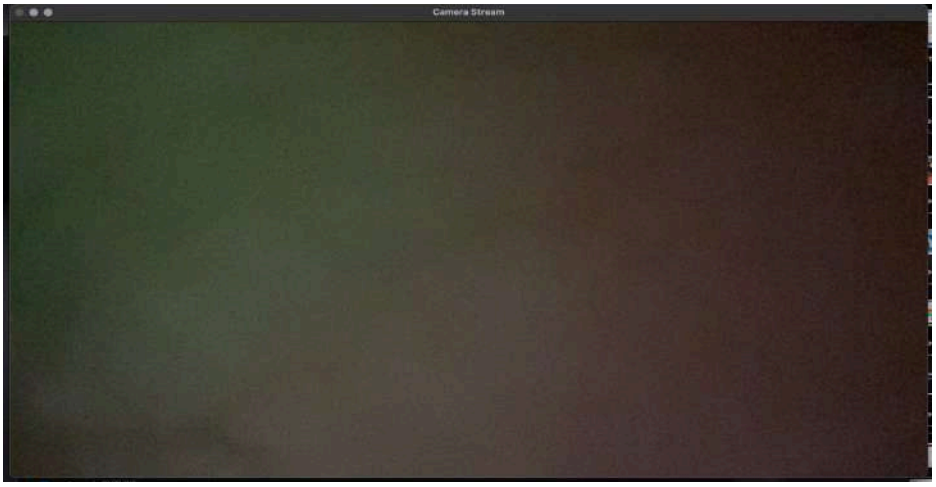
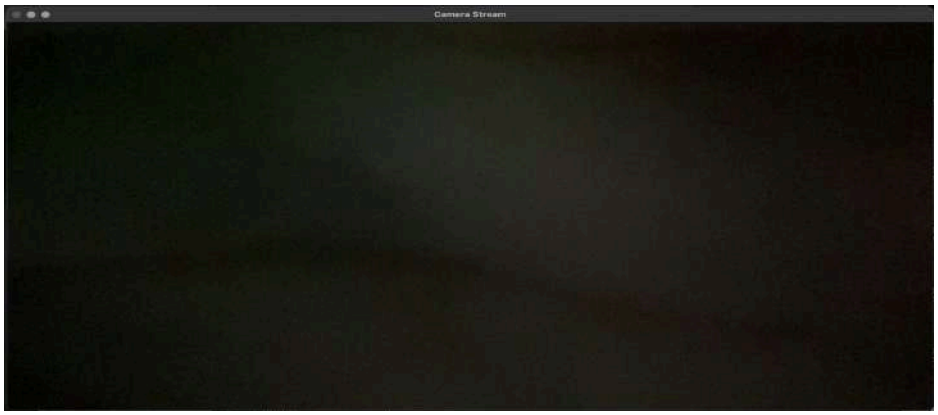
## CODE 1: (code_1.py)

- **Concept used:** Real-time video capture and display using OpenCV (`cv2.VideoCapture`).
- **`OBSERVATION:`**
- A live camera feed window opens and it continuously shows  frames until the user exits.
- 


## CODE 2 : (code_2.py)

- **Concept used:** Capturing and saving video frames using OpenCV (`cv2.imwrite`).
- **OBSERVATION:**
- The camera stream displays, and each frame is automatically saved as an image (`frame_000000.jpg`, etc.) in the framesfolder until user quits.
- 

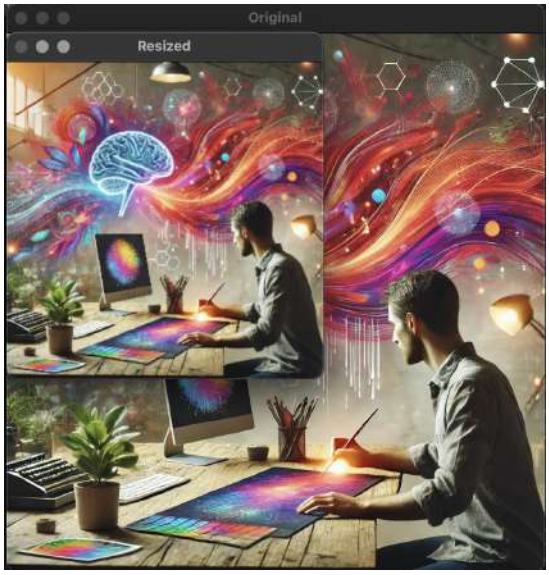
## CODE 3 : (code_3.py)

- **Concept used:** Reading and displaying an image using OpenCV (`cv2.imread`, `cv2.imshow`).
- **OBSERVATION:**
- The selected image opens in a window and remains visible until any key is pressed, after which the window closes.
- 


## CODE 4: (code_4.py)

- **Concept used:** Image transformation using OpenCV's `cv2.flip` function.
- **OBSERVATION:**
- The original image is shown along with flipped versions — vertically, horizontally, and both — to visualize how flipping affects orientation.
- 

## CODE 5 ( code_5.py)

- **Concept used:** Image resizing using OpenCV (`cv2.resize`).
- **OBSERVATION:**
- The original image and its resized (300×300) version are displayed side by side, and the resized image can also be saved as a new file.
- 


## CODE 6 (code_6.py)

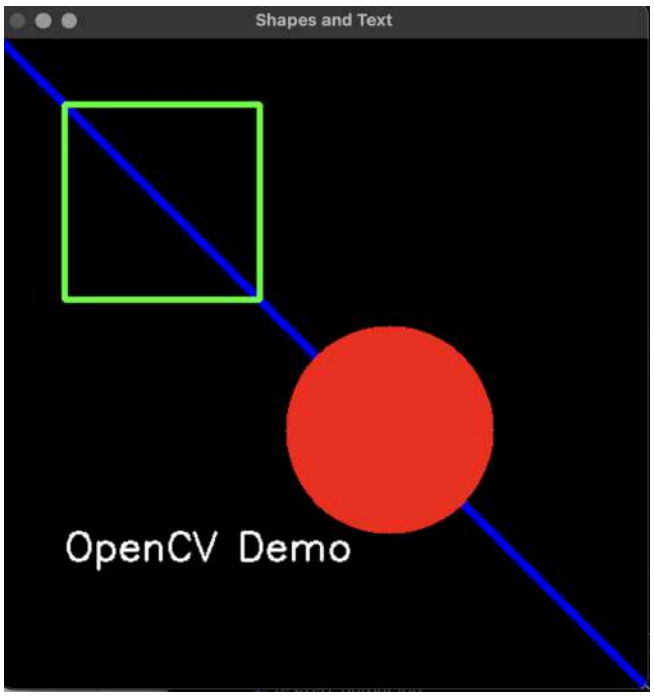
- **Concept used:** Color space conversion using OpenCV (`cv2.cvtColor`).
- **OBSERVATION:**
- The original colored image and its grayscale version are displayed, highlighting the loss of color information but retention of structure/details.

- 


## CODE 7 : (code_7.py)

- **Concept used:** Image smoothing using Gaussian Blur (`cv2.GaussianBlur`).
- **OBSERVATION**
- The blurred version of the image appears softer, with reduced noise and less sharp edges compared to the original.
- 

## CODE 8: (code_8.py)

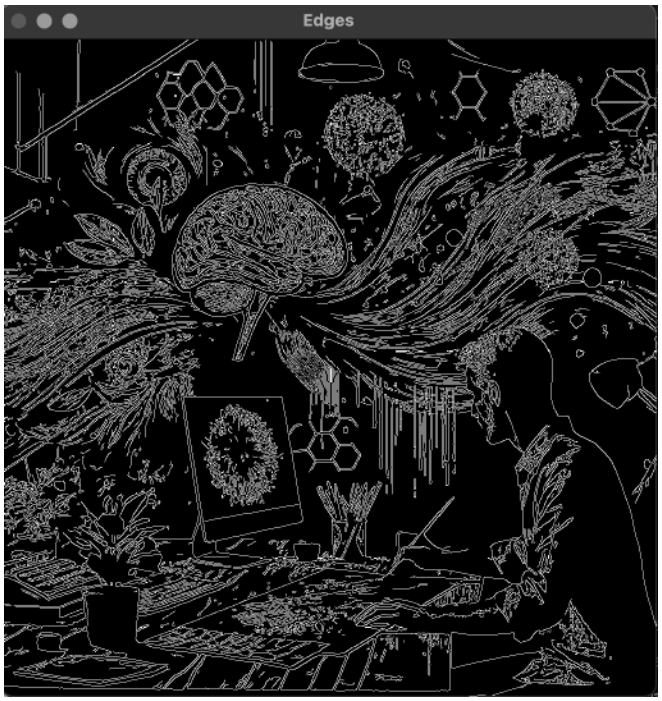
- **Concept used:** Drawing shapes and text on an image using OpenCV (`cv2.line`, `cv2.rectangle`, `cv2.circle`, `cv2.putText`).
- **OBSERVATION:**
- A blank canvas is displayed with a blue line, green rectangle, filled red circle, and white text, showing how graphics can be overlaid on images.
- 


## CODE 9: (code_9.py)

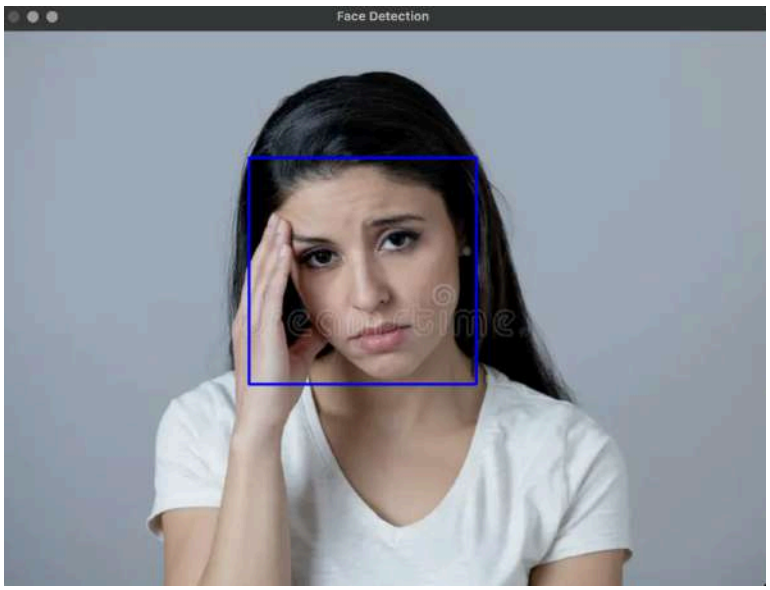
- **Concept used:** Image thresholding to create binary images using OpenCV (`cv2.threshold`).
- **OBSERVATION:**
- The grayscale image is converted to black-and-white based on the threshold value, highlighting foreground vs background regions.

- 


## CODE 10: (code_10.py)

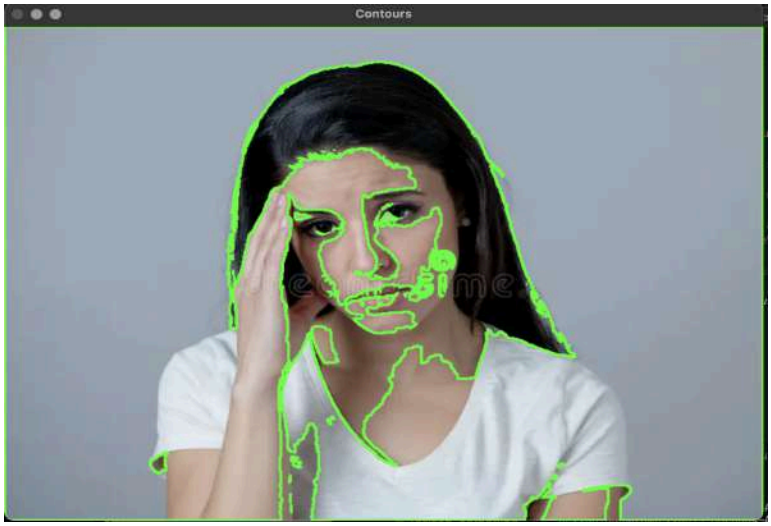
- **Concept used:** Edge detection using Canny algorithm (`cv2.Canny`).
- **OBSERVATION:**
- Only the edges of objects in the image are highlighted in white, while the rest remains black, showing the structure of shapes.
- 

## CODE 11: (code_11.py)

- **Concept used:**Face detection using Haar Cascade classifier in OpenCV (`cv2.CascadeClassifier`).
- **OBSERVATION:**
- Faces in the image are detected and highlighted with blue rectangles, showing automatic identification of human faces.
- 

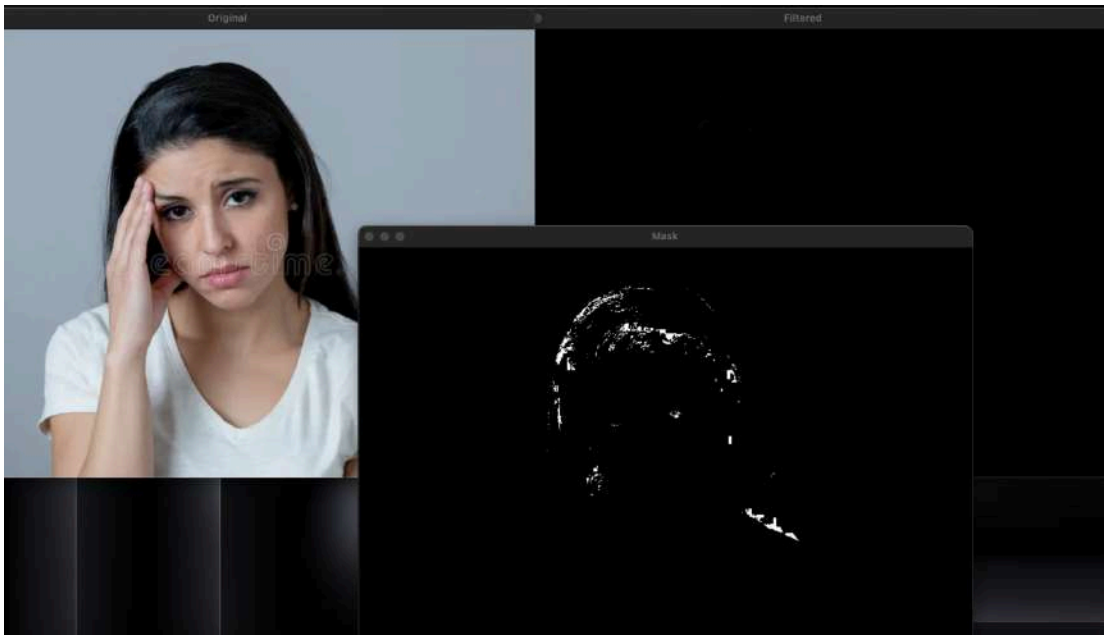

## CODE 12: (code_12.py)

- **Concept used:** Contour detection and drawing using OpenCV (`cv2.findContours`, `cv2.drawContours`).
- **OBSERVATION:**
- All detected contours in the thresholded image are outlined in green, highlighting object boundaries.

- 


## CODE 13: (code_13.py)

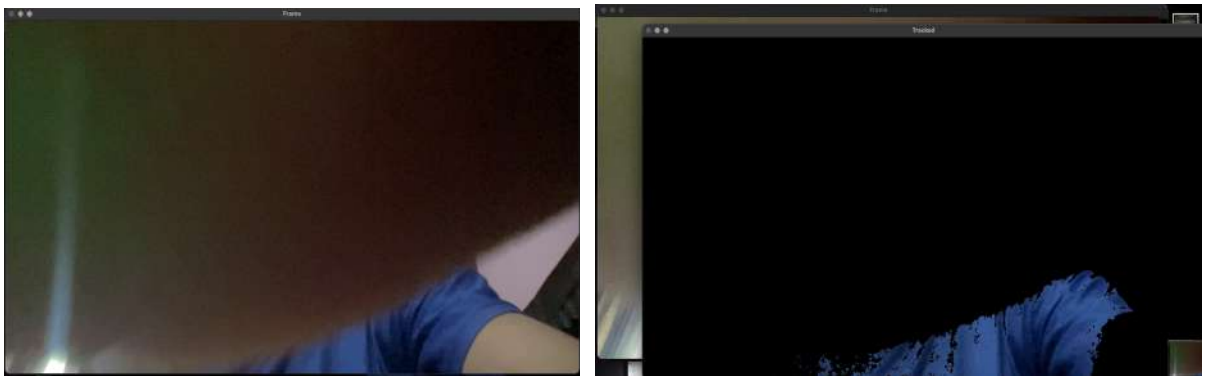
- **Concept used:** Color detection and masking using HSV color space in OpenCV (`cv2.inRange`, `cv2.bitwise_and`).
- **OBSERVATION:**
- Only the blue regions of the image are highlighted in the filtered result, while other colors are masked out.

- 

## CODE 14: (code_14.py)

- **Concept used:** Foreground extraction using GrabCut algorithm in OpenCV (`cv2.grabCut`)**.**
- **OBSERVATION:**
- The main subject inside the selected rectangle is isolated, while the background is removed or blacked out.
- 


## CODE 15: (code_15.py)

- **Concept used:** Real-time color detection and tracking using HSV masking in OpenCV.
- **OBSERVATION:**
- Live webcam feed highlights blue-colored regions, showing a mask and the filtered result in real time until user exits
- 

## CODE 16: (code_16.py)

- **Concept used:** Morphological operations (erosion and dilation) in OpenCV.
- **OBSERVATION:**
- The binary image shows shrunk objects after erosion and expanded/thicker objects after dilation, demonstrating noise removal and structure modification.
-