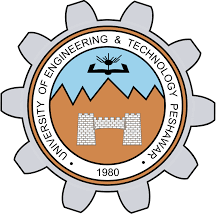
PROJECT REPORT



GROUP MEMBER’S: WADAN KHAN (23PWCSE2290) HUMAIRA NAZ (23PWCSE2315) MUHAMMAD HAMZA (23PWCSE2274)

SUBMITTED TO :

Dr. SUMAYYEA SALAHUDDIN

# Project Report: Image Processing System

1. **Introduction** The Image Processing System is a Python-based application that allows users to apply various transformations and enhancements to an image. The program utilizes the Pillow (PIL) library to perform operations such as filtering, resizing, rotating, and contrast enhancement.

# Objectives

* + To provide a user-friendly interface for basic image processing operations.
  + To enable users to apply predefined filters to images.
  + To allow resizing, rotating, and contrast enhancement of images.
  + To automate the process of opening, modifying, and saving images in a specified directory.

# Features and Functionalities

* + **Opening an Image:** The system loads an image from a specified input path.
  + **Applying Filters:** Users can choose from a set of predefined filters (BLUR, CONTOUR, DETAIL, SHARPEN, BLACK\_AND\_WHITE).
  + **Resizing:** The system allows users to resize an image to specified dimensions.
  + **Rotating:** Users can rotate an image by a specified angle.
  + **Enhancing Contrast:** The system allows contrast enhancement using a user-defined factor.
  + **Saving the Image:** The processed image is saved to a specified output path.

# Implementation Details

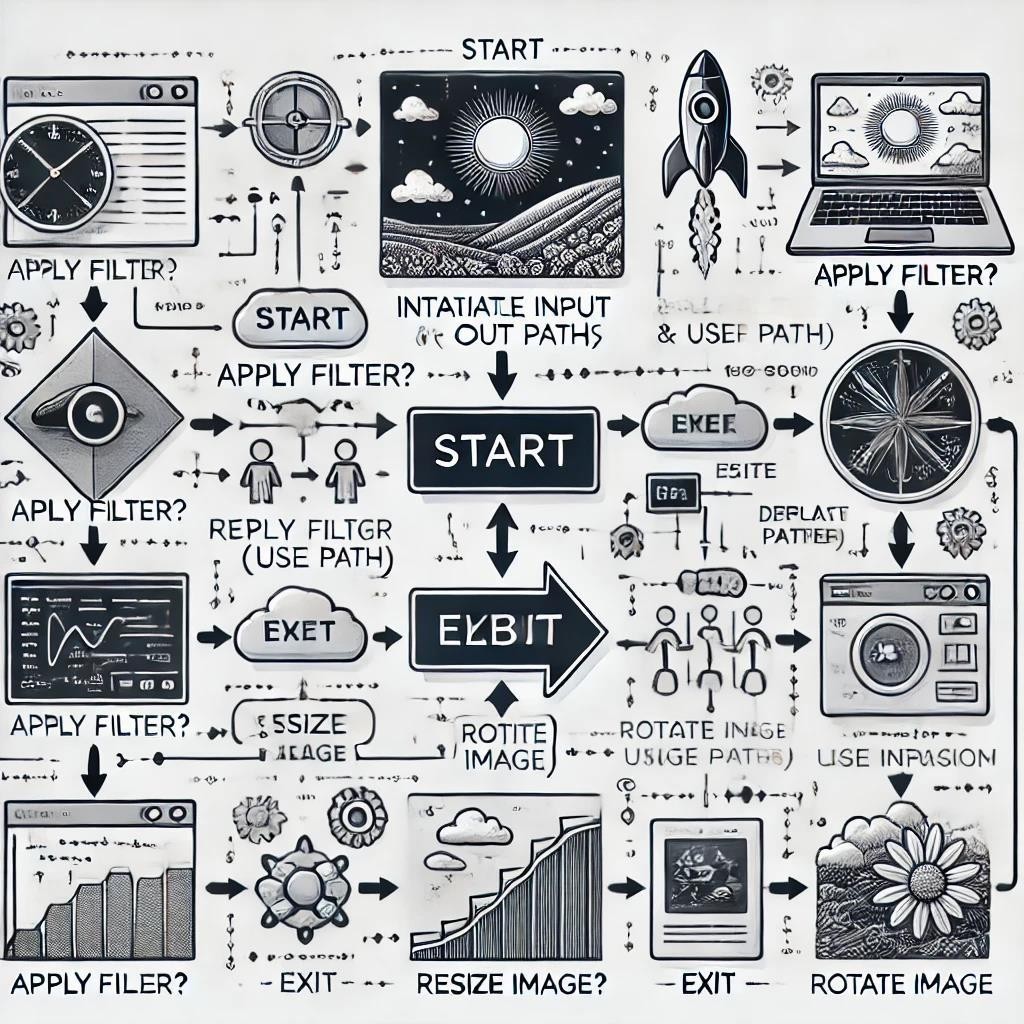
* + The system is implemented in Python and uses the **Pillow (PIL)**

library for image processing.

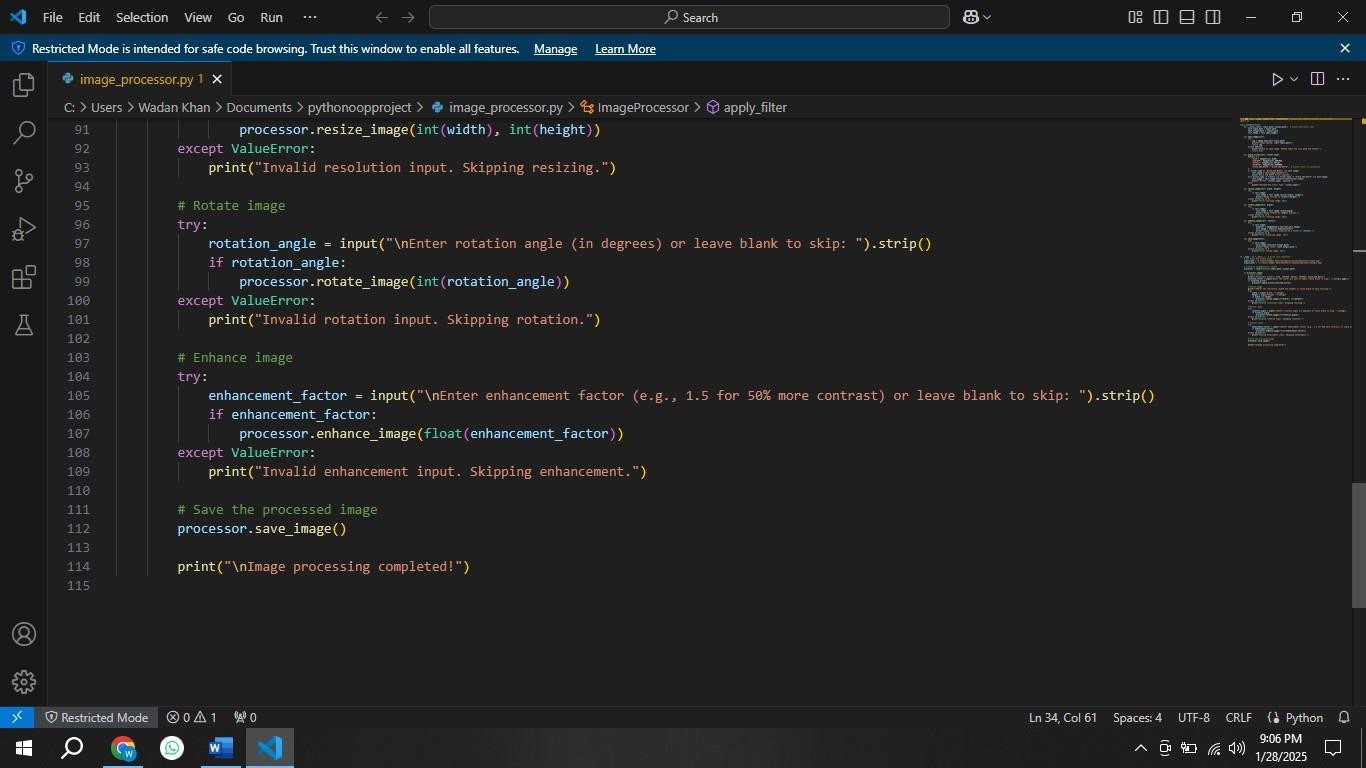
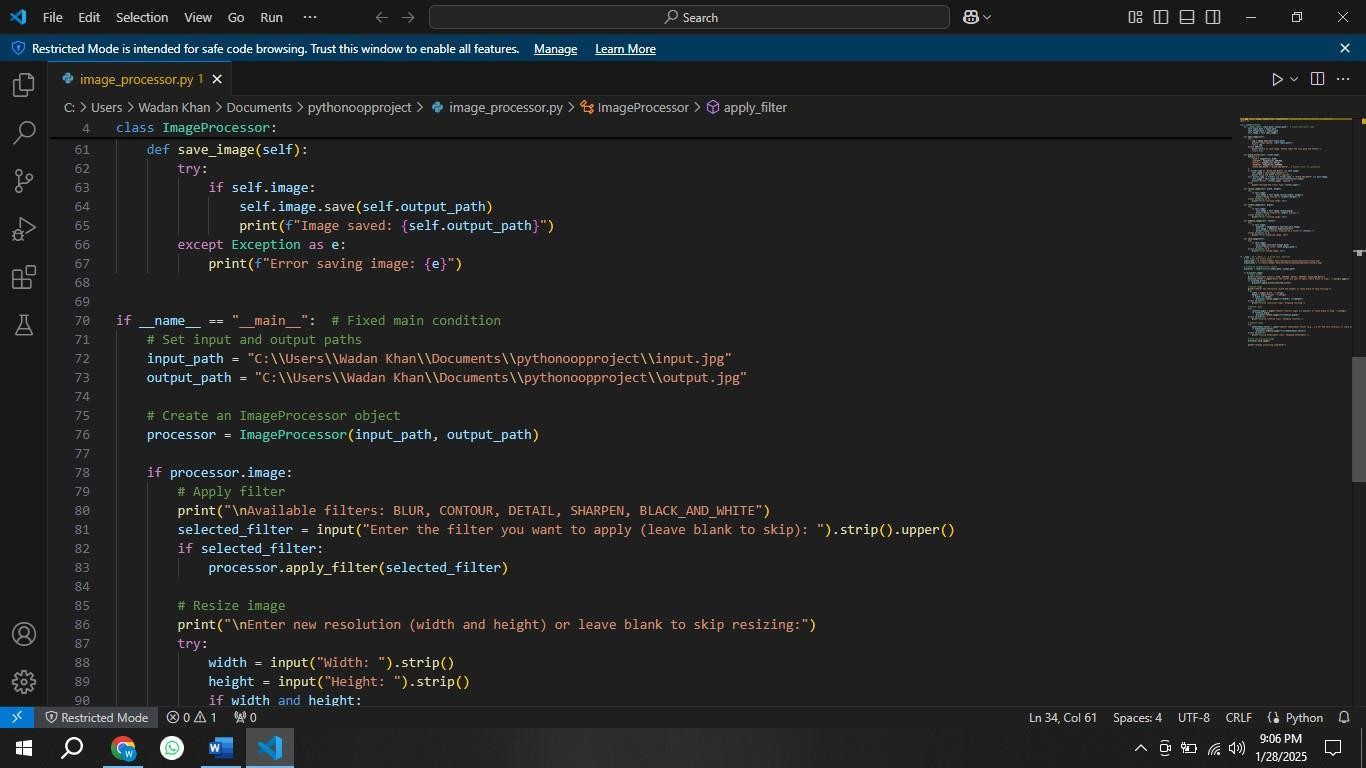
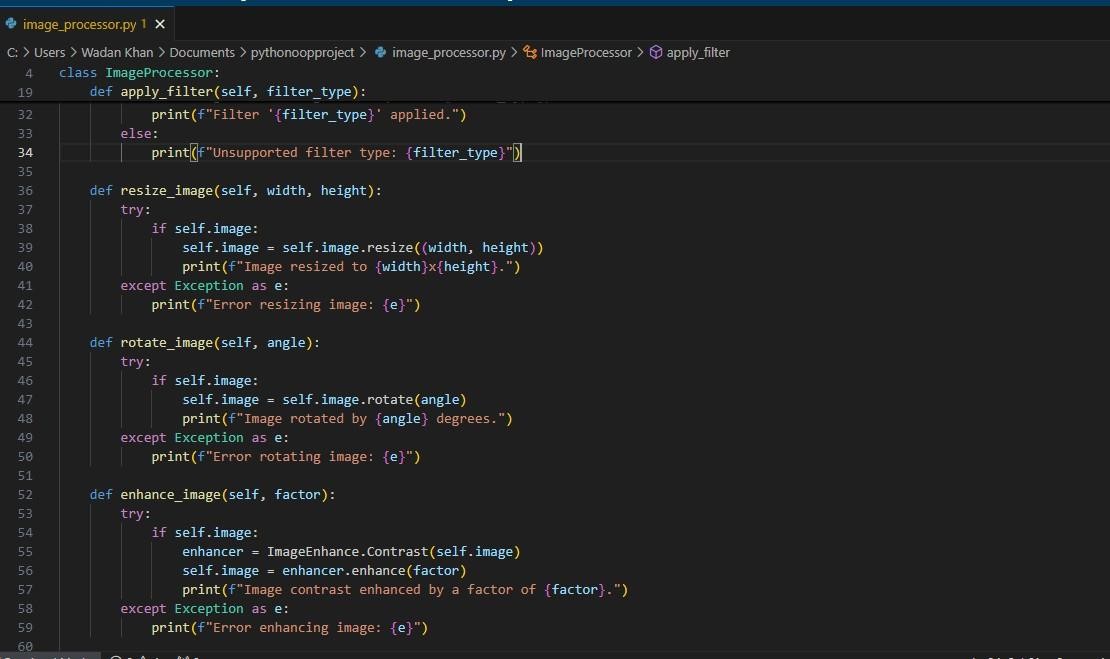
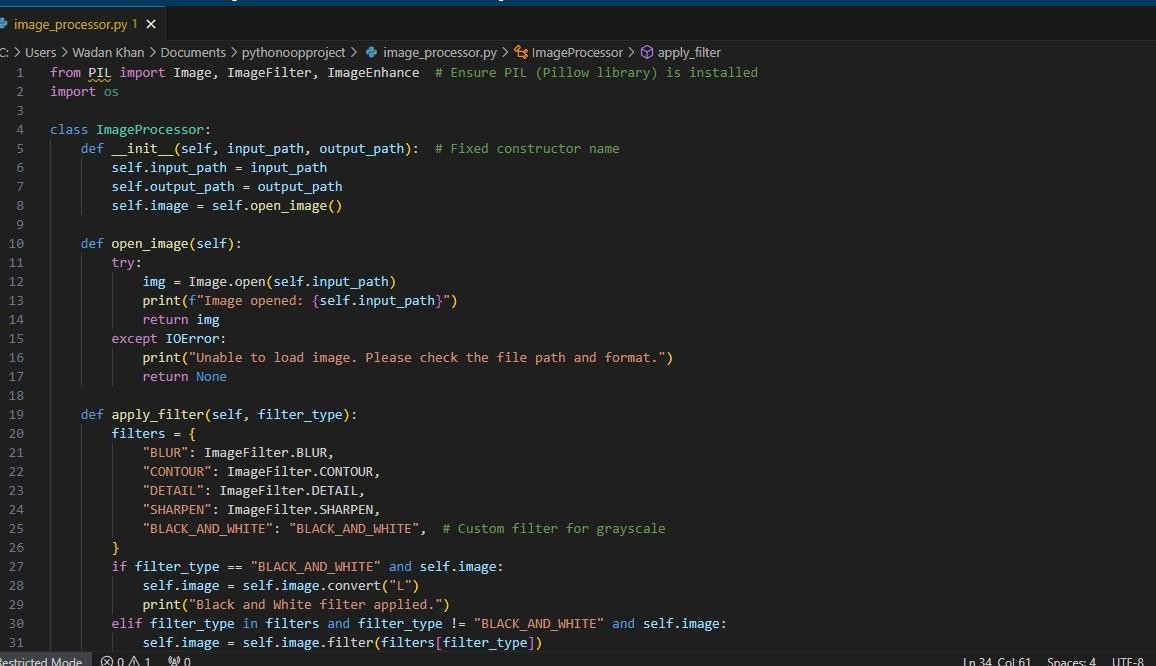
* + The **ImageProcessor** class encapsulates all functionalities, ensuring modular and reusable code.
  + The program employs exception handling to manage errors such as invalid inputs or file-related issues.
  + A command-line interface is provided to allow user interaction.

# Flowchart Breakdown

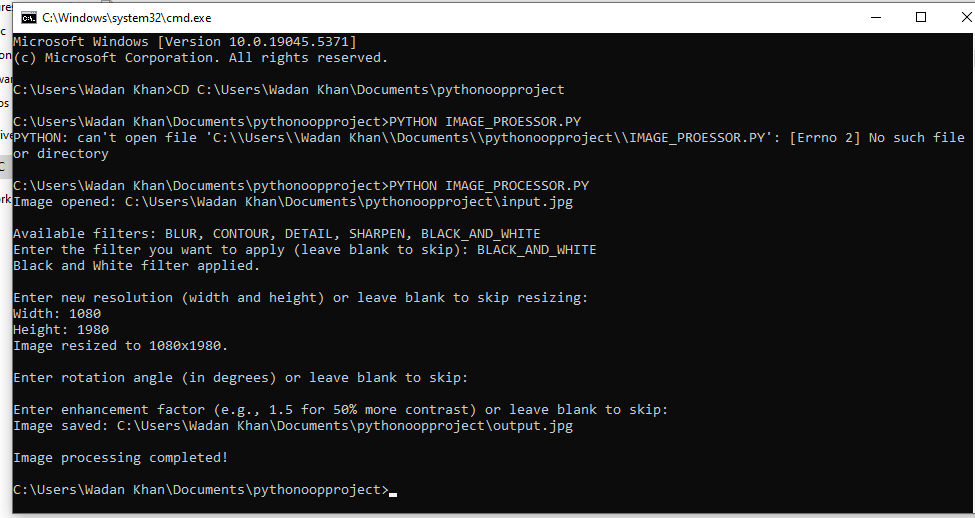
1. **Start** → Initialize input and output paths.
2. **Open Image** → Try to open the image; if unsuccessful, exit.
3. **Apply Filter (Optional)** → If a filter is selected, apply it.
4. **Resize Image (Optional)** → If dimensions are provided, resize the image.
5. **Rotate Image (Optional)** → If an angle is given, rotate the image.
6. **Enhance Image (Optional)** → If a contrast factor is provided, enhance the image.
7. **Save Image** → Save the modified image.
8. **End** → Display completion message.



CODE:



# OUTPUT:





**EXPLANATION:**

# Import Required Libraries

python CopyEdit

from PIL import Image, ImageFilter, ImageEnhance # Ensure PIL

(Pillow library) is installed import os

* + **Pillow (PIL)**: A powerful Python library for image processing.
  + Image: Used to open and manipulate images.
  + ImageFilter: Provides predefined filters like BLUR, SHARPEN, etc.
  + ImageEnhance: Allows modifying image properties such as contrast.
  + os: Included for potential file path operations (though not used in this case).

# Defining the ImageProcessor Class

python CopyEdit

class ImageProcessor:

def init (self, input\_path, output\_path): # Fixed constructor name

self.input\_path = input\_path self.output\_path = output\_path self.image = self.open\_image()

* + The init method initializes an ImageProcessor object with:
    - input\_path: Path of the original image.
    - output\_path: Path where the modified image will be saved.
    - Calls open\_image() to load the image.

# Opening an Image

python CopyEdit

def open\_image(self): try:

img = Image.open(self.input\_path) print(f"Image opened: {self.input\_path}") return img

except IOError:

print("Unable to load image. Please check the file path and format.")

return None

* + Opens the image using Image.open(self.input\_path).
  + If successful, returns the image.
  + If an error occurs (e.g., file not found or wrong format), prints an error message and returns None.

# Applying Filters

python CopyEdit

def apply\_filter(self, filter\_type): filters = {

"BLUR": ImageFilter.BLUR, "CONTOUR": ImageFilter.CONTOUR, "DETAIL": ImageFilter.DETAIL, "SHARPEN": ImageFilter.SHARPEN,

"BLACK\_AND\_WHITE": "BLACK\_AND\_WHITE", # Custom filter for grayscale

}

* + A dictionary filters maps filter names to corresponding PIL filter objects.
  + "BLACK\_AND\_WHITE" is a special case that requires grayscale conversion.

python CopyEdit

if filter\_type == "BLACK\_AND\_WHITE" and self.image:

self.image = self.image.convert("L") # Convert image to grayscale print("Black and White filter applied.")

elif filter\_type in filters and filter\_type != "BLACK\_AND\_WHITE" and self.image:

self.image = self.image.filter(filters[filter\_type]) # Apply standard filter

print(f"Filter '{filter\_type}' applied.") else:

print(f"Unsupported filter type: {filter\_type}")

* + If "BLACK\_AND\_WHITE" is selected, the image is converted to grayscale ("L" mode).
  + If another valid filter is selected, it is applied using self.image.filter().
  + If the filter type is invalid, an error message is shown.

# Resizing the Image

python CopyEdit

def resize\_image(self, width, height): try:

if self.image:

self.image = self.image.resize((width, height))

print(f"Image resized to {width}x{height}.") except Exception as e:

print(f"Error resizing image: {e}")

* Resizes the image to the given width and height using self.image.resize().
* If an error occurs (e.g., non-integer input), it catches the exception and prints an error message.

# Rotating the Image

python CopyEdit

def rotate\_image(self, angle): try:

if self.image:

self.image = self.image.rotate(angle) print(f"Image rotated by {angle} degrees.")

except Exception as e: print(f"Error rotating image: {e}")

* + Rotates the image by the specified angle using self.image.rotate().
  + Prints an error message if the operation fails.

# Enhancing Image Contrast

python CopyEdit

def enhance\_image(self, factor): try:

if self.image:

enhancer = ImageEnhance.Contrast(self.image) self.image = enhancer.enhance(factor)

print(f"Image contrast enhanced by a factor of {factor}.") except Exception as e:

print(f"Error enhancing image: {e}")

* Uses ImageEnhance.Contrast to modify the contrast of the image.
* A factor greater than 1 increases contrast, while a value between 0 and 1 decreases it.

# Saving the Processed Image

python CopyEdit

def save\_image(self): try:

if self.image:

self.image.save(self.output\_path) print(f"Image saved: {self.output\_path}")

except Exception as e: print(f"Error saving image: {e}")

* + Saves the modified image to output\_path using self.image.save().
  + If an error occurs, it prints an error message.

# Running the Program

python CopyEdit

if name == " main ": # Fixed main condition

* + Ensures that the script only runs if executed directly (not imported as a module).

# User Input for Processing Steps

python CopyEdit

input\_path = "C:\\Users\\Wadan Khan\\Documents\\pythonoopproject\\input.jpg"

output\_path = "C:\\Users\\Wadan Khan\\Documents\\pythonoopproject\\output.jpg"

processor = ImageProcessor(input\_path, output\_path)

* + Specifies input and output image paths.
  + Creates an ImageProcessor object.

# Applying a Filter (User Input)

python CopyEdit

print("\nAvailable filters: BLUR, CONTOUR, DETAIL, SHARPEN, BLACK\_AND\_WHITE")

selected\_filter = input("Enter the filter you want to apply (leave blank to skip): ").strip().upper()

if selected\_filter: processor.apply\_filter(selected\_filter)

* + Displays available filters.
  + If the user enters a filter name, it is applied.

# Resizing the Image

python CopyEdit

print("\nEnter new resolution (width and height) or leave blank to skip resizing:")

try:

width = input("Width: ").strip() height = input("Height: ").strip()

if width and height: processor.resize\_image(int(width), int(height))

except ValueError:

print("Invalid resolution input. Skipping resizing.")

* + Asks for new width and height.
  + Converts input to integers and resizes the image.
  + If input is invalid, resizing is skipped.

# Rotating the Image

python CopyEdit try:

rotation\_angle = input("\nEnter rotation angle (in degrees) or leave blank to skip: ").strip()

if rotation\_angle: processor.rotate\_image(int(rotation\_angle))

except ValueError:

print("Invalid rotation input. Skipping rotation.")

* + Asks for a rotation angle and applies it.

# Enhancing Image Contrast

python CopyEdit

try:

enhancement\_factor = input("\nEnter enhancement factor (e.g.,

* 1. for 50% more contrast) or leave blank to skip: ").strip() if enhancement\_factor:

processor.enhance\_image(float(enhancement\_factor)) except ValueError:

print("Invalid enhancement input. Skipping enhancement.")

* + - Asks for an enhancement factor and modifies the contrast.

# Saving the Image

python CopyEdit

processor.save\_image()

print("\nImage processing completed!")

* + - Saves the processed image.
    - Prints a success message.

# Summary of Features

⬛✓ Open and process an image

⬛✓ Apply filters (BLUR, CONTOUR, DETAIL, SHARPEN, BLACK\_AND\_WHITE)

⬛✓ Resize the image

⬛✓ Rotate the image

⬛✓ Enhance contrast

⬛✓ Save the modified image