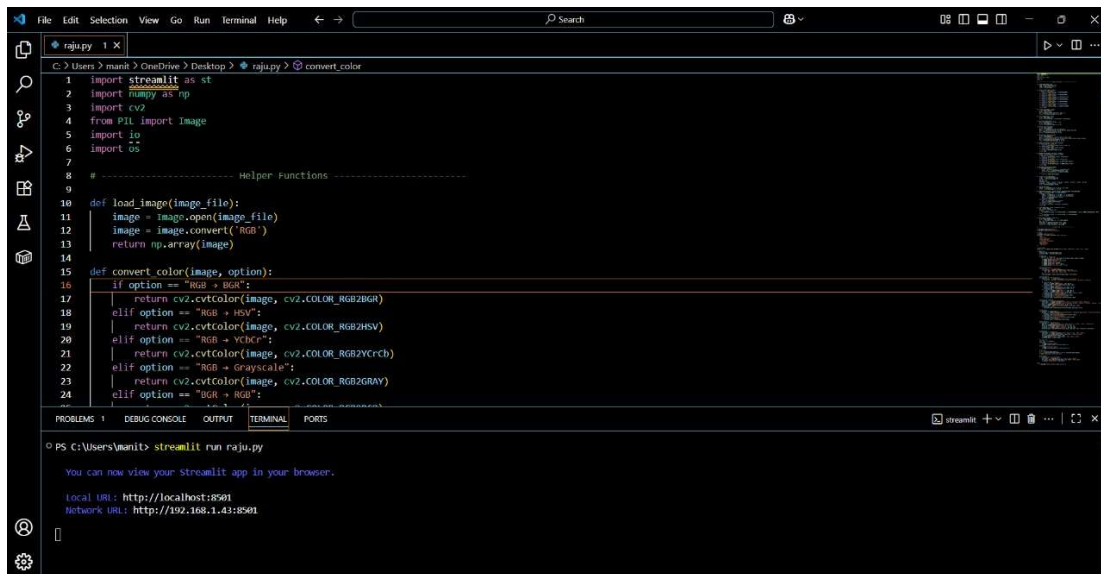


Image Information Output:



Code Execution in VS Code:



The screenshot shows the Visual Studio Code interface with a Python file named `raj.py` open. The file contains a Streamlit application that loads an image and converts its color space based on a user input. The code is as follows:

```
1 import streamlit as st
2 import numpy as np
3 import cv2
4 from PIL import Image
5 import io
6 import os
7
8 # ----- Helper Functions -----
9
10 def load_image(image_file):
11     image = image.open(image_file)
12     image = image.convert('RGB')
13     return np.array(image)
14
15 def convert_color(image, option):
16     if option == "BGR + BGR":
17         return cv2.cvtColor(image, cv2.COLOR_BGR2BGR)
18     elif option == "RGB + HSV":
19         return cv2.cvtColor(image, cv2.COLOR_RGB2HSV)
20     elif option == "RGB + YCrCb":
21         return cv2.cvtColor(image, cv2.COLOR_RGB2YCrCb)
22     elif option == "RGB + Grayscale":
23         return cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
24     elif option == "BGR + RGB":
25         return cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

The bottom panel of the VS Code window shows the terminal output of the command `streamlit run raj.py`. The output indicates that the Streamlit app is running and provides the local and network URLs to view the application in a browser.

```
PS C:\Users\manik> streamlit run raj.py
You can now view your Streamlit app in your browser.
Local URL: http://localhost:8501
Network URL: http://192.168.1.43:8501
```

6. Conclusion

The Image Processing GUI Toolkit successfully demonstrates how complex image operations can be simplified using a user-friendly interface. It eliminates the need for in-depth programming, making it useful for beginners, researchers, and students exploring image processing techniques.

